TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Agrarsystemtechnik

Multisensor Data Fusion in einem mobilen landtechnischen BUS-System für die Real-time Prozessführung in sensorgestützten Düngesystemen

Ralph Ostermeier

Vollständiger Abdruck der von der Fakultät Wissenschaftszentrum Weihenstephan für Ernährung, Landnutzung und Umwelt der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. J. Meyer

Prüfer der Dissertation:
1. Univ.-Prof. Dr. H. Auernhammer
2. Univ.-Prof. Dr. M. Faulstich

Die Dissertation wurde am 27.08.2012 bei der Technischen Universität München eingereicht und durch die Fakultät Wissenschaftszentrum Weihenstephan für Ernährung, Landnutzung und Umwelt am 15.04.2013 angenommen.

Im Selbstverlag: Ralph Ostermeier

Bezugsquelle: Technische Universität München
Lehrstuhl für Agrarsystemtechnik
Am Staudengarten 2
85354 Freising

Vorwort

Nach Abschluss dieser Dissertation bedanke ich mich bei allen sehr herzlich, die zum Gelingen dieser Arbeit beigetragen haben.

Mein ganz besonderer Dank gilt Herrn Prof. Dr. H. Auernhammer für die Überlassung des Themas und für die Gesamtbetreuung. Seine engagierte und motivierende Begleitung aller Phasen dieser Arbeit, die gewährten Freiräume sowie die wertvollen Hinweise und Anregungen zur Abfassung schätze ich sehr. Für seine Förderung beginnend mit meiner Diplomarbeit, bei meiner Tätigkeit als Mitarbeiter am Fachgebiet wie auch in meiner außeruniversitären Berufstätigkeit bin ich außerordentlich dankbar.

Weiterhin bedanke ich mich bei Herrn Prof. Dr.-Ing. M. Faulstich für die Übernahme des Koreferats und bei Herrn Prof. Dr. J. Meyer für die Übernahme des Prüfungsvorsitzes.

Danken möchte ich der Deutschen Forschungsgemeinschaft (DFG) für die Finanzierung des IKB-Dürnast-Projektes und allen Mitgliedern der DFG-Forschergruppe IKB-Dürnast für die enge und erfolgreiche Zusammenarbeit als ein Garant für das Gelingen dieser Arbeit.

Mein Dank gilt auch allen Kolleginnen und Kollegen am Lehrstuhl für Agrarsystemtechnik und dem Fachgebiet Technik im Pflanzenbau für die schöne und erlebnisreiche Zeit. Ich bleibe Ihnen freundschaftlich verbunden.

Abschließend möchte ich mich bei meinem derzeitigen Arbeitgeber John Deere, vor allem bei Herrn Dr. G. Kormann und Herrn Dr. T. Engel, für die motivierenden Worte und die eingeräumte Flexibiltät bedanken, um Beruf und Abfassung der Arbeit in Einklang zu bringen.

Ein besonderer Dank gebührt meiner Familie und meinen Freunden.

Freising im August 2012 Ralph Ostermeier

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Abkürzungen

API	Application Programming Interface
BMBF	Bundesministerium für Bildung und Forschung
BUS	Binary Unit System
CAN	Controller Area Network
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CSDGM	Content Standard for Digital Geospatial Metadata
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
DAML	DARPA Agent Markup Language
DDI	Data Dictionary Identifier (Abk. gemäß ISO 11783)
DFG	Deutsche Forschungsgemeinschaft
DGPS	Differential Global Positioning System
DIN	Deutsche Industrie Norm
DLR	Deutsche Zentrum für Luft- und Raumfahrt e. V.
DMS	Dehnungsmessstreifen
DSS	Decision support systems
DTM	Digital Terrain Modeling
ECU	Electronic Control Unit
EM	Elektro-magnetisch
EM38	Bezeichnung eines Messgeräts der Fa. Geonics Limited zur Messung der scheinbaren elektrischen Leitfähigkeit von Böden
E/E/PES	Elektrisch/elektronisch/programmierbare elektronische Systeme
FMIS	Farm Management Information System
GHz	Gigahertz
GIS	Geo-Informationssystem
GPR	Ground Penetrating Radar
GPS	Global Positioning System
GUI	Graphical User Interface
HCI	Human Computer Interface
HTML	Hypertext Markup Language
I/O	Input/Output
IEEE	Institute of Electrical and Electronics Engineers
IKB Dürnast	Informationssystem Kleinräumige Bestandesführung Dürnast

IL	Implementation Level
IMU	Inertial Measurement Unit
IR	Infrarot
IS	International Standard
ISO	International Organization for Standardization
ISOBUS	Markenbezeichnung für die internationale Norm ISO 11783
IT	Informationstechnologie
JDL	Joint Directors of Laboratories
JDK	Java Development Kit
JESS	Java Expert System Shell
JOE	Java Oriented Editing
bit/s	Einheit Übertragungsgeschwindigkeit über eine digitale (Kommunikations-)Strecke
KNN	Künstliches Neuronales Netz
LBS	Landwirtschaftliches BUS-System
M2M	Maschine-zu-Maschine
MICS	Mobile Implement Control System
MIMO	Multi Input Multi Output
MIS	Management Information System
MMI	Man Machine Interface
MS	Microsoft
MSDF	Multisensor Data Fusion
MOE	Measures of Effectiveness
MOP	Measures of Performance
N	Stickstoff
NDVI	Normalized Difference Vegetation Index
NIRS	Nahinfrarot-Spektroskopie
NMEA	National Marine Electronics Association
NMR	Nuclear Magnetic Resonance
NP	Knotenprozessor (Node Processor) für (Multisensor) Data Fusion
NWI	New Work Item
OSI	Open System Interconnection
OWL	Web Ontology Language
PDGPS	Phase Differential Global Positioning System oder in dt. auch Präzises DGPS
PDNACK	Process Data Negative Acknowledge

PGN	Parameter Group Number
REIP	Red Edge Inflection Point
RFID	Radio Frequency Identification
RIF	Rule Interchange Format
RS 232	Standard der Electronic Industries Association als Recommended Standard unter der Nummer 232
SAR	Synthetic Aperture Radar
SDK	Software Development Kit
SNR	Signal to Noise Ratio
SPN	Suspect Parameter Number
TECU	Tractor ECU (Abk.: gemäß ISOBUS)
TP	Teilprojekt
TU	Technische Universität
UAV	Unmanned Air Vehicles
UTM	Universal Transverse Mercator (Globales Koordinatensystem)
VRS	Virtual Reference Station
VRT	Variable Rate Technology
WED	Wissensentdeckung
W3C	World Wide Web Consortium
WGS84	World Geodetic System 1984
WSN	Wireless Sensor Networks
WZW	Wissenschaftszentrum Weihenstephan für Ernährung, Landnutzung und Umwelt der Technischen Universität München
XGA	Extended Graphics Array
XML	Extensible Markup Language

„Wir können überhaupt nicht denken,
ohne unsere fünf Sinne zu gebrauchen.“[1]
Albert Einstein (1943) [Cal11]

1 Einleitung

1.1 Hinführung

In Zeiten des Klimawandels eine wachsende Weltbevölkerung von geschätzten 9,5 Milliarden Menschen in 2050 nachhaltig mit ausreichend Nahrungsmitteln und zugleich mit nachwachsenden Rohstoffen für die stofflich-industrielle und energetische Nutzung zu versorgen, wurde von dem deutschen BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG (2010) [Bun10] in seinem Strategiepapier „BioÖkonomie 2030 – Unser Weg zu einer bio-basierten Wirtschaft“ als eine der größten Herausforderung des 21. Jahrhunderts identifiziert. Die Herausforderung hinsichtlich der Nachhaltigkeit wird darin gesehen, mit einer effizienten und standortangepassten agrarischen Produktion Luft, Böden und Gewässer, Klima und Ökosystem insgesamt wenig zu belasten, die biologische Vielfalt zu fördern sowie mit begrenzten Ressourcen wie Boden, Wasser und Nährstoffen verantwortungsvoll umzugehen. Zudem wird herausgestellt, dass der Aufbau einer wissensbasierten Bioökonomie nur mittels interdisziplinärer ganzheitlicher Forschungsansätze erreichbar sei, die wirtschaftliche, ökologische und gesellschaftliche Belange gleichermaßen berücksichtigen. Für den Handlungsbereich der Agrartechnik werden daher Weiterentwicklungen der Präzisionslandwirtschaft und eine entsprechende Verknüpfung mit den pflanzenbasierten Innovationen gefordert.

In den Kontext des aufgezeigten Themen- und Spannungsfeldes passt sich das bereits im Zeitraum 1998-2005 durchgeführte interdisziplinäre Forschungsvorhaben der DFG-Forschergruppe „Informationssystem Kleinräumige Bestandesführung – Dürnast (IKB-Dürnast)“ an der Technischen Universität München idealtypisch ein. Die Forschergruppe arbeitete mit dem Fokus auf der Stickstoff-Applikation nach der Hypothese [AOMSSW06]: „Kleinräumig differenzierte N-Düngung unter Berücksichtigung des pflanzenverfügbaren Bodenwassers und der N-Aufnahme in die Biomasse erhöht auf Basis des standortspezifischen Ertragspotentials die Ressourceneffizienz.“[2] Hierzu definierte die

[1] A. a. O., S. 337: *„No idea is conceived in our mind independent of our five senses“*

[2] A. a. O., S. 6

Forschergruppe ein für den Präzisen Ackerbau (*Precision Farming*) neues Prozessführungssystem Sensor-Ansatz mit Kartenüberlagerung (*Real-time approach with map overlay*). An sechs Lehrstühlen und Fachgebieten des WZW der TU München wurden die hierzu nötigen Untersuchungen in 12 Teilprojekten durchgeführt. Die Teilprojekte wurden den vier Themengebieten „Boden und Wasser“, „Biomasse“, „Prozesssteuerung“ sowie „Ökonomie und Ökologie“ zugeordnet. Am Fachgebiet Technik im Pflanzenbau der TU München wurden unter Leitung von Prof. Auernhammer, der Projektinitiator war und auch als Sprecher der DFG-Forschergruppe fungierte, die Themenstellungen im Bereich der Prozesssteuerung im Rahmen von drei Teilprojekten bearbeitet. In dem definierten Forschungsansatz war es in der Verantwortung der Prozesssteuerung, die Informationsgewinnung in technischen Systemen, speziell im Bereich der Prozessdatenerfassung und der lokalen Information über den Ertrag, deren Bereitstellung für das Informationssystem und die Umsetzung des Ansatzes in ein Applikationssystem zu realisieren. Letzterer Aspekt war eines der beiden Teilziele des Teilprojektes 8 „Entwicklung und Test einer *Real-time*-Prozessführung für sensorgestützte Düngesysteme“. Dieses Teilziel wurde vom Autor während seiner Tätigkeit am Fachgebiet bearbeitet. Die daraus resultierenden Leitgedanken, die Modellbildung und die realisierte Simulation bilden die Grundlage der vorliegenden Arbeit.

1.2 Problemstellung

Agrarelektronik nimmt bei der Umsetzung von Prozesssteuerungen in der Außenwirtschaft eine zentrale Rolle ein. Dabei wird der Stand der Technik und die Weiterentwicklung der Agrarelektronik bei gegebener Applikationstechnik von den drei Hauptsäulen Rechnertechnik (*Computing*), Kommunikationstechnik (*Communication*) und Sensortechnik (*Sensors*) bestimmt. Bei Einzug der Elektronik in mobile landtechnische Prozesssteuerungen und der anschließenden Fortentwicklung in den achtziger Jahren des letzten Jahrhunderts war die Leistungsfähigkeit noch geprägt von der zentralen Rechnereinheit einer Insellösung oder eines mobilen Agrarcomputers mit direkt analog angebundenen Sensoren und Aktoren. Hingegen wird die Entwicklung der letzten 15 Jahre zunehmend von verteilten Lösungen dominiert, bestehend aus über Bussysteme vernetzten Rechnereinheiten, sogenannten elektronischen Steuereinheiten (ECU), um dem wachsenden Aufgabenspektrum und den damit gestiegenen Leistungsbedarf gerecht zu werden. Diese Entwicklung ist vergleichbar bzw. wurde angelehnt an die Entwicklung und Einführung der Bussysteme in die Kraftfahrzeugtechnik. Im Unterschied zum Kraftfahrzeug ist jedoch bei einer Traktor-Geräte-

Kombination stets auch die Vernetzung und Kommunikation mit an- und abkoppelbaren Anbaugeräten unterschiedlichster Hersteller sowie eine Anbindung an das Betriebsmanagementsystem zu realisieren. Landtechnische bzw. landwirtschaftliche BUS-Systeme sind die Antwort auf dieses Anforderungsprofil.

Das im Titel dieser Arbeit gewählte Attribut mobil stellt die Abgrenzung zu den Prozesssteuerungssystemen der Innenwirtschaft heraus. Angetrieben durch die Erkenntnis, dass nur eine standardisierte Lösung entscheidende Vorteile für den Nutzer und Kunden dieser Technik bringt, wurde 1987 mit der Standardisierung des Landwirtschaftlichen BUS-Systems LBS begonnen und als nationale Norm DIN 9684/2-5 [DIN9684] veröffentlicht. Nach Festlegung der ersten wichtigen Protokolldefinitionen von LBS wurde schon 1991 die Erweiterung in Richtung eines internationalen Standards durch Einbringung von LBS in die ISO-Normung initiiert [Aue93], [SMFB99]. Die daraus entstandene Norm ISO 11783 [ISO11783], am Markt als ISOBUS bezeichnet [AS06], ist die standardisierte Form eines landwirtschaftlichen Bussystems, das auf internationaler Ebene stetig an Bedeutung gewinnt und wachsende Akzeptanz erfährt. Jedoch liegen derzeit noch nicht alle Teile der ISO 11783 in vollständig verabschiedeter Form vor. Zugleich wird mit der Definition neuer Teile und Ergänzungen auf neue Anforderungen reagiert. Wie weit die einzelnen Landmaschinen- und Agrarsoftwarehersteller eigene Protokolldefinition zugunsten von ISBOUS komplett aufgeben bzw. auf diesen Standard migrieren, ist aufgrund fehlender Veröffentlichung der Implementierungsdetails schwierig zu beurteilen. Jedoch ist in Europa, gefolgt von Nordamerika, ein Trend erkennbar, dass ISOBUS-Kompatibilität bei Traktoren, Geräten und Systemlösungen stark beworben wird.

In einem landwirtschaftlichen BUS-System reicht die Palette an Sensortechnik von Sensoren, die analog an ECU's angeschlossen werden, bis hin zu hochentwickelten Sensoren mit eigener Rechnertechnik und digitalem Kommunikationsinterface, wie beispielsweise einem GPS-Empfänger. Im Hinblick auf sogenannte Online-Sensorik, die den Ernährungszustand der Pflanzen messen und in Echtzeit die auszubringende Nährstoffmenge anpassen können, wird mittlerweile sogar von einer „Sensorschwemme“[3] [Rec10] gesprochen. Sensortechnik und die Interaktion mit der Bedienperson stellen jedoch nicht die alleinige Daten- und Informationsquelle dar. Über die Anbindung an das stationäre Betriebsmanagement, dem *Farm Management Information System (FMIS),* wird ein breites Angebot von Daten- und Wissensarten erschlossen und (indirekt) auf dem mobilen System verfügbar gemacht.

[3] A. a. O., S. 81

Dokumentationslösungen und automatisierte Prozessdatenerfasssung auf dem mobilen System realisieren dann einen Informationszuwachs, der auf das FMIS zurückfließt. Über Telematiklösungen wird bereits der Weg zur drahtlos in die globalen Telekommunikations- und Datennetzwerke eingebundenen Landmaschine aufgezeigt und beschritten.

Mit den landwirtschaftlichen BUS-Systemen einhergehend ist das Denken in Systemlösungen zur Realisierung der Überwachungs-, Dokumentations-, Steuerungs- und Regelungsaufgaben sowie deren Optimierung. Mit der in diesen Systemen wachsenden Verfügbarkeit von Sensoren sowie von weiterer lokal und entfernt vorgehaltener Information liegt es nahe, einen Informationsgewinn und robustere oder optimierte Lösungen über die Berücksichtigung der verschiedenen Informationsquellen und Fusion der zugehörigen Daten anzustreben. Die im System verfügbaren Sensordaten nehmen folglich eine entscheidende Rolle ein, jedoch ist die Fusion keinesfalls darauf begrenzt. *Multisensor Data Fusion* (Abk.: MSDF) in einem mobilen landwirtschaftlichen BUS-System umschreibt die Problematik und den daraus abgeleiteten Ansatz und Fokus dieser Arbeit daher am treffendsten.

Derzeit ist die Verrechnung der GPS-Position mit den Messwerten von Inertialsensoren zur Berücksichtigung der Wank- und Nickbewegungen der Landmaschine Stand der Technik. Meist wird diese Fusion bereits als integrierte Lösung in einem GPS-Empfänger ausgeführt und repräsentiert eine überschaubare Lösung. Beruhen Fusionslösungen jedoch auf der Einbeziehung von Daten mehrerer Systemeinheiten, so führt dies zwangsläufig zu einer steigenden Komplexität, bei deren Bewältigung strukturierte Analyse-, Entwurfs- und Entwicklungsprozesse helfen. Im Bereich der Informationstechnik für Prozesssteuerungen des Außenbereiches zielen bisherige Arbeiten vor allem auf die Gewährleistung der funktionalen Sicherheit [Mar04], [HH07], auf *Open Source*-Software [SAD01] sowie auf agile Softwareentwicklung [LM05] ab. Landtechnische Sensorfusionsansätze wurden hauptsächlich im Zusammenhang mit automatischen Lenksystemen oder bei der Multisensorentwicklung für die Zustandsüberwachung, dem *Condition Monitoring*, untersucht, ohne speziell auf landwirtschaftliche BUS-Systeme (und durchgängige Analyse- und Entwicklungsprozesse) einzugehen. Interessant ist jedoch die Aussicht, die KRALLMANN (2005) [Kra05] über den Einsatz eines Multisensors für ein *Condition Monitoring* von mobilen Arbeitsmaschinen ausspricht: „…. In Zukunft werden weitere Multisensor-Systeme auf mobilen Arbeitsmaschinen eingesetzt. Diese können das *Condition Monitoring*-System oder die Steuer- und Regelungssysteme unterstützen. Dabei kommen nicht nur neu entwickelte Multisensoren zum Einsatz; es kann sich auch um bereits vorhandene, auf den Maschinen

verteilte Sensoren handeln. …“[4]

Genau an dieser Stelle der Berücksichtigung von Steuer- und Regelungsaufgaben vor dem Hintergrund zukünftiger aber auch bereits existierender verteilter Sensoren und Information setzt diese Arbeit an.

1.3 Hinweise zur Textgestaltung

Vorab sei auf die Verwendung von englischen Fachbegriffen hingewiesen, da für weite Bereiche des Themas Englisch die vorherrschende Wissenschafts-, Normierungs- und Implementierungssprache ist. Dabei wird hauptsächlich dem Vorgehen von BALZERT (2000)[5] [Bal00] in seinem Lehrbuch über Software-Technik gefolgt. Falls sich für einen Fachbegriff noch kein eingebürgerter deutscher Begriff etabliert hat, dann wird der englische Originalbegriff in kursivem Schriftformat verwendet. Wird ein deutscher Fachbegriff verwendet, bei dem der englische Begriff nicht direkt ersichtlich oder eine unmittelbare Zuordnung zu englischsprachigen ISO-Normfestlegungen nicht gegeben ist, so wird die englische Bezeichnung kursiv in Klammern angefügt.

In gleicher Weise wird in den Kapiteln zur Realisierung der Simulation immer wieder auf englische Bezeichnungen zurückgegriffen, da auch die Implementierung der Simulation in englischer Sprache ausgeführt wurde. Wird in dieser Arbeit ein Codeausschnitt angeführt, so wird ein deutscher Erklärungstext beigestellt. Wiederkehrende Bezeichner oder Variablennamen werden einmal eingeführt und dann nicht übersetzt durchgängig verwendet. Ebenso werden Codesegmente, Variablen und Pseudo-Code stets durch geänderte Formatierung kenntlich gemacht.

Auch in Sachen weiblicher und männlicher Anrede folgt diese Arbeit der Wahl von BALZERT (2000)[6] [Bal00]. Wird im Text die maskuline Form einer Person oder einer Anrede benutzt, wie z.B. Nutzer, Landwirt oder Leser, so ist sowohl das weibliche als auch das männliche Geschlecht gemeint.

Ein zusätzlicher Hinweis fokusiert sich auf die Zitation. Den allgemeinen wissenschaftlichen Regeln folgend wird bei wörtlichen Zitaten in einer Fußnote die Seitenangebe zur verwendeten Quelle genannt. Zusätzlich erfolgt an einigen Literaturstellen ein integrierter Seitenhinweis, wenn die getätigten Aussagen in einem mehr oder weniger direkten

[4] A. a. O., S. 116

[5] A. a. O., S. 17-18

[6] A. a. O., S.13

Zusammenhang mit wichtigen Ergebnissen, bzw. Erkenntnissen der genannten Autoren stehen.

2 Theoretische Betrachtungen und Stand des Wissens

2.1 Theorie der Multisensor Data Fusion

2.1.1 Grundlagen, Anwendungen, Motivation

Die Fähigkeit verschiedene Sinne und Informationsquellen für eine Entscheidungsfindung zu nutzen, ist nicht neu und wird vom Menschen oder im Tierreich schon über Jahrtausende erfolgreich bei der Bewältigung des alltäglichen Lebens und im Überlebenskampf angewandt. So ist das Interesse an industrieller Anwendung und wissenschaftlicher Erforschung der Techniken, Daten von unterschiedlichen Sensoren und weitere zugehörige Informationen zu fusionieren, mit dem Ziel einer besseren Informationsableitung und Entscheidungsfindung zu erreichen bzw. erst zu ermöglichen, als durch den Einsatz nur eines Sensors, nur folgerichtig. Dabei verstehen HALL UND MCMULLEN (2004) [HM04] unter einem MSDF-System ein System, wenn Daten aus mehreren Quellen miteinander in Verbindung gebracht werden. Diese Daten können von verschiedenen Sensoren stammen, daher auch der Begriff Multisensor, wie auch von Nutzern eingegeben werden oder bereits als Information in Datenbanken gesammelt und abgelegt sein. Daraus ist bereits ersichtlich, dass es sich im engeren Sinn nicht nur um *Sensor Fusion*, sondern um die übergreifende Disziplin der *Data Fusion* handelt.

Die Anwendungen sind mittlerweile weitreichend und erstrecken sich über militärische und zivile Anwendungen. HALL UND LLINAS (2001) [HL01] listen für den militärischen Bereich folgende repräsentative Beispiele auf:

- *Ocean Surveillance,*
- *Air-to-air and surface to air defense,*
- *Battlefield intelligence, surveillance and target acquisition,*
- *Strategic warning and defense.*

Über zivile Anwendungsfelder gibt Tabelle 2.1 einen repräsentativen Überblick und zeigt zugleich für die einzelne Anwendung auch deren Charakteristika unterschieden nach Deduktion/Ableitung, primäre beobachtbare Daten, Größe des Beobachtungsraumes und Sensorposition.

Tabelle 2.1: Beispiele ziviler MSDF-Anwendungen [HL01], [Kra05]

Anwendung	Deduktion/Ableitung	Primär beobachtbare Daten	Größe des Beobachtungs-raumes	Sensor-Platzierung
Zustandsabhängige Wartung	Erkennung und Charakterisierung von Systemfehlverhalten; Vorschläge für Wartung bzw. Korrekturen	EM-Signale; Akustische Signale; Magnetische Felder; Temperaturen; Röntgenstrahlen	Mikroskopisch bis einige hundert Meter	Schiffe; Flugzeuge; Stationär (z.B. Fabriken)
Robotik	Objekterkennung; Objektlokalisierung; Bewegungssteuerung des Roboters (z.B. „Hände“ und „Füße“)	Maschinelles Sehen; Akustische Signale; EM-Signale; Röntgenstrahlen	Mikroskopisch bis zu einigen Metern im Umkreis des Roboters	Roboter
Medizinische Diagnose	Lokalisierung und Identifizierung von Tumoren, Abnormitäten und Krankheiten	Röntgenstrahlen; NMR; Temperatur; IR; Visuelle Untersuchung; Chemische und biologische Daten	Menschlicher Körper	Labor
Umweltbeobach-tung und -erkundung	Identifizieren und Lokalisierung von Naturphänomenen (z.B. Erdbeben, Wetter, …)	SAR; Seismische Daten; EM-Strahlung; Bodenproben; Chemische und biologische Daten	Hunderte von Kilometern	Satelliten; Flugzeuge; Bodenstationen; Unterirdische Beprobungs-stellen

Dieses weitreichende Anwendungsgebiet ist durch eine Reihe von Vorteilen motiviert. *Data Fusion* schöpft die Synergie, angeboten von Information aus unterschiedlichen Quellen, aus. Die Kombination von zusätzlichen unabhängigen und/oder redundanten Daten resultiert in der Regel in einer Verbesserung von Ergebnissen, von System-Robustheit und Zuverlässigkeit. Zusätzlich bietet es eine Lösung bei der Bewältigung der momentanen Explosion von verfügbarer Information. WALTZ UND LLINAS (1990) [WL90] stellt folgende neun Vorteile, wie in Tabelle 2.2 aufgelistet, fest.

Diese Tabelle macht deutlich, dass ein großer Teil der Motivation für MSDF-Techniken militärischen Hintergrund besitzt, mit den zentralen Aufgabenstellungen, wie der Objektverfolgung, der Freund-Feind-Erkennung und der „Schlachtfeldüberwachung und -intelligenz“. Jedoch ist auch jeder einzelne Vorteil gerade für komplexe zivile Anwendung entscheidend für die Leistungserbringung und Effizienz des Systems.

Tabelle 2.2: Vorteile der MSDF (nach [WL90])

Vorteil	Bedeutung
Stabile Funktion im Betrieb	Ein Sensor kann Informationen liefern, während die anderen Sensoren nicht verfügbar oder gestört sind, oder räumlich zu weit entfernt von einem Objekt sind oder zeitlich mit einem Ereignis übereinstimmen.
Erweiterte räumliche Abdeckung	Ein Sensor kann den Raum beobachten, der von anderen Sensoren nicht einsehbar ist.
Erweiterte zeitliche Abdeckung	Ein Sensor kann Ziele oder Ereignisse entdecken bzw. erfassen, wenn andere Sensoren dazu nicht in der Lage sind.
Höhere Konfidenz	Ein oder mehrere Sensoren können das gleiche Ziel oder Ereignis bestätigen.
Reduzierte Mehrdeutigkeiten	Gemeinsame Informationen von mehreren Sensoren reduzieren die Anzahl an Hypothesen über ein Ziel oder Ereignis.
Verbesserte Entdeckung	Die effektive Integration von mehreren Messungen erhöht die Entdeckungssicherheit eines Ziels oder Ereignisses.
Erweiterte räumliche Auflösung	Mehrere Sensoren können eine synthetische Auflösung erzeugen, welche die Auflösung eines einzelnen Sensors übersteigt.
Höhere Systemzuverlässigkeit	Multisensor-Systeme besitzen eine inhärente Redundanz.
Erweiterte Dimensionalität	Ein System, dass unterschiedliche Sensoren zur Beobachtung unterschiedlicher physikalischer Phänomene zur Anwendung bringt, ist weniger anfällig für die Störung durch eine feindliche Aktion oder durch ein natürliches Phänomen.

So unterscheiden sich die grundlegenden Basistechniken für MSDF auch nicht und sind abgeleitet bzw. entliehen aus etablierten Disziplinen wie Signalverarbeitung, Statistik, Mustererkennung, Künstlicher Intelligenz, Kognitiver Psychologie und Informationstheorie [HL01].

HALL UND MCMULLEN (2004) [HM04] weisen darauf hin, dass in der 1992 erschienenen Erstausgabe ihres Buches noch festgestellt wurde: *„.... Data fusion is not a discipline in the same sense as more well-defined studies such as signal processing or numerical methods. Well defined techniques, terminology, and a professional community do not yet exist. ...“*[7]. Etwas mehr als ein Jahrzehnt später bescheinigen sie, dass *Data Fusion* zwar noch keine ausgereifte Fachdisziplin ist, aber wesentliche Fortschritte bei der Entwicklung von Modellen, einer Terminologie, der Erarbeitung von Systems Engineering Methoden für die Implementierung von *Data Fusion*-Systemen und der Etablierung einer Gemeinschaft von Wissenschaftlern, Entwicklern und Anwender von *Data Fusion* erzielt wurden [HM04].

Die Ergebnisse dieser Arbeiten werden in den nachfolgenden Kapiteln vorgestellt.

[7] A. a. O., S. 2

2.1.2 Definition von Data Fusion

Eine der Hauptanstrengungen des Mitte 1986 vom Verteidigungsministerium der USA gegründeten Expertengremiums „*US Joint Directors of Laboratories (JDL) Data Fusion Working Group*“ war die Erarbeitung einer einheitlichen Terminologie für *Data Fusion* mit der Zielsetzung, die Kommunikation unter Forschern und Systementwicklern des US-Militärs zu verbessern. Das erarbeitete *JDL Data Fusion*-Lexikon formuliert folgende Definition für *Data Fusion* [SB01]:

„A process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and assessments, and the evaluation of the need for additional sources, or modification of the process itself, to achieve improved results.“ [8]

Der militärische Ursprung ist unverkennbar und die Definition damit zu restriktiv für die Anwendung auf andere weite Anwendungsgebiete der Ingenieurwissenschaften, die sich mit denselben zugrundeliegenden Fragestellungen der *Data association and combination* beschäftigen. STEINBERG UND BOWMAN (2001) [SB01] schlagen daher folgende umfassende Definition vor:

„Data fusion is the process of combining data or information to estimate or predict entity states.“ [9]

Gebräuchlich ist der Begriff MSDF, falls wenigstens ein Teil der Daten von Sensoren stammt.

2.1.3 Sensoren und Sensordaten

Sensoren und Sensordaten sind sehr wesentliche Elemente in einem MSDF-System. Dieser Abschnitt gibt einen Überblick über Definitionen und Klassifizierung.

Nach der von FRADEN (2004) [Fra04] angeführten Definition ist ein Sensor ein Bauteil, das ein Signal oder Stimulus empfängt und darauf antwortet. Da diese Definition jedoch äußerst weit gefasst ist, schlägt er auch eine wesentlich enger gefasste Definition vor: *„A sensor is a device that receives a stimulus and responds with an electrical signal.“*[10] Dem liegt der Grundgedanke der Aufteilung der Welt in natürliche und vom Menschen erstellte Objekte zugrunde. Während natürliche Sensoren, so wie sie in Lebewesen gefunden werden, in der

[8] A. a. O., S. 3

[9] A. a. O., S. 4

[10] A.a. O., S. 2

Regel mit elektrochemischen (Ionen-Transport wie in Nervenbahnen) Signalen reagieren, wird in vom Menschen erstellten Systemen Information vorherrschend in elektrischer Form übermittelt und verarbeitet. Daraus folgt auch, dass Sensoren, die in der Regel nur eine Unterkomponente eines technischen Gesamtsystems sind, über eine derartig angepasste Schnittstelle verfügen. Der Stimulus ist eine Größe, eine Eigenschaft oder ein Zustand, der erfasst und in ein elektrisches Signal umgewandelt wird. Klar abzugrenzen ist der Begriff eines Wandlers (*transducer*). Ein Wandler ist ein Gerät, das eine Energieform in eine andere umwandelt. Wandler können somit Bestandteile eines komplexeren Sensors sein. Abbildung 2.1 bildet obige Ausführungen in ein allgemeines Sensormodell ab.

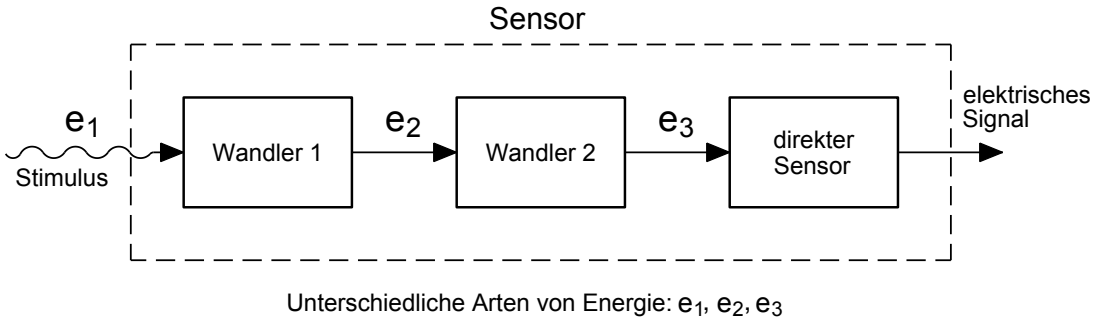


Abbildung 2.1: Sensormodell (nach [Fra04])

Auch diese allgemeine Definition für einen Sensor ist noch sehr weit gefasst und bietet ausreichend Raum für eine weitere Unterscheidung bzw. Klassifizierung von Sensoren. Die Bandbreite der Klassifizierungsschemata ist sehr weitreichend. Je nach Intention werden sehr einfache oder komplexe Einordnungskriterien gewählt.

Hinsichtlich der Energieverwendung lassen sich Sensoren in aktive oder passive Sensoren unterscheiden. Ein passiver Sensor benötigt keine Zusatzenergie und erzeugt direkt ein elektrisches Signal als Reaktion auf einen externen Stimulus. Ein aktiver Sensor hingegen benötigt eine Zusatzenergie, auch *excitation signal* genannt. Dieses *excitation signal* wird durch Sensorparameter entsprechend des Stimulus moduliert, worin sich Information des Messwertes abbildet.

Weiterhin können Sensoren in absolute oder relative unterteilt werden. Ein absoluter Sensor detektiert den Stimulus in Referenz zu einer absoluten physikalischen Skala unabhängig von den Messbedingungen, während ein relativer Sensor ein Signal erzeugt, das nur in Beziehung zu den Messbedingungen interpretiert werden kann, wie z.B. eine Temperaturdifferenz.

Eine wesentlich breiter angelegte Differenzierung ist die Unterscheidung nach verschiedenen Sensoreigenschaften wie sie FRADEN (2004) [Fra04] in sechs Tabellen zusammengestellt hat.

Die sechs zugrundeliegenden Eigenschaften sind die Art des Stimulus, die technischen Sensorspezifikationen, das Sensormaterial, die *Detection Mean* im Sensor, das genutzte Konversionsphänomen und das Anwendungsgebiet.

Aus Sicht der Mechatronik klassifiziert ISERMANN (2006) [Ise06] Sensoren nach den wichtigsten Messgrößenklassen, wie mechanische Größen, thermische Größen, elektrische Größen und die chemischen und physikalischen Größen. Des Weiteren gibt er auch noch eine Einteilung der elektrischen Ausgangssignale an, in amplitudenmodulierte, in frequenzmodulierte und den digitalen Signalformen an.

Laut AUERNHAMMER (1991) [Aue91] lassen sich verallgemeinernd in der Landwirtschaft gebräuchliche Sensoren nach der Sensortechnik in vier Gruppen einteilen. Neben *mechanischen Sensoren* haben die *elektrotechnischen Sensorelemente* das größte Einsatzfeld, von zunehmender Bedeutung sind Sensoren auf Mikrowellenbasis und auch die höheren Formen der *Optoelektronik*. Einer weiteren Gruppe, *den biotechnischen Sensoren,* sagt er ebenfalls eine wachsende Bedeutung voraus. Eine Übersicht zum aktuellen Stand der Technik im Gebiet der Biosensoren findet sich bei [Li06].

Aus Sicht der Militärtechnik geben WALTZ UND LLINAS (1990) [WL90] einen Überblick über die Bandbreite von Sensoren, die für *Data Fusion* verfügbar sind. Sie klassifizieren dabei nach detektierbarer Charakteristik, dem zugehörigen Spektralbereich und ordnen entsprechende Sensoren bzw. Sensorsysteme zu. Als detektierbare Charakteristika werden akustische Frequenzen, Elektromagnetismus, Infrarotstrahlung, sichtbares Licht, ultraviolettes Licht, nukleare und nichtnukleare Partikel aufgeführt.

Ausgehend von der militärischen *Data Fusion*-Sichtweise schlagen HALL UND MCMULLEN (2004) [HM04] ein generisches Sensormodell vor. Auch wenn dieses Modell sich sehr stark an der militärischen Anwendung orientiert, bildet es die Essenz obiger Klassifizierungsschemata ab und schenkt auch externen und internen natürlichen Störgrößen sowie einer bewussten Störung des Sensorbetriebs Aufmerksamkeit. Außerdem erweitern sie das Modell um das Konzept eines „Intelligenten Sensors“ (s*mart sensor*). Dabei umfasst der Sensor neben den Stimulus-Empfangselementen auch noch Einheiten zur Signalkonditionierung, der Signalverarbeitung, der Informationsverarbeitung und Entscheidungsfindung, sowie der Ausgangssignalverarbeitung. Dieses System wird durch eine Sensorkontrollinstanz vervollständigt.

2.1.4 Funktionale Modelle

2.1.4.1 JDL Modell und Erweiterungen

Einleitend folgt eine kurze Darstellung des Nutzen und der Rolle von Modellen und Architekturen im Rahmen der Systemtechnik. Laut STEINBERG UND BOWMAN (2001) [SB01] ist eine Architektur nach IEEE Definition eine *„.... structure of components, their relationships, and the principles and guidelines governing their design and evolution over time. ....“*[11] Daraus lassen sich die folgenden Anforderungen an eine Architektur ableiten:

- Identifizierung eines fokussierten Zwecks,
- Erleichterung des Verständnisses und der Kommunikation von Anwendern,
- Ermöglichung von Vergleichen und Integration,
- Förderung von Erweiterbarkeit, Modularität und der Möglichkeit zur Wiederverwendung,
- Förderung einer kosteneffizienten Systementwicklung,
- Anwendbarkeit für die geforderte Breite von Situationen.

Modelle wiederum sind ein Element einer Architektur. Ein Modell ist eine abstrakte Beschreibung eines Sets von Funktionen und Prozessen, die Komponenten eines Systems eines bestimmten Typs sein können, ohne Indikation von Software oder physikalischer Implementierung.

STEINBERG UND BOWMAN (2001) [SB01] unterscheiden drei verschiedene Modelltypen, ein funktionales Modell, ein Prozess bzw. prozedurales Modell und ein formales Modell. Danach ist ein funktionales Modell ein Set von Definitionen und Funktionen, die ein System ausmachen können. Ein Prozessmodell hingegen spezifiziert die Interaktion zwischen Funktionen innerhalb eines Systems. Hingegen besteht ein formales Modell aus einem Set von Axiomen und Regeln zur Manipulierung von Entitäten.

Grundsätzlich soll ein Modell die verschiedenen Aspekte der Probleme und der Lösungsansätze erhellen, um Gemeinsamkeiten der Probleme und der Lösungsmöglichkeiten leichter erkennen zu können. Des Weiteren sollen Modelle die Verständigung und Zusammenarbeit von Managern, Theoretikern, Entwicklern, Tester und den Anwendern von *Data Fusion*-Systeme fördern sowie ein kosteneffektives Systemdesign, Entwicklung und Betrieb positiv unterstützen.

Ein dem Stand der Technik entsprechendes Funktionales Modell für *Data Fusion* wurde von

[11] A. a. O., S. 4

der „*U.S. Joint Directors of Laboratories (JDL) Data Fusion Group*“ spezifiziert [SB01]. Nach Einschätzung der beiden Autoren ist es das am weitesten etablierte funktionale Modell für *Data Fusion*. Jedoch verrät allein die gewählte Terminologie schon den Fokus auf die militärische Anwendung. Mit ihrem Vorschlag für ein „*Revised JDL data fusion model*“ schlagen STEINBERG ET AL. (1998) [SBW98] eine überarbeitete Version des ursprünglichen *JDL data fusion model* vor, das dem Modell die militärische Ausrichtung nimmt und damit den Ansprüchen der gesamten weltweiten *Data Fusion*-„*Communities*“ genügt. Abbildung 2.2 illustriert dieses funktionale Modell.

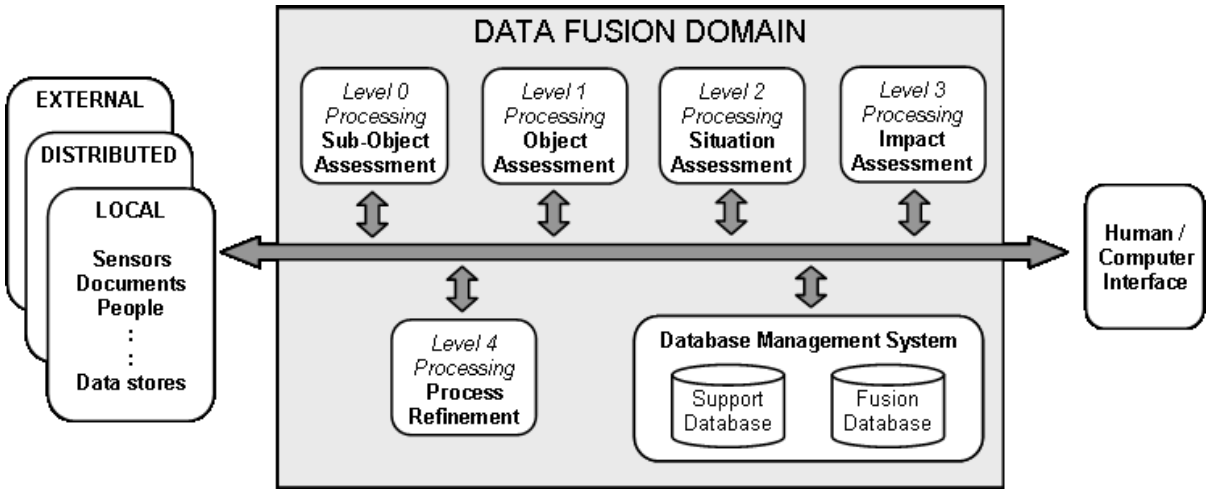


Abbildung 2.2: *Revised JDL data fusion model* [SBW98]

Auf der linken Seite des Modells versorgen unterschiedliche Daten- und Informationsquellen den Fusionsprozess mit Eingangsinformationen. Diese Quellen können Sensordaten, oder Datenbankinformationen, Ausgangsprodukte anderer *Data Fusion*-Prozesse oder von „Intelligenten Sensoren“ sowie auch Ein- bzw. Vorgaben von menschlichen Nutzern sein. Diese Informationsquellen können sowohl systemintern, jedoch auch weitreichend extern verteilt und angeordnet sein.

Auf der rechten Seite des Modells wird mit dem HCI die Schnittstelle zum Nutzer geschaffen, über dessen Dienste die Resultate der Fusionsprozesse dem Nutzer angezeigt werden. Anderseits kann der Nutzer über jene Schnittstelle seine Eingaben tätigen wie auch Steuerungsfunktionen über den Fusionsprozess ausüben.

Im Bereich des eigentlichen Fusionsprozesses, der *Data Fusion*-Domäne, differenziert das Modell fünf Verarbeitungsebenen (*processing levels*) auf der Basis von Typen eines Bewertungs- bzw. Beurteilungs-Prozesses, der in etwa mit den Typen von Einheiten bzw. Instanzen, für die ihr Zustand abzuschätzen ist, korrespondiert:

Level 0 Processing - Sub-Object Assessment:
„Estimation and prediction of signal- or object-observable states on the basis of pixel/signal level data association and characterization"[12] [SB01].
Basissignalverarbeitung ist die Grundvoraussetzung für Fusionsaktivitäten auf allen nachfolgenden Ebenen. Signale und messbare Merkmale einer Entität müssen entdeckt, extrahiert, gefiltert und räumlich und zeitlich auf Signal/Pixel-Ebene ausgerichtet und abgestimmt werden.

Level 1 Processing - Object Assessment:
„Estimation and prediction of entity states on the basis of inferences from observations"[13] [SB01].
Diese Ebene setzt sich aus *Data Alignment*, Daten/Objekt-Korrelation, Objekteigenschaften und örtlicher und kinematischer Bewertung/Abschätzung und Objektidentifizierung zusammen:

- *Data Alignment*:
  Daten von verschiedenen Informationsquellen müssen im Raum (z.B. gemeinsames Koordinatensystem WGS 84) und in der Zeit oder in einem anderem Merkmalsraum oder -referenzsystem (z.B. Frequenz) ausgerichtet und aneinander angepasst werden. Zeitliches *data alignment* schließt die Verrechnung von Differenzen in der Beobachtungsfrequenz und den Zeitverzögerungen zwischen der physikalischen Beobachtung und dem Dokumentations- bzw. Auswertezeitpunkt ein. *Data Alignment* ist auch Angleichung bzw. Anpassung der Einheiten.

- *Data/Object correlation (association)*:
  Data/Object correlation umfasst ein Zuordnungsproblem, bei dem Beobachtungen mit anderen Beobachtungen in Verknüpfung gebracht werden. Im Allgemeinen werden verschiedene Sensoren genutzt, um mehrere Objekte und Teile oder verschiedene Attribute eines Objektes zu untersuchen oder zu beobachten. Vor der eigentlichen Fusion ist es unerlässlich zu bestimmen, welche Beobachtung zu welchem Objekt oder Objektattribut gehört. Neue Daten müssen mit zeitlich weiter zurückliegenden Daten korreliert werden, um zeitliche oder räumliche Veränderungen entdecken und segmentieren zu können.

[12] A. a. O., S. 6
[13] A. a. O., S. 6

- *Object attribute, positional and kinematic estimation:*
  Diese Funktionalität bedingt, dass Sensordaten und Kontextdaten zusammen fusioniert werden, um eine Abschätzung über die Entitäts- bzw. Objekteigenschaften zu treffen sowie deren Position und Geschwindigkeit zu bestimmen, die am besten zu den beobachteten bzw. gemessenen Daten passen.

- *Object identity estimation:*
  Die Bestimmung der Identität beteiligter Objekte.

Level 2 Processing - Situation Assessment:

„Estimation and prediction of entity states on the basis of inferred relations among entities"[14] [SB01].

Diese Ebene umfasst die Abschätzung oder Beurteilung und Vorhersage von Entitätszuständen auf der Basis abgeleiteter organisatorischer, kausaler, biologischer und räumlich-zeitlicher Beziehungen zwischen Objekten.

Level 3 Processing - Impact Assessment:

„Estimation and prediction of effects on situations of planned or estimated/predicted actions by the participants"[15] [SB01].

Diese Ebene erweitert die Ebene 2 Beurteilung *(Level 2 Processing)* in Richtung einer Projektion der aktuellen Situation in die Zukunft, um Schlussfolgerungen über mögliche Handlungsoptionen und -änderungen ziehen zu können.

Level 4 Processing - Process Refinement:

„Adaptive data acquisition and processing to support mission objectives"[16] [SB01].

Dieser Metaprozess überwacht den gesamten *Data Fusion*-Prozess um ihn zu bewerten und die Echtzeitfähigkeit der laufenden *Data Fusion* zu verbessern gemäß der definierten *Measures of Performance* (MOP) und *Measures of Effectiveness* (MOE) [Lli01]. Dies umfasst Funktionen wie Sensorkoordination und -überwachung, Ressourcenoptimierung und die Bestimmung der Sensor-„Blickrichtungen" (*sensor cueing*).

Von unterstützender Funktion ist das Database Management System. Eine

[14] A. a. O., S. 6

[15] A. a. O., S. 6

[16] A. a. O., S. 6

Schlüsselkomponente eines *Data Fusion*-Prozesses ist die Bewältigung von sehr umfangreichen Datenmengen [Ant95]. Grob lässt sich hierbei noch zwischen den unterstützenden Datenbanken mit Daten, wie z.B., um im landwirtschaftlichen Kontext zu bleiben, unterschiedlichem Kartenmaterial zu Topographie, historischem Ertrag, Bodenbeschaffenheit, als auch der Fusionsdatenbank mit all den Zwischenprodukten und dem Endergebnis des *Data Fusion*-Prozesses unterscheiden.

Ein noch neuer Updatevorschlag für dieses „*revised JDL data fusion model*" stammt von HALL ET AL. (2001) [HHT01] und wurde von BLASCH UND PLANO (2002) [BP02] hinsichtlich „*group tracking*" diskutiert. Entsprechend deren Vorschlag bedarf das Modell der Erweiterung um einen „*Level 5 Processing - Cognitive refinement*", da sehr viele *Data Fusion*-Systeme einen „*human in the loop*"-Entscheidungsfinder mit involvieren. Die Funktionsweise dieser Bewertungsebene wäre ähnlich den *Level 4 Processing* - Funktionalitäten, da die Effizienz des Systems Sensor, Datenverarbeitung und Nutzer im Endeffekt davon beeinflusst wird, wie der Nutzer die vom Fusionssystem bereitgestellte Information wahrnimmt und demzufolge seine weiteren Eingabe- und Kontrollaktivitäten durchführt.

STEINBERG UND BOWMAN (2001) [SB01] weisen kritisch auf die Missinterpretationsmöglichkeit ihres Modells hin. Es wäre ein Fehler, das Modell als ein Prozessmodell zu verstehen und dabei anzunehmen der Informationsfluss im Fusionsprozess muss strikt vom Level 0 zum Level 1, dann Level 2 und anschließend Level 3 stattfinden. Dieser „*bottom up*" Fusionsprozess gilt nur, wenn Sensorbeobachtungen jeweils in einzelne Messergebnisse aufgeteilt werden können, die von einer echten Entität stammen. Und Zweitens die komplette Information, die für die Einschätzung eines Entitäts-Zustands nötig ist, in den Messungen von einer individuellen Entität enthalten ist.

Jedoch ist ja gerade eine Stärke der *Data Fusion*, dass sie den Aspekt der Kontextsensitivität mit ins Spiel bringt und zum Beispiel einer *Level 1 Processing* Funktion durch auf *Level 2 Processing* gewonnenes oder vorhandenes Kontextwissen ihre Funktion erleichtert bzw. erst ermöglicht.

2.1.4.2 Dasarathy's Functional Model

DASARATHY (1994) [Das94] definiert eine Kategorisierung von *Data Fusion*-Funktionen basierend auf dem Typus von Daten und Information, die verarbeitet werden, und den Informationsarten die als Resultat aus dem Fusionsprozess hervorgehen. Tabelle 2.3 illustriert Dasarathy's *Data Fusion* I/O Modell.

Die sich ergebenden *Data Fusion*-Funktionen kürzt Dasarathy mit den in der Tabelle aufgeführten Abkürzungen ab, z.B. DEI-FEO. STEINBERG UND BOWMAN (2001) [SB01] verweisen darauf, dass diese Kategorisierung einen intuitiven Weg aufzeigt, mit dem *Data Fusion*-Techniken bzw. -Algorithmen den einzelnen Feldern zugeordnet werden können. Da damit jedoch nur *JDL Level 0* und *1 Processing* abgedeckt werden, haben die beiden Autoren mit einer Erweiterung um die Kategorien *Relations*, *Impacts* und *Responses* auch noch die fehlenden Levels 2 bis 4 mit abgedeckt.

Tabelle 2.3: Dasarathy's *Data Fusion* I/O Modell (nach [SB01])

	Output		
Input	*Data*	*Features*	*Objects*
Data	*Signal Detection* (DAI-DAO)	*Feature Extraction* (DAI-FEO)	*Gestalt-Based Object Characterization* (DAI-DEO)
Features	*Model-Based Detection/ Feature Extraction* (FEI-DAO)	*Feature Refinement* (FEI-FEO)	*(Feature-Based) Object Characterization* (FEI-DEO)
Objects	*Model-Based Detection/ Estimation* (DEI-DAO)	*Model-Based Feature Extract* (DEI-FEO)	*Object Refinement* (DEI-DEO)

2.1.5 Prozessmodelle

2.1.5.1 Hall's taxonomy

Beginnend mit diesem Kapitel wird der Blick weg von den funktionalen Modellen hin zu den Modellen mit prozeduralen Charakter gerichtet. Basierend auf dem funktionalen *JDL data fusion model* haben HALL UND MCMULLEN (2004) [HM04] eine Taxonomie von Algorithmen, die die Funktionen der einzelnen Levels erfüllen können, abgeleitet und in dem Buch „*Mathematical Techniques in Multisensor Data Fusion*" veröffentlicht. Diese Sammlung an Algorithmen ist zu umfangreich, um hier wiedergegeben werden zu können. Sie unterscheiden grundsätzlich zwischen Algorithmen nach folgender Grobgliederung:

- *Level 1 Processing: Data Association and Correlation,*
- *Level 1 Fusion: Kinematic and Attribute Estimation,*
- *Identity Declaration,*
- *Decision-Level Identity Fusion,*
- *Knowledge-Based Approaches,*
- *Level 4 Processing: Process Monitoring und Optimization.*

Es stellt sich die Frage, ob eine reine, geordnete Algorithmensammlung bereits als ein Prozessmodell bezeichnet werden kann. Da ein Prozessmodell die Interaktion zwischen Funktionen innerhalb eines Systems spezifiziert und die grundsätzlichen Lösungsansätze abbildet, liefert Hall's Taxonomie zumindest eine dem JDL funktionalen Modell zugeordnete Übersicht über zur Verfügung stehende *Data Fusion*-Lösungsansätze. Damit ist ein prozeduraler Charakter gegeben. Teile der aufgeführten Algorithmen könnten anderseits auch als Kernstück eines rein formalen Modells herangezogen werden.

2.1.5.2 Boyd's Decision Loop

HALL UND MCMULLEN (2004) [HM04] führen als ein prozedurales Modell „Boyd's Decision Loop" auf. Ursprünglich wurde dieses Modell zur Formalisierung von Konzepten der militärischen Taktik und Entscheidungsfindung verwendet; es wird jedoch auch für die Ausbildung von Fahrschülern genutzt. Die Entscheidungsschleife setzt sich zusammen aus den Komponenten „*observe*", „*orient*", „*decide*" und „*act*". Verkürzt dargestellt finden folgende Prozesse statt.

- *Observe:* Es wird eine systematische und fortwährende Datensammlung ausgeführt, um Information über eine Situation abzuleiten.
- *Orient:* Eine Situationseinschätzung und -bewertung wird basierend auf den gesammelten Daten durchgeführt.
- *Decide:* Mit den vorliegenden beobachteten Daten, der abgeleiteten Information und der durchgeführten Situationsbewertung ist der dritte Schritt das Fällen einer Entscheidung. Solch eine Entscheidungsfindung wird durch Abwägung von alternativen Hypothesen und den darauf basierenden unterschiedlichen Konsequenzen herbeigeführt.
- *Act*: Schließlich soll der Entscheider die gefundene Entscheidung auch in die Tat umsetzten. Dies können eine militärische Handlung, die Sammlung weiter Daten, die Neukonfiguration von Informationsressourcen oder andere Aktivitäten sein.

2.1.5.3 Omnibus Process Model

BEDWORTH UND O'BRIEN (1999) [BO99] vergleichen verschiedene funktionale und prozedurale Modelle für MSDF und schlagen dann ihrerseits ein übergreifendes „Omnibus" - Prozessmodell vor, das auf dem Zyklus <OBSERVE - ORIENT - DECIDE - ACT > basiert. Es ist nicht erforderlich, dass dieser Prozess ein einzelner zyklischer Prozess ist, sondern es wird viel eher ein hierarchischer und rekursiver Prozess mit Analyse- bzw. Entscheidungsschleifen sein, die Entdeckung, Bewertung, Evaluierung und Antwortentscheidung auf verschiedensten Ebenen unterstützen. Dieser Prozess impliziert,

dass Ressourcenbelegung und Kontrollaktion, wie z.B. Sensormanagement, stattfinden.

2.1.5.4 Antony's Data Fusion Process Model

Der Vorschlag von ANTONY (1995) [Ant95] für ein Prozessmodell gründet auf der gleichen fundamentalen Idee, jedoch ist er konkreter und wendet eine biologisch motivierte Sichtweise an, d.h. er kopiert die herausragenden (Multisensor) *Data Fusion* Fähigkeiten von Menschen und Tieren hinsichtlich der Situationswahrnehmung. Dieses intuitive Prozessmodell des *Data Fusion*-Prozesses führt zu der Identifikation von 15 Klassen von Fusions-Problemen und einer Taxonomie von 16 kanonischen Problemlösungsformen (Index: I-XVI). Dabei existiert eine wohl definierte Beziehung zwischen dem 5-Ebenen funktionalem Fusions-Modell und den 15 Klassen von Fusions-Problemen, sowie eine definierte Beziehung zwischen diesen Klassen und den kanonischen Problemlösungsformen. Deswegen bietet dieses Prozessmodell einen geradlinigen Ansatz zur Auswahl eines geeigneten Algorithmus.

Data Fusion-Prozessmodell mit mehreren Abstraktionsebenen (*multiple levels of abstraction*) zur Unterstützung von kontext-sensitivem Schlussfolgern:

> Das Ziel von *Data Fusion* ist es, Daten in nützliche Information umzuwandeln. Das Produkt des (Multisensor) *Data Fusion*-Prozesses ist eine Situationsbeschreibung (*situation representation*), die durch einen Fusionsalgorithmus bzw. -technik erreicht werden soll. Die Situationsbeschreibung M (Abb. 2.3 a) ist das „zusammengesetzte" Produkt der „*Revised JDL Processing Level 0 – 3*". Das *Data Fusion*-Prozessmodell abgebildet in Abbildung 2.3 a. unterstützt die Kombination der Beobachtungen (*observables*) O (Sensorabgeleitete Messergebnisse bzw. Messungen), dem a priori Schlussfolgerungswissen K und der *multiple levels of abstraction* - Situationsbeschreibung M. Alle drei Klassen von Wissen sind potenziell sensitiv hinsichtlich impliziten Domänen-Wissen D. In einem zyklischen und oder ereignisgesteuertem Ablauf werden die Situationsbeschreibung M, das a priori Schlussfolgerungswissen K und das implizite Domänen Wissen D stets aktualisiert.
>
> Genauso wie kurz-, mittel- und langfristiges Gedächtnis (bzw. Erinnerung) der Dauerhaftigkeit von Information in biologischen Systemen zugeordnet werden kann, steht kurz-, mittel- und langfristiges Wissen mit der Dauerhaftigkeit von Information in künstlichen *Data Fusion*-Systemen in engem Zusammenhang. Wie in Abbildung 2.3 b dargestellt, können die Elemente des generellen *Data Fusion*-Prozessmodells, gezeigt in Abbildung 2.3 a, leicht mit diesen drei Klassen von Wissen assoziiert werden. Die

Sensor-Beobachtungen O repräsentieren kurzfristiges deklaratives Wissen, die Situationsbeschreibung M repräsentiert mittelfristiges deklaratives Wissen, K stellt statisches deklaratives Wissen dar, D kann sowohl statisches wie auch dynamisches nicht-sensor-abgeleitetes deklaratives Domänenwissen (Kontext) darstellen und F repräsentiert langfristiges prozedurales Schlussfolgerungswissen.

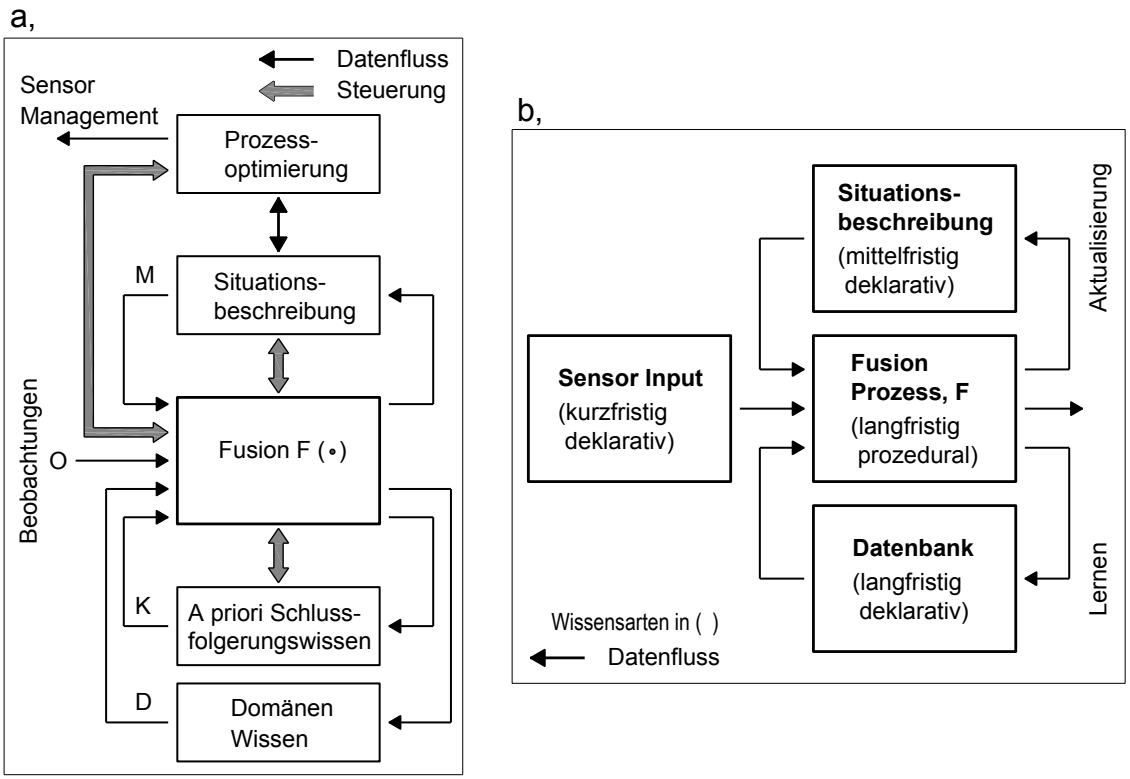


Abbildung 2.3 a, b: Prozessmodell (nach [Ant95])

Definition von 15 Fusions-Klassen:

Daher kann der *Data Fusion*-Prozess ebenso wie im biologischen Analogon als die generelle Komposition und Interaktion zwischen kurzfristigem deklarativen, mittelfristigem deklarativen, langfristigem deklarativen und langfristigem prozeduralen Wissen gesehen werden. Als Ergebnis dieser Charakterisierung können 15 eindeutige *Data Fusion*-Klassen identifiziert werden, die sich aus allen möglichen Kombinationen aus eins, zwei, drei oder vier Wissensklassen ergeben.

Definition einer Taxonomie von 16 kanonischen Problemlösungsformen:

Acht Algorithmusformen können unterschieden werden. Sie unterscheiden sich durch acht Klassen von prozeduralem Wissen, das basierend auf zwei Klassen von langfristigem deklarativem Wissen (spezifisch, generell) und vier Klassen von Kontroll-

Wissen (rigide, flexibel, einfache Abstraktionsebene, mehrfache Abstraktionsebenen) definiert ist. Ausgehend von diesen acht Algorithmusformen und den beiden grundsätzlichen Basis-Problemlösungs-Paradigmas vom Typ generierungsbasiert oder hypothesenbasiert können insgesamt 16 kanonische Problemlösungsformen definiert werden.

Bei der Ableitung einer geeigneten Problemlösungsform wird folgendermaßen vorgegangen. Durch die Identifizierung der Klassen von relevantem Wissen, das zur Anwendung kommen soll, wird eine geeignete Fusions-Klasse bestimmt. Dann werden ausgehend von der Beziehung zwischen den 15 Fusions-Klassen und den 16 kanonischen Problemlösungsformen eine oder alternativ mögliche Problemlösungsformen bestimmt. Die letztendliche Festlegung auf eine geeignete kanonische Problemlösungsform wird vor allem anhand der erforderlichen Algorithmus-Robustheit und Kontext-Sensitivität durchgeführt.

2.1.6 Systemarchitektur

Die „Übersetzung“ des funktionalen Modells und des Prozessmodells in eine physikalische (Hardware und Software) Architektur, eine Systemarchitektur, erfordert die Zuordnung zahlreicher Anforderungen in Funktionen und Subfunktionen innerhalb eines *Data Fusion*-Systems [WL90].

HALL UND LLINAS (2001) [HL01] unterscheiden drei grundsätzliche Alternativen für MSDF und zwei Typen von Datenquellen für die Eingangsgrößen:

- Direkte Fusion der Sensordaten,
- Repräsentation der Sensordaten mittels eines Merkmals-Vektor (*feature vector*) mit nachfolgender Fusion von Merkmals-Vektoren,
- Verrechnung bzw. Verarbeitung in jedem einzelnem Sensor, um Ableitungen und Entscheidungen auf höherer Ebene treffen zu können, die anschließend kombiniert werden.

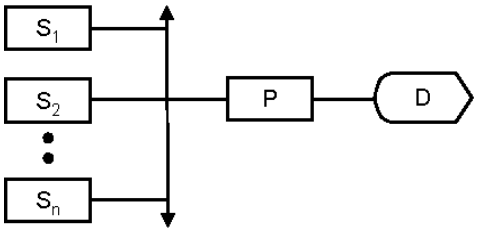
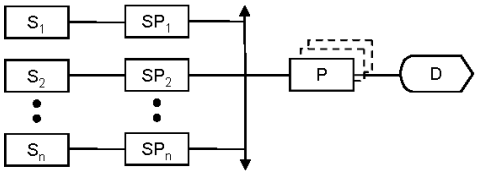
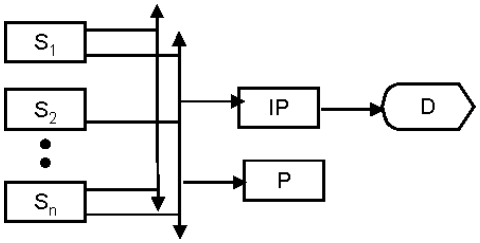
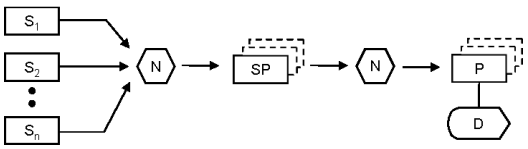
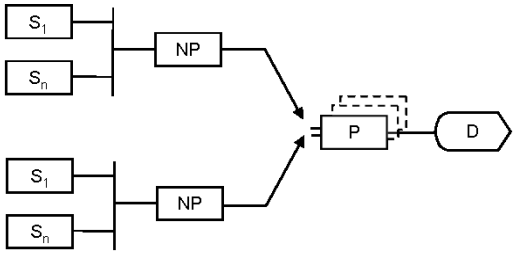
Es existiert ein ausgeprägter Unterschied zwischen gleichartigen (*commensurate*), d.h. wenn Sensoren das gleiche physikalische Phänomena messen, und ungleichartigen (*noncommensurate*) Multisensor Daten. Dabei gilt, falls (Multisensor) Daten gleichartig sind, dann können die Rohdaten direkt kombiniert werden. Im Gegensatz dazu, falls die Sensordaten ungleichartig sind, dann sind die Daten auf Merkmals- oder Zustands-Vektor-Ebene oder Entscheidungsebene zu fusionieren.

Die Abbildung der Funktionen und Prozesse, abgeleitet aus dem funktionalem und dem

Prozessmodell, in eine Systemarchitektur erfordert deren Zuordnung zu physikalischen Einheiten, die über Kommunikationsschnittstellen verbunden sind. WALTZ UND LLINAS (1990) [WL90] führen dazu folgende Top-Level Systemarchitekturen auf, wie sie in Tabelle 2.4 zusammengefasst sind. Dabei werden die *JDL Level 1-3 Data Fusion*-Prozesse unter Beachtung nachfolgend aufgelisteter Systemcharakteristika unterschiedlichen Prozessoren zugeordnet:

- Systemdurchsatz, z.B. Latenzzeit, Nebenläufigkeit, erreichbare Zykluszeiten,
- zentralisierte gegenüber verteilter Entdeckung und Abschätzung,
- zentralisierte gegenüber verteilter Datenbankspeicherung und –zugriff,
- Einschränkungen des Nachrichtenverkehrs, z.B. *Routing*, Übertragungskapazität, Verzögerungszeiten.

Tabelle 2.4: Repräsentative Systemarchitekturen (nach [WL90])

Typ	Beschreibung	Schema
Central Fusion Processing	Sensoren (S) stellen *report-level* Daten einem zentralen Prozessor (P) zur Verfügung. Alle *levels of fusion* werden von einem einzigen Prozessor (P) ausgeführt, der die Daten auf ein Display (D) ausgibt.	
Distributed (Federated) Level 1 Processing	*Sensor-level tracking* und Vorverarbeitung wird in Sensorprozessoren (SP) durchgeführt. Alle *levels of fusion* werden von einem einzigen (/mehreren) Prozessor (/en) (P) ausgeführt.	
Partitioned Level 1 Processing	*Imaging* Sensoren (S) führen Segmentierungs- und *feature extraction* im Sensor durch. Bilddaten und *Feature*-Daten werden dem Fusionsprozess auf unterschiedlichen Pfaden zugeführt. Ein Bildverarbeitungsprozessor (IP) registriert die Bilddaten und ein Fusionsprozessor (P) klassifiziert die *Feature*-Daten.	
Fully integrated	*Switch* Netzwerke (N) erlauben den Sensorprozessoren (SP) zwischen allen Sensoren (S) hin und her geschalten zu werden, um die Rechenlast zu teilen und Redundanz bereitzustellen. Mehrere Prozessoren (P) führen alle *levels of fusion* durch. Die *level* können unter den Prozessoren aufgeteilt werden.	
Distributed Sensor/Fusion	Sensorknoten ($S_1 \ldots S_n$) sind physikalisch verteilt, dabei hat jeder Sensorknoten einen lokalen, zentralen Prozessor (NP), der die lokale *Data Fusion* (*JDL Level 1*) durchführt. Jeder Knotenprozessor (NP) leitet seine Informationen an einen oder mehrere globale Prozessoren (P) weiter, die *JDL Level 1 Association* und *Level 2,3* Datenverarbeitung durchführen.	

2.2 MSDF Ansätze und Implementierungen in der mobilen Agrarsystemtechnik

2.2.1 Ortung und Navigation

Zur Erhöhung der Verfügbarkeit und Verbesserung der Genauigkeit der Positionsbestimmung als Eingangsgröße für die Georeferenzierung oder die Navigation von landwirtschaftlichen Fahrzeugen (Traktoren und Selbstfahrern) wie auch von Geräten, wird auf MSDF-Implementierungen zurückgegriffen. Ein großer Teil der veröffentlichen Fusionstechnik wendet die Kalman-Filter-Theorie an, die Eignung für DGPS-Parallel-Fahrsysteme haben beispielsweise HAN ET AL. (2002) [HZN02] gezeigt.

FREIMANN (2003) [Fre03] definiert nach Literaturrecherche folgende Sensorsysteme, die in der Agrarsystemtechnik zur Erzielung höherer Genauigkeiten in der Positions-, Fahrwegs- und Fahrzustandsbestimmung gekoppelt werden:

- Das globale Positionssystem (GPS) teilweise mit differentieller Korrektur (DGPS, PDGPS),
- Digitales Kartenmaterial (z.B. Sollfahrspur, Feld bzw. virtuelle Grenzlinien),
- Trägheitssensoren (Längs- und Querbeschleunigung, Wanken, Nicken),
- Drehratensensoren/Kreiselkompass (Gieren, Fahrzeugausrichtung),
- Radar, Ultraschall (z.B. Geschwindigkeit, Abstandserkennung),
- Laserscanner (Abstandserkennung, Kantendetektion),
- Kamerasysteme (z.B. Kantendetektion, Schwaderkennung).

2.2.2 Umfelderfassung

Gerade die Integration und Fusion der Information der Sensorklassen wie Radar, Ultraschall, Laserscanner und Kamerasysteme gewinnen an Bedeutung, wenn Umfelderfassung bzw. Hinderniserkennung gefordert ist. BLACKMORE UND GRIEPENTROG (2006) [BG06] stellten die Anforderung an Sensorsysteme zur Umfelderfassung zusammen, die für die Realisierung autonomer landwirtschaftlicher Maschinen nötig sind. Hervorgehoben haben sie einen Ansatz mit 17 Ultraschallsensoren, die rund um das Vehikel angeordnet sind, um somit eine Rundumsicht-Hinderniserkennung um das Fahrzeug zu ermöglichen.

Umfelderfassung zur Gewährleistung von Sicherheit ist jedoch nur ein Aspekt. Sie ist auch gefordert, um einzelne Prozessschritte automatisieren zu können. So beschrieben KONDO ET AL. (1998) [KFMS98] einen wesentlich komplexeren MSDF-Ansatz für einen Pflückroboter

zum Ernten von Früchten wie Äpfeln oder Orangen. Dabei wird die Frucht mittels eines Systems zum Maschinellen Sehen detektiert. Dies erfolgt in einer 2D-Projektion auf einer Frucht-Baum-Oberflächenhüllfläche (*„canopy“*). Näherungssensoren und „Tastfühler“ am Greifarm ergeben eine 3D-Wahrnehmung und ermöglichen das zerstörungsfreie Greifen der Frucht.

2.2.3 Prozesssteuerung für Applikationssysteme

Auch im Bereich der Applikationssysteme sind erste Untersuchungen zur Anwendbarkeit von Multisensor Daten Fusion bekannt. BILLER (2003) [Bil03] entwickelte im Rahmen des Projekts „*Advanced Optoelectronic System* (AOS)“ ein Multisensor-System, welches mit optoelektronischen Sensoren online die Reflexion des Umgebungslichtes durch Nutzpflanzen und Unkräuter in ausgewählten Wellenlängen erfasst (spektraler Fingerabdruck) und anhand typischer Merkmale die Unterscheidung ermöglicht sowie mit weiteren Sensoren und Algorithmen sicherstellt, dass unabhängig von äußeren Einflüssen (Windeinfluss, Spritzgestängeschwankungen) nur dort online appliziert wird, wo auch etwas detektiert wurde. RAMON ET AL. (2006) [RMBVD06] verglichen unterschiedliche Sensortechniken zur Identifizierung von Pflanzenkrankheiten und Stresssymptomen im Hinblick auf Tauglichkeit für teilflächenspezifischen Pflanzenschutz. Da jede Sensorik ihre eigenen technischen Probleme und Schwierigkeiten bei der Interpretation der Messergebnisse besitzt, stellen sie die Frage in den Raum, ob ein MSDF-Ansatz nicht der erfolgversprechendere Weg zur Realisierung der Prozesssteuerung ist. Einen konkreten Implementierungsvorschlag bleiben sie aber schuldig. Ebenfalls im Bereich der Pflanzenschutzapplikation wählten OOMS ET AL. (2002) [OLRD02] einen Sensorfusions-Ansatz zur Bestimmung der horizontalen Spritzgestängebewegungen. Dabei war das Ziel mittels *Data Fusion* Gier- und Wankbewegungen sowie dynamische Gestängeverformungen unterscheiden zu können. Hierzu wurde das Fahrzeug mit einem Radarsensor (Geschwindigkeit) und einem dreiachsigen Kreiselsystem ausgestattet, während die Gestänge mit Ultraschall- und Beschleunigungssensoren ausgerüstet waren. Ebenfalls für den Bereich der selektiven Pflanzenschutzapplikation entwickelten KLOSE ET AL. (2008) [KTRM08] einen autonomen Feldroboter, der ein *Sensor Fusion*-System, bestehend aus einer „Intelligenten Kamera“ (*smart camera*), einem Ultraschallsensor und IR-Distanzmessungssensoren unterschiedlicher Reichweite, sowohl zur Navigation als auch zur Beikraut-Erkennung nutzt. ADAMCHUK ET AL. (2011) [ARSS11] geben eine aktuelle Aufstellung über Sensorfusion-Ansätze für *Precision Agriculture*. Die letztendliche Motivation zielt dabei auf Applikationssysteme ab.

Bei der Aufzählung der wenigen Implementierungsbeispiele aus dem wissenschaftlichen Bereich stehen Sensorfusion-Ansätze zur Erfassung bzw. Ermittlung von Bodeneigenschaften im Vordergrund. Im Bereich der Erfassung von Pflanzen- und Bestandeseigenschaften wird nur ein Sensorfusion-System zur gleichzeitigen optischen Reflexions-, Bestandeshöhe und Temperaturmessung angeführt. Für eine verbesserte Prozessführung für Applikationssysteme fordern die Autoren zur Ermittlung der teilflächenspezifischen Applikationsrate, die Daten über Bodeneigenschaften und Pflanzeneigenschaften zu verknüpfen. Der *Data Fusion*-Vorschlag ist ein Flussdiagramm, welches Karten und (Sensor-)Messergebnisse zu neuen Karten aggregiert und miteinander verschneiden soll, um letztendlich zu „*Management Zone Maps*“, „*Interpolated Thematic Maps*“ und „*Temporal Data*“ zu kommen. Daraus ließen sich in einem weiteren Schritt die Applikationsraten ableiten. Diese Darstellung entspricht im weitesten Sinne einem funktionalen Modell für *Data Fusion*. Auch wenn die aufgeführten Sensorfusion-Beispiele noch spärlich sind, gehen die Autoren davon aus, dass ihre Anzahl in Zukunft (stark) steigen wird[17].

2.2.4 Zustandsüberwachung und Teleservice

Landwirtschaftliche Maschinen, Traktoren und mobile Arbeitsmaschinen sind aufgrund der rauen und wechselnden Einsatzbedingungen besonders wartungsintensiv und Stillstandszeiten sind aufgrund der knapp bemessenen „Einsatzfenster“ kaum tragbar bzw. äußerst kostspielig. Daher nimmt das *Condition Monitoring*, die Zustandsüberwachung, meist in Verbindung mit dem Teleservice eine entscheidende Rolle ein. Für die Zustandsbestimmung von einzelnen Komponenten oder Maschinenfunktionen reicht ein Sensor alleine meist nicht aus, weshalb erst der Einsatz von MSDF-Techniken zum Ziel führt. Hierfür hat KRALLMANN (2005) [Kra05] die Rahmenbedingungen für die Nutzung multisensorieller Daten in einem *Condition Monitoring*-System von mobilen Arbeitsmaschinen analysiert. Zur Umsetzung hat er als konkretes Beispiel einen Multisensor, der verschiedene charakteristische Kenngrößen von Hydraulikflüssigkeiten ermittelt, verwendet und eine Zustandsbestimmung der Hydraulikflüssigkeiten realisiert.

FÖLSTER (2006) [Föl06] hat bei seiner Arbeit zur Expertensystemtechnik für Teleservice bei Landmaschinen die prinzipiellen technischen Möglichkeiten eines solchen Systems untersucht, in dem aus der Ferne kontakt- und reaktionsfähige Komponenten diagnostiziert werden können. Weiterer Untersuchungsgegenstand war die Fragestellung, inwieweit

[17] [ARSS11], a. a. O, S. 37: *„... New research in the area of sensor fusion for precision agriculture is expected to provide a variety of such examples.“*

Aussagen über nicht direkt diagnosefähige Komponenten und Baugruppen oder über den Gesamtzustand der Maschine möglich sind und/oder aus den gemessenen Größen abgeleitet werden können. Dazu wird einerseits auf direkt während des Arbeitsprozesses gemessene Daten zurückgegriffen, anderseits auf zentral hinterlegte Prozess- und Maschinendaten, die von anderen Maschinen über längere Zeiträume gesammelt werden.

OSTERMEIER ET AL. (2007) [OGAHG07] haben ein „Intelligentes Kugellager“ hinsichtlich der Eignung für den Einsatz auf Landmaschinen untersucht. Dieses Kugellager war durch Einsatz eines Multisensors, der aus mehreren über den gesamten Kugellageraußenring gleichmäßig verteilten DMS-Sensoren und einem Temperatursensor bestand, in der Lage, die axiale und radiale Last, die Drehzahlen, die Unwucht, den Lastwinkel und die Temperatur zu bestimmen.

BLANK ET AL. (2011) [BKB11] betrachteten nicht nur ein Multisensor-System oder eine Gesamtmaschine sondern richteten den Blick auf dynamisch sich ändernde Traktor-Geräte-Kombinationen oder Flottenlösungen von Landmaschinen. Sie stellten eine Lösung vor, wie dafür eine Zustandsbestimmung (*vehicle status determination*) gelingen kann. Es wurde ein modularer Sensorfusion-Ansatz vorgeschlagen, der die dynamische Rekonfiguration unterstützt. Der Vorschlag umfasst ein Mehrebenen-Sensorfusion-System mit „Intelligenten Sensoren“ und Fusionsalgorithmen für die Bestimmung der Fahrgeschwindigkeit und des Arbeitszustandes gemäß dem JDL Level 1 und 2 Processing.

2.3 Real-time Prozessführung für sensorgestützte Düngesysteme im Sensor-Ansatz mit Kartenüberlagerung

Ausgehend von der Einordnung einer Real-time Prozessführung für sensorgestützte Düngesysteme in den Komplex der „Präzisen Landwirtschaft“ werden unterschiedliche Systemansätze zur kleinräumigen Bestandesführung vorgestellt. Nach einer systemtheoretischen Einordnung der Systemansätze werden die benötigten Sensoren, Kartierungstechniken und Prozesssteuerungselemente, d.h. die Schlüsselelemente und Techniken des „Präzisen Ackerbaus“ und der in der Agrarsystemtechnik angewandten Steuerungs- und Automatisierungstechnik in den weiteren Unterkapiteln näher dargestellt.

2.3.1 Methodische Ansätze zur kleinräumigen Bestandesführung - Düngung

Intelligente Landtechnik als Garant für „Präzision“ und damit für „Information“ führt laut AUERNHAMMER (2002) [Aue02] zur Präzisionslandwirtschaft (*Precision Agriculture*), in der neben der „Präzisen Tierhaltung“ der „Präzise Ackerbau“ (*Precision Farming*) eine zentrale

Rolle spielt (Abb. 2.4). Der Präzise Ackerbau wiederum lässt sich in Dokumentation, Flottenmanagement, Feldrobotik und Teilfächenbewirtschaftung (*site specific crop management*) differenzieren. Der Präzise Ackerbau verfolgt das Ziel, mit Hilfe modernster Technik das acker- und pflanzenbauliche sowie arbeits- und betriebswirtschaftliche Management weiter zu optimieren, besonders im Hinblick auf eine optimale Nutzung der eingesetzten Ressourcen, bei gleichzeitiger Sicherung der Nachhaltigkeit der landwirtschaftlichen Produktion [DD06]. Hierbei spielt besonders die Forderung der guten fachlichen Praxis nach standortangepassten Maßnahmen mit minimalen Stoffeinträgen in benachbarte Systeme eine wichtige Rolle.

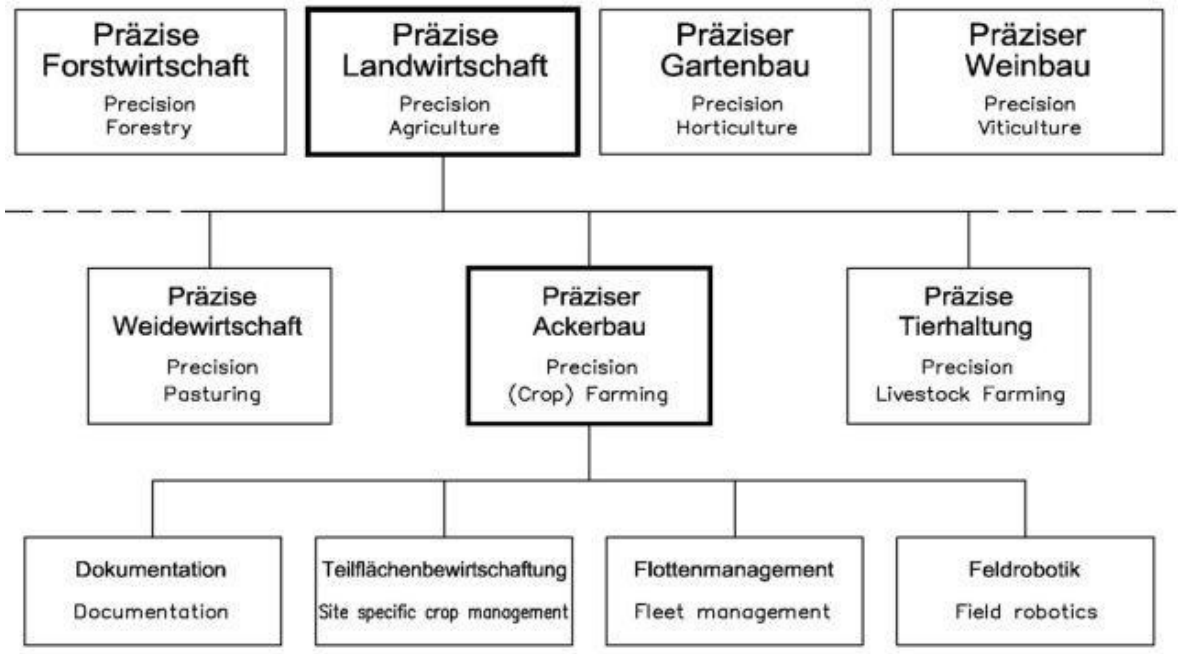


Abbildung 2.4: Übersicht - Präzise Landwirtschaft [Aue02]

Mittels dieser Teilflächenbewirtschaftung bzw. teilflächenspezifischen Bewirtschaftung kann die Heterogenität von Boden- und Pflanzenzuständen eines Schlages adressiert werden. Ist eine teilflächenspezifische Maschinensteuerung verfügbar, können lokal variierte Behandlungsmaßnahmen, wie Bodenbearbeitung, Saat, Düngung und Pflanzenschutz durchgeführt werden. AUERNHAMMER UND SCHUELLER (1999) [AS99] unterschieden drei unterschiedliche Systemansätze, die die Prozessführung in mobilen Applikationssystemen für teilflächenspezifische Applikation bestimmen. Diese sind der Kartierungsansatz (*mapping approach*), der Sensor-Ansatz (*sensor approach*) oder die Kombination der beiden Ansätze, der Sensor-Ansatz mit Kartenüberlagerung (r*eal-time approach with map overlay*) (Abb. 2.5).

Im Folgenden wird nur die Stickstoffdüngung betrachtet.

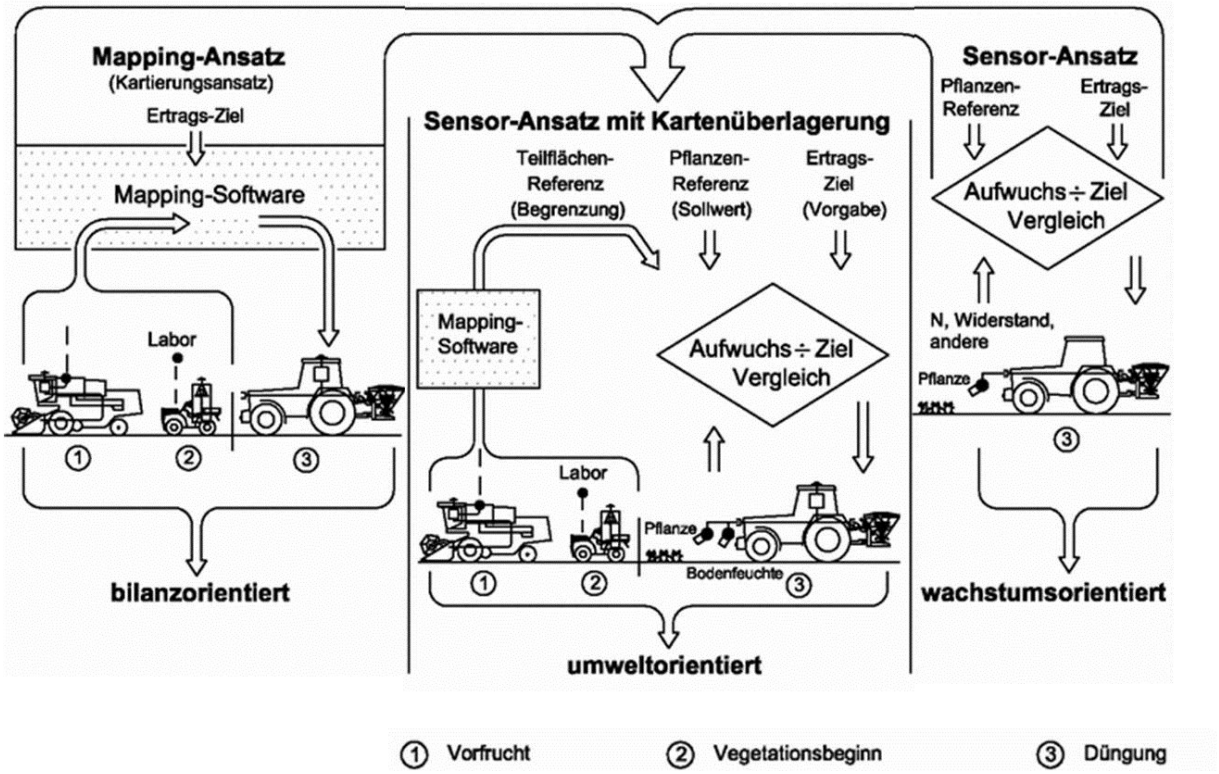


Abbildung 2.5: Systemansätze für die Teilfächenbewirtschaftung [Aue01a]

Kartierungsansatz: Beim diesem Ansatz werden die erforderlichen Düngungsmaßnahmen basierend auf historischen Informationen zur Variabilität der Böden und der Pflanzen abgeleitet. Gekennzeichnet ist dieser Ansatz durch die zeitlich und räumlich abgesetzte Informationsbeschaffung sowie -weiterverarbeitung und schließlich der eigentlichen Applikation. Für die in dieser Arbeit maßgebliche Stickstoffdüngung bedeutet dies, dass in der Regel aus mehrjährigen Ertragskarten und/oder Bodenkarten am stationären Betriebsrechner Managementzonen abgeleitet werden. Innerhalb der einzelnen Zonen wird eine einheitliche Düngung angewendet und ein entsprechender Applikationswert unter Einbeziehung aller vorliegenden Informationen abgeleitet. Dies kann z.B. nach Simulationsmodellen der Stickstoffdynamik [Eng91] oder nach der Methode der N-Bilanzierung und der damit verbundenen Düngung nach Entzug [Auf98] erfolgen. Die Grundidee dabei ist, dem Boden die Menge an Nährstoffen über Düngung wieder zuzuführen, die mit dem Erntegut entzogen worden ist. Dies bedeutet, dass an Hochertragsstandorten eine höhere Stickstoffmenge appliziert wird als in der Mittelertragszone und an Niedrigertragsstandorten wiederum eine geringere Menge. In der Abbildung 2.5 wird dies durch das Schlagwort „bilanzorientiert“ widergespiegelt.

Die am Betriebsrechner vorbereitete Applikationskarte wird auf das Prozesssteuerungssystem

der mobilen Applikationstechnik übertragen und kann dort zur Ausführung kommen. Unabdingbar ist jedoch die Verfügbarkeit eines Ortungssystems, um die jeweilige Teilfläche lokalisieren und die vorab berechnete Düngermenge ausbringen zu können. Aktuelle Pflanzen- oder Bodenzustände werde dabei systembedingt zum Zeitpunkt der Düngergabe jedoch nicht mehr erfasst.

Sensor-Ansatz: Beim Sensor-Ansatz, in der Literatur auch als Real-time-Ansatz bezeichnet, ist die Traktor-Gerätekombination oder der Selbstfahrer zur Düngerapplikation mit einem Sensorsystem ausgestattet, das Messwerte liefert, auf deren Basis die Düngermenge direkt bestimmt und dann unmittelbar mittels der Applikationstechnik (Düngerstreuer) ausgebracht wird. In der Regel wird mit geeigneter Sensorik das lokal vorliegende Pflanzenwachstum erfasst, mit einem standardisierten Wachstumswert verglichen, mit einer Ertragszielerwartung abgeglichen und dann der Düngersollwert abgeleitet. In der Abbildung 2.5 wird die Ausrichtung daher mit dem Schlagwort „wachstumsorientiert“ gekennzeichnet.
Mit der Entwicklung eines Real-time-Sensors, in der Literatur auch Online-Sensor bezeichnet, zur Erfassung des Stickstoffstatus der Pflanzen und der Kombination dieser Information mit der Düngerapplikation ist erstmals eine praxistaugliche Umsetzung dieses Ansatzes gelungen [WRK99].
Der reine Sensor-Ansatz ist ein autarkes System, es benötigt keine Boden- und Ertragsdaten, keine Zuordnung zu einer Managementzone und Applikationskarte und somit auch kein Ortungssystem.

Sensor-Ansatz mit Kartenüberlagerung: Systembedingt haben der Kartierungsansatz und der Sensor-Ansatz Nachteile, die jedoch durch die Kombination der beiden bereits aufgeführten Ansätze, dem Sensor-Ansatz mit Kartenüberlagerung, vermieden werden können [Aue01a].
Die Vor- und Nachteile der beiden Systemansätze sind genau gegensätzlich. Der Kartierungsansatz berücksichtigt aktuelle Veränderungen des Pflanzen- und Bodenzustandes der vorliegenden Teilfläche nicht. Zum anderen spielen gerade die Ertragskarten zur Ableitung der Managementzonen eine wesentliche Rolle. Hierbei haben sich neben all den Ungenauigkeiten der Ertragsmessung und -kartierung (siehe Kap. 2.3.2.3) auch eine Problematik der zeitlichen Instabilität von Ertragskarten gezeigt [BGF03].
Mit dem Sensor-Ansatz werden hingegen der Standorteinfluss (Ertragspotential) und N-Nachlieferungsvermögen des Bodens unberücksichtigt gelassen [Lie03]; eine sich daraus

ergebende Überdüngung in Niedrigertragsbereichen ist unvermeidbar [MSH04]. Damit einhergehend ist die Auswaschungsgefährdung von Nitratstickstoff sehr hoch. Mit der Kombination des reinen Sensor-Ansatzes mit den Heterogenitätsinformationen (z.B. Bodenkarten, Ertragskarten) entsteht ein umfassenderer Ansatz, der mögliche negative Auswirkung der Stickstoffdüngung auf die Umwelt verringern bzw. ausschließen kann. Dieser Charakter wird in der Abbildung 2.5 mit „umweltorientiert" bezeichnet.

Eine erste einfache praktische Umsetzung des Sensor-Ansatzes mit Kartenüberlagerung beschrieben LUDOWICY ET AL. (2002) [LSL02] mit der „Hinterlegung von Faktorkarten" und Integration eines Ortungssystems (GPS) in den Sensor-Ansatz basierend auf dem „HydroN-Sensor"[18]. In breiter wissenschaftlicher Tiefe wurde der Sensor-Ansatz mit Kartenüberlagerung für intensive Stickstoffdüngung im Rahmen des sechsjährigen (1999-2005) von der DFG geförderten Verbundprojektes IKB-Dürnast („Informationssystem Kleinräumige Bestandesführung Dürnast") [ADMSSW99], [AOMSSW06] ausführlich untersucht und implementiert.

Letztendlich muss dieser Düngesystemansatz auch im Hinblick auf die Forderungen von Ökologie und Ökonomie bestehen können. Einen umfassenden Überblick über Studien, in denen „Präziser Ackerbau"-Ansätze mit einheitlichen Ansätzen verglichen werden, gibt beispielsweise [Gan05]. WEIGERT (2006) [Wei06] kommt zu dem Schluss, dass sich bei den Studien kein eindeutiges Bild über die ökonomische Vorteilhaftigkeit von teilflächenspezifischen Düngeansätzen ergibt. Jedoch verweist er auch darauf, dass dies nur eine Momentaufnahme ist und noch keine Rückschlüsse auf das zugrunde liegende ökonomische Potential einer teilflächenspezifischen Düngung gezogen werden kann. Da nach Einschätzung von WEIGERT (2006) [Wei06] sowohl der Sensor- wie auch der Kartierungsansatz nicht zur ökonomischen Optimierung der Stickstoff-Ausbringung entwickelt wurden, schlägt er vor, dass für den Sensor-Ansatz mit Kartenüberlagerung ein neues Konzept mit Ausrichtung auf ein konkretes Zielsystem (z.B. Stickstoffkostenfreie Leistung) notwendig ist.

2.3.2 Prozessführung in der Agrarsystemtechnik

Die Umsetzung theoretischer Systemansätze des Präzisen Ackerbaus und hierbei im Speziellen der teilflächenspezifischen Applikation wird erst durch den Einsatz von Elementen und Methoden der Agrarsystemtechnik (bisher Landtechnik oder auch Agrartechnik genannt

[18] Dieser Online-Sensor wird mittlerweile unter der Bezeichnung „Yara N-Sensor" der Fa. Yara International ASA, Oslo, Norwegen am Markt angeboten.

(*Agricultural Engineering*)) ermöglicht. Grundsätzlich ist die Realisierung einer Real-time Prozessführung in einem sensorgestützten Düngesystem eine Aufgabenstellung der Automatisierungstechnik bzw. der Steuerungs- und Regelungstechnik. Aufgrund der besonderen Anforderungen und Einsatzbedingungen im Bereich des Ackerbaus kommen für diesen Industrie- und Wissenschaftsbereich angepasste Sensoren, Aktoren und informationsverarbeitende Techniken zur Anwendung. Ausgehend von den Grundlagen der Steuerungs- und Regelungstechnik werden die Schlüsseltechniken bzw. Grundelemente für Präzisen Ackerbau vorgestellt. Der Blick ist dabei auf die teilflächenspezifische Düngerapplikation ausgerichtet.

2.3.2.1 Prozessleittechnik

Die Bedeutung des Begriffs Prozess im täglichen Sprachgebrauch ist sehr vielschichtig, beispielsweise Wachstumsprozess, Denkprozess, Gerichtsprozess, Rechenprozess oder auch technischer Prozess. Im Folgenden soll der technische Prozess im Mittelpunkt stehen. Ein Prozess ist die Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Materie, Energie oder auch Informationen umgeformt, transportiert oder auch gespeichert werden [DINIEC60050-351]. Mit Prozessleittechnik wird das gezielte Einwirken auf den Ablauf eines Prozesses bezeichnet. Unter Leiten wird die Gesamtheit aller Maßnahmen verstanden, die einen im Sinne festgelegter Ziele erwünschten Ablauf eines Prozesses bewirken. Obwohl die Prozessführung im Prinzip als die Bedienung und Beobachtung auf Ebene der Leitwarte definiert ist, wird sie im täglichen Sprachgebrauch und in der Literatur oft für die Prozessleittechnik und das Leiten nach obiger Definition gebraucht. Im diesem Sinne wird Prozessführung auch im weiteren Verlauf der Arbeit verwendet.

Technische Prozesse können einfach oder auch sehr komplex sein, in der Regel werden in einem Prozessleitsystem sehr viele unterschiedliche Teilprozesse verwendet und zu einem übergeordneten Gesamtprozess zusammengefügt. Zur Prozessführung müssen die Zustandsgrößen technischer Prozesse mit technischen Mitteln gemessen und interpretiert werden können, sowie auch Zugriffsmöglichkeiten existieren, über die die Prozesse gezielt gesteuert und/oder geregelt werden können [RL94]. Die Regelungs- und Steuerungstechnik unterscheidet zwei grundsätzliche Methoden, einerseits die Prozesssteuerung und anderseits die Prozessregelung.

2.3.2.1.1 Steuerung und Regelung

Das Steuern – die Steuerung – ist der Vorgang in einem System bzw. Prozess, bei dem eine oder mehrere Größen als Eingangsgrößen andere Größen als Ausgangsgrößen auf Grund der dem System bzw. Prozess eigentümlichen Gesetzmäßigkeiten beeinflussen. Kennzeichen für das Steuern ist der offene Wirkungsablauf über das einzelne Übertragungsglied oder die Steuerkette. Es gilt zu beachten, dass die Benennung Steuerung vielfach nicht nur für den Vorgang des Steuerns, sondern auch für die Gesamtanlage verwendet wird, in der die Steuerung stattfindet [BDCD99].

Das Regeln – die Regelung – ist ein Vorgang, bei dem eine Größe, die zu regelnde Größe (Regelgröße x) fortlaufend erfasst, mit einer anderen Größe, der Führungsgröße w verglichen und abhängig vom Ergebnis des Vergleichs im Sinne einer Angleichung an die Führungsgröße beeinflusst wird. Der sich dabei ergebende Wirkungsablauf findet in einem geschlossenen Kreis, dem Regelkreis statt.

Die Regelung hat die Aufgabe, trotz störender Einflüsse den Wert der Regelgröße an den durch Führungsgröße vorgegebenen Wert anzugleichen, auch wenn diese Angleichung im Rahmen gegebener Möglichkeiten nur unvollständig geschieht.

Laut der Definition der Prozessleittechnik sind typische Aufgaben einer Prozessführung das Schützen, das Steuern, das Regeln und das Optimieren sowie die Prozessdokumentation. Dabei liegt die Güte der Prozessführung darin, wie genau und wie verzögerungsfrei sie die aufgelisteten Aufgaben zu erfüllen in der Lage ist.

Gemäß ARTMANN ET AL. (1993) [APS93] gliedern sich die Komponenten für Prozessleitsysteme je nach Prozessnähe in Elemente für die

- Erfassungsebene (mit den Funktionen Messen und Stellen),
- prozessnahe Steuerebene (mit den Funktionen automatisches Steuern und Regeln),
- prozessferne Leitebene (mit den Funktionen Dialog mit dem Menschen).

Hierbei beginnt und endet der Datenfluss der Leittechnik stets direkt an der Materie und Energie mit der Erfassung der Prozessdaten durch Sensorik und der Beeinflussung des Prozesses durch Stellglieder, den Aktoren. Die Sensorik im Allgemeinen wurde bereits in Kapitel 2.1.3 erörtert und auf die Sensorik, die speziell im weiteren Verlauf der Arbeit und im Rahmen der Teilschlagtechnik von Bedeutung ist, wird in Kapitel 2.3.2.2 näher eingegangen. Die nötige Aktorik zur teilfächenspezifischen Applikationstechnik für die Düngung ist Thema des Kapitels 2.3.2.4.

Auf der Steuerebene sind die prozessnahen Funktionen wie Überwachen, Steuern und Regeln angesiedelt, die einen automatischen Betrieb auch ohne andauernden menschlichen Eingriff

gewährleisten müssen. Auf den Stand der Technik in der Agrarsystemtechnik wird in den Kapiteln zu Prozessführungsstrategien, Systemarchitektur und Landtechnischen BUS-Systemen eingegangen. Auf der Leitebene werden die Funktionen wie Anzeigen, Bedienen, Dokumentieren und die gezielte Beeinflussung durch den Bediener realisiert.

Echtzeitverhalten ist eines der wesentlichen Merkmale eines Prozessführungssystems. Laut FÄRBER (1992) [Fär92] wird unter Schritt haltender Verarbeitung (Real-time-Betrieb, Echtzeit-Betrieb) die Betriebsweise einer Rechenanlage verstanden, welche die durch den Prozess gestellten Aufgaben zeitlich Schritt haltend verarbeiten kann. Ein bestimmter Rechenprozess muss demnach innerhalb einer maximal zulässigen Zeit ausgeführt werden können. Diese Zeitschranke gibt in der Regel der technische Prozess vor und kann sich z.B. bei Motorsteuerungen im Bereich von einigen µs bewegen. Charakteristisch für das Zeitverhalten eines technischen Prozesses sind die Ankunftszeitpunkte für Ereignisse aus diesem technischen Prozess und die maximal zulässigen Reaktionszeiten. FÄRBER (1992) [Fär92] weist darauf hin, dass es sich bei der maximal zulässigen Reaktionszeit meist nicht um eine naturgegebene Zeitkonstante, sondern vielmehr ein Maß für die Güte der Prozesssteuerung, in welchen Zeitraum eine Steuerungsaufgabe abgeschlossen werden kann, handelt. Diese Güte kann auch durch Kosten beschrieben werden, die im Prozess entstehen und ermöglicht die weitere Unterscheidung von *weichen* und *harten* Echtzeitbedingungen. Entstehen beim Überschreiten der angenommenen maximalen Reaktionszeit zusätzliche Kosten (z.B. erhöhter Betriebsmitteleinsatz oder Maschine kann nicht voll ausgelastet werden), die jedoch bei weiterer Zunahme der „Überschreitungszeit“ nicht erheblich ansteigen und sich eventuell einem Grenzwert annähern, dann liegen weiche Echtzeitbedingungen vor. Im gegensätzlichen Fall der harten Echtzeitbedingungen führt die Überschreitung der maximalen Reaktionszeit zum Auftreten von außerordentlich hohen Kosten, wie Stillstandszeiten einer Fertigungseinrichtung oder dem Eintritt von schweren Personen- und Sachschäden.

Echtzeitverhalten bedeutet jedoch nicht nur die Einhaltung einer maximalen Reaktionszeit, sondern auch die Beherrschung folgender Anforderungen:

- Reaktion darf nicht vor einem bestimmten Zeitpunkt erfolgen,
- Reaktion darf nicht vor einem minimalen Zeitpunkt und nicht nach einem maximalen Zeitpunkt erfolgen,
- Reaktion muss zu einem genau definierten Zeitpunkt erfolgen.

2.3.2.1.2 Prozessführungsstrategien

Auf der Leitebene wird grundsätzlich angestrebt, komplexe Zielsetzungen vorgeben zu können, die mittels der technischen Prozessführungselemente automatisch umgesetzt werden. ARTMANN ET AL. (1993) [APS93] nennen als wichtigste in der Praxis bewährte Kontrollstrategien die Ablaufsteuerungen, die Regelungen sowie die Optimierung mit Hilfe von Modellvorstellungen.

Ablaufsteuerungen sind feststehende Programme, nach denen ein Prozess geführt wird (z.B. Reinigungsprogramm oder Rezepturen). Rückwirkungen aus dem Prozess, sich ändernde Bedingungen und daraus resultierende Störgrößen bleiben unberücksichtigt.

Regelungen können hingegen aufgrund der Rückmeldungen aus dem Prozess Störungen und Abweichungen selbstständig ausregeln.

Optimierung zielt nicht primär um Ausregelung von Abweichungen, sondern um die Steuerung des gesamten Prozesses auf ein Optimum auszurichten. Ausgehend von einer modellhaften Vorstellung eines Prozessoptimums besteht die Strategie darin, entsprechend der gemessenen Einflussgrößen im Modell die geeigneten Einstellungen zu bestimmen bzw. berechnen, um den Prozess mit der vorhandenen Aktorik auf das Optimum hinführen zu können. Das Optimum kann ein ökonomisches, ein ökologisches, ein zeitliches, ein energetisches oder eine beliebige Kombination, ein „Kompromiss“, davon sein. Innerhalb der Optimierung lassen sich wiederum zwei Strategien unterscheiden:

- Einstellung optimaler Betriebspunkte,
- Fahren optimaler Betriebsprofile.

2.3.2.1.3 Algorithmen und Entwurfsverfahren

Laut VAN STRATEN und WILLIGENBURG (2006) [VV06] ist die Modellierung der zentrale Ansatzpunkt bei einem wissenschaftlich fundierten Entwurf einer Steuerung- und Regelung. Dabei ist es nötig Systemgrenzen zu definieren und zu spezifizieren, wie ein System mit seiner Umgebung interagiert. Dieser Vorgang kann als Modellierung betrachtet werden. Das Ergebnis, das Modell, lässt den Entwickler verstehen, wie das System auf Aktionen, die auf das System einwirken bzw. eingebracht werden, reagiert. Dadurch kann er mit einer abgestimmten Steuerungs- und Regelungstechnik das gewünschte Systemverhalten erzeugen. Auch wenn in der Praxis bisweilen zufriedenstellende Steuerungen und Regelungen ohne eine explizite Modellierung, sondern über „*trial and error*“ mit bereits vorhandenen Standardreglern, erreicht werden können, ist der modellbasierte Ansatz doch der zielgerichtete und dem Stand der Technik entsprechende Weg.

Die Steuerungs- und Regelungstechnik ist weit vorgeschritten mit der Bereitstellung von mathematischen Modellen zur Beschreibung von technischen Prozessen bzw. dynamischen Systemen. KREBS (1991) [Kre91] beschreibt die dabei angewandte Vorgehensweise folgendermaßen: „…. Die Methoden zur Gewinnung derartiger mathematischer Modelle sind sehr weit entwickelt und als Systemidentifikation bekannt; diese beginnt zunächst mit der theoretischen a priori Analyse (physikalisch-strukturelle Beschreibung) und dem daraus resultierenden *Strukturmodell*. Danach erhält man durch Auswertung der gemessenen Eingangs- und Ausgangsgrößen mittels Zustands- und Parameterschätzverfahren ein quantitatives Schätzmodell, das zusammen mit dem Strukturmodell das gesuchte *funktionale mathematische Modell* ergibt. Bei manchen praktischen Fragestellungen begnügt man sich u. U. mit der isolierten Anwendung einzelner Modelltypen. So kann man Strukturmodelle zur Analyse auf Steuerbarkeit und Beobachtbarkeit heranziehen; Black-Box-Modelle setzt man ein, wenn Strukturaussagen schwierig oder unnötig sind und bildet damit auf der Basis einer nichtparametrischen Modellbeschreibung (z.B. Impulsantwort) das Ein-/Ausgangsverhalten in Form eines quantitativen *Verhaltensmodells* nach. …“[19]

Die Beschreibung eines Systemverhaltens wird wesentlich vereinfacht, wenn die Systemeigenschaften der Linearität und Zeitinvarianz vorliegen. Unglücklicherweise sind jedoch die meisten realen weltlichen Systeme nicht von linearer und zeitinvarianter Natur. Oft lässt sich allerdings mittels Linearisierung in einem Betriebspunkt auf die Verfahren der Regelungstechnik für lineare Systeme zurückgreifen und dadurch die Komplexität verringern bzw. überhaupt einer analytischen Lösung zuführen. ALBERTOS UND SALA (2004) [AS04] weisen auf eine weitere vereinfachende Annahme bei der Modellbildung hin, den „*lumped parameters*“[20]. Dies bedeutet, dass nur die Zeit als einzig unabhängige physikalische Variable angenommen und die räumliche Ausdehnung des Systems außer Acht gelassen wird, d.h. die Dynamik eines Systems wird als punktuelles Phänomen betrachtet. Dies widerspricht jedoch den realen Bedingungen von Übertragungsleitungen, großen Behältern usw. In vielen Fällen hilft die räumliche Diskretisierung (wie finite Elemente) angenäherte Modelllösungen zu entwerfen.

Auf der grundsätzlichen Ebene verweist KREBS (1991) [Kre91] darauf, dass viele regelungstechnische Aufgaben auf der Basis analytischer Modelle mit Mitteln der numerischen Informationsverarbeitung in algorithmischen Strukturen nur unzureichend gelöst werden können. Vielmehr erfordert die rechnergestützte oder vollautomatische Bearbeitung

[19] A. a. O., S. 3-4

[20] A. a. O., S. 8

dieser Problemstellungen den Aufbau heuristischer Modelle der Prozesse bzw. der Strategien und Entscheidungen von Fachleuten, die dann nach einer Formalisierung und Abbildung auf dem Rechner mit Methoden der Wissensverarbeitung gelöst werden müssen. Im anwendungsorientierten Teil dieser Arbeit (siehe Kap. 5) wird eine Entwurfsmethodik und Implementierung dieser alternativen Herangehensweise für den Prozessführungsansatz Sensor-Ansatz mit Kartenüberlagerung ausführlich vorgestellt.

Bisher wurde die Regelungstechnik nur in Form von Eingrößensystemen erörtert, d.h. einem System mit einer Eingangsgröße- und einer Ausgangsgröße. Für eine zunehmende Anzahl an technischen Systemen ist diese Voraussetzung nicht gültig, sondern es liegt ein Mehrgrößensystem, auch multivariable Regelstrecke bezeichnet, vor. Dies ist ein System mit mehreren Ausgangs-(Regel)Größen, die von mehreren Eingangs- bzw. Stellgrößen beeinflusst werden. Für eine theoretische Behandlung der Mehrgrößensteuerungs- und Regelungstechnik sei der interessierte Leser auf [AS04] verwiesen.

2.3.2.2 Sensorik

Sensoren sind Schlüsselkomponenten für die Realisierung eines Prozessführungssystems zur Teilflächenbewirtschaftung, wenn es um die Bereitstellung der nötigen Informationen als Online-Sensor oder auch für die Datengrundlage zur Erstellung der Karteninformation geht. Im Folgenden werden wichtige Sensorgruppen aufgeführt und dabei die mit besonderer Bedeutung für den anwendungsorientierten Teil dieser Arbeit ausführlicher vorgestellt. Der Einsatz von Sensordaten zur Kartenerstellung setzt voraus, dass eine exakte räumliche Zuordnung der Messwerte möglich ist. Die bei der Kartierung angewandten Methoden werden im Kapitel 2.3.2.3 erläutert, die dazu nötige IT-Infrastruktur im Kapitel 2.3.2.5.

Die Eignung von Sensoren für den Einsatz in mobilen Landmaschinen wird entscheidend von den rauen Einsatzbedingungen, wie beispielsweise Schock, Vibration, Staub, Temperaturextremen, Chemikalien, Feuchtigkeit, zusammengefasst von der Robustheit und Zuverlässigkeit, bestimmt. Weiterhin entscheidend ist die Messgenauigkeit und Preiswürdigkeit. Vielfach ist dies nur erreichbar durch den Einsatz von altbewährter Sensortechnik, für die sich bereits in über die Landtechnik hinausgehenden Einsatzfeldern eine Art Standardisierung ergeben hat und die durch Massenfertigung auch zu günstigen Preisen verfügbar ist. AUERNHAMMER (1991) [Aue91] beschreibt daher den Leitgedanken für den mit Serienentwicklung befassten Landtechniker folgendermaßen: „Gerade in der Sensortechnik darf deshalb nicht nur nach dem Neuesten und Modernsten gefragt werden. Vielmehr sind die Einsatzbedingungen in der Landwirtschaft und die dort erworbene

Bewährung wesentlich mehr maßgebend, als noch stärkere Miniaturisierung, noch schnellere Reaktion und eventuell noch höhere Auflösung“ [21].

PHELAN ET AL. (2008) [PKL08] differenzieren dementsprechend drei Kategorien von Sensoren für den landtechnischen Einsatz, „*Mature*“, „*Pivotal*“ und „*Emerging*“. Aus Sicht eines globalen Landmaschinenherstellers unterscheiden sie bei den drei Stufen nicht nur den technischen Reifegrad sondern berücksichtigen zusätzlich auch die jeweilige Kenntnis des (erzielbaren) Nutzwertes des Sensors. Ebenso wie der technische Reifegrad sich schrittweise von „*Emerging*“ zu „*Mature*“ entwickelt, geschieht dies in gleicher Weise für das Verständnis für den (betriebswirtschaftlichen) Wert eines Sensors. Erst damit werden gezielte Kosten/Nutzen-Analysen für einen Sensor im Gesamtzusammenhang einer Maschine oder kompletten Prozesskette möglich.

Ausgereifte („*Mature*“)-Sensortechnik steht für die grundlegenden Funktionen der Steuerungs- und Regelungssysteme in Landmaschinen zur Verfügung. Dies sind beispielsweise Sensoren zur Erfassung von Geschwindigkeit, Drehzahl, Kraft, Lage, Füllzustand oder Durchfluss.

Für den Präzisen Ackerbau sind die „*Pivotal*“-Sensoren für die Ortung sicherlich von herausragender Bedeutung. Sie ermöglichen die nötige Georeferenz für teilflächenspezifische Applikationsmaßnahmen sowie die Dokumentation und schaffen erst die Voraussetzung für die Vielzahl der am Markt angebotenen Spurführungssysteme der unterschiedlichsten Genauigkeitsklassen. GRIEPENTROG ET AL. (2006) [GBV06] unterscheiden hierzu absolute und relative Ortungssysteme. Bei einem absoluten Ortungssystem wird die Position der Landmaschine in einem absoluten Koordinatensystem (z.B. UTM oder WGS84) bestimmt. Währenddessen wird bei einem relativen Ortungsverfahren die Position der Landmaschine stets nur relativ zu einem räumlichen Bezugspunkt ermittelt. Die derzeit am weitesten verbreitete Form der absoluten Ortung basiert auf Satellitenortungssystemen, an erster Stelle dem *„Global Positioning System“* (GPS NAVSTAR), das vom amerikanischen Verteidigungsministerium ursprünglich für die militärische Nutzung entwickelt wurde, jedoch auch für zivile Nutzung global zur Verfügung gestellt wird. Seit Mai 2000 steht dem zivilen Nutzer GPS mit einer Systemgenauigkeit mit Fehlern von 15 - 20 m zur Verfügung. Durch den Einsatz von differenziellen GPS-Techniken lässt sich der Einsatz auf weniger als 1 - 3 m im mobilen Einsatz reduzieren und hochgenaue „Real-time Kinematik DGPS“-Systeme machen die Positionsbestimmung mit Fehlern kleiner 10 cm möglich [DD06].

[21] A. a. O., S. 14

Sowohl für den reinen Kartierungsansatz als auch für den Sensor-Ansatz mit Kartenüberlagerung nimmt die teilfächenspezifische Ertragsmessung eine besondere Rolle ein. Der Ertrag ist zum einen ein Maß für den Erfolg der Bewirtschaftungsmaßnahmen des zurückliegenden Wirtschaftszeitraums, zum anderen hilft er bei der Beschreibung von Heterogenität auf der Bewirtschaftungsfläche. Für die kontinuierliche Durchsatz- und Ertragsermittlung kommen unterschiedliche Messprinzipien zum Einsatz [ADMRW93], [RK96]. Im Mähdrescher ermitteln diese Sensoren die Durchflussmenge des Erntegutes im Elevator. Diese Menge wird über eine Fläche integriert, die sich durch die eingestellte Arbeitsbreite und den jeweils zurückgelegten Weg bestimmen lässt und dann einem auf dem Fahrweg gelegenen Positionswert zugeordnet. Das integrative Verfahren und der Weg, den das Gut im Mähdrescher vom Schneidwerk bis zum Elevator durchläuft, verursachen unterschiedlichste Fehler in der Ertragsmessung. STEINMAYER (2002) [Ste02] hat diese ausführlich dargelegt. Derzeit weisen die am Markt verfügbaren Sensoren ein ähnliches Fehlerniveau mit relativen Standardabweichungen zwischen 3 % und 4 % auf [DD06]. Diese durchschnittlichen Fehlerwerte wie auch das dynamische Verhalten der Sensoren sind für die weitere Verarbeitung zu Ertragskarten unbedingt zu berücksichtigen. Untersuchte und angewandte Methoden zur Ertragskartierung sind im Kapitel 2.3.2.3 aufgeführt.

Eine gleichzeitige Feuchtemessung verhindert, dass Schwankungen in der Erntegutfeuchte als Ertragsunterschiede registriert werden. Die eingesetzte Sensorik gründet weitestgehend auf einem kapazitiven Messverfahren. Wird eine absolute Messgenauigkeit kleiner 2 % angestrebt, so ist diese mittels der Nah-Infrarot-Spektroskopie realisierbar und bereits für selbstfahrende Feldhäcksler als Serienprodukt eingeführt [WK07]. Diese Technik weist auch erhebliches „*Emerging*“-Potential auf, für den Fall, dass sie für die Inhaltsstoffbestimmung des Erntegutes auf der Erntemaschine verwendet wird [EIH06], [RHHKH08]. Auch die grundsätzliche Eignung zur Online-Inhaltstoffbestimmung bei Gülle konnte gezeigt werden [ADH06].

DOLESCHEL UND DEMMEL (2006) [DD06] fordern aus Sicht des Pflanzenbauers und Landtechnikers für die Realisierung und Optimierung vieler notwendiger Anwendungen des Präzisen Ackerbaus weitere bzw. weiterentwickelte Sensorsysteme:

- Sensoren zur Online-Ermittlung des Ernährungszustandes (Wasser, Nährstoffe) und des Gesundheitsstatus (Krankheiten, Schädlingsbefall) der Pflanzen,
- Sensoren zur Online-Ermittlung bodenphysikalischer (Dichte, Porengrößenverteilung) und bodenchemischer Parameter (Bodenfeuchte, Nährstoffgehalte),
- Sensoren zur „Sicherung“ autonomer Landmaschinen im Feld.

Die angeführte Sensorik ist derzeit der Kategorie „*Emerging*“ zuzuordnen und Gegenstand zahlreicher Forschungsvorhaben. Die größten Fortschritte bei der Entwicklung geeigneter (Online)-Sensorik wurden sicherlich im Bereich der optischen Erfassung des Pflanzenzustandes und der optischen Unkrauterkennung erreicht. OMASA (2006) [Oma06] hat die wichtigsten technologischen Möglichkeiten zusammengestellt, mit denen durch bildgebende, nah- und fernerkundungstechnische Verfahren phytobiologische Informationen von Pflanzen gewonnen werden können. Bei Naherkundung kann das Sensorsystem direkt an der Landmaschine montiert sein, bei Fernerkundung wird die Sensorik in der Regel von Satelliten, Flugzeugen oder UAV getragen [OOS06].

HEEGE (2007) [Hee07] verweist darauf, dass die optische Erfassung der pflanzlichen Stickstoff-Versorgung deutlich billiger und schneller als durch chemische Analyse möglich ist und daher zunehmende Anwendung bei der teilflächenspezifischen Düngung findet. Die am Markt angebotenen Sensorsysteme arbeiten nach zwei verschiedenen Wirkprinzipien, die HEEGE (2007) [Hee07] als „Kieler Verfahren“ und „DLR-Verfahren“ nach physikalischen und pflanzlichen Aspekten unterscheidet. Beim „Kieler Verfahren“ werden spezielle Wellenlängen aus dem rot-infraroten Bereich der pflanzlichen Reflexion gemessen und daraus auf die Chlorophyll-Konzentration in den Blättern plus den Blattflächenindex geschlossen. Beim „DLR-Verfahren“ wird die Chlorophyll-Fluoreszenz, induziert durch einen Rotlaser, gemessen und daraus auf die Chlorophyll-Konzentration in den Blättern plus zuweilen die Deckfläche der Frucht geschlossen. Technische Umsetzungen dieser Verfahren in auf Landmaschinen montierbaren Sensorsystemen existieren beispielsweise mit dem „Yara N-Sensor“ der Fa. Yara International ASA, Oslo, Norwegen für das „Kieler Verfahren“ und mit dem „MiniVegN-Sensor“ der Fa. Fritzmeier Umwelttechnik, Großhelfendorf, Deutschland für das „DLR-Verfahren“.

Neben all den optischen Sensorsystemen wurde auch ein rein mechanischer Pendelsensor zur Bestimmung der Bestandesdichte realisiert und weiterentwickelt [EJO07]. Über den Grad der Auslenkung eines Pendels, verursacht durch den Kontakt mit den Pflanzen, wird auf die Biomassemenge geschlossen und dementsprechend eine Düngermenge abgeleitet.

Da für den anwendungsorientierten Teil dieser Arbeit das reflexionsoptische Messprinzip eine vorrangige Rolle spielt, soll im Folgenden die Technik ausführlicher erläutert werden. Mit einer Sensoreinheit wird die Reflexion des Pflanzenbestandes im roten und nah-infraroten Spektralbereich erfasst. Die Reflexion des sichtbaren Lichtes sinkt mit steigender N-Versorgung aufgrund der zunehmenden Chlorophyllkonzentration in den Blättern, die zu höherer Photosynthese und damit zu zunehmender Absorption führt. Im infraroten Bereich

hingegen ist die Reflexion erhöht durch das Mehr an Biomasse aufgrund einer besseren N-Versorgung. Ein Blick auf die Reflexionskurve zeigt daher einen steileren Übergang im Bereich des Übergangs vom roten zum nahinfraroten Licht, der als „*red edge*“ bekannt ist. Damit geht eine Verschiebung des Hauptwendepunktes (*Red Edge, Red Edge Inflection Point* (*REIP*)) bei unterschiedlichen Chlorophyllgehalten einher. Über diesen Effekt kann somit aus der Lage des Hauptwendepunktes auf den Ernährungszustand geschlossen werden [Reu97]. Aus der Wellenlänge des Hauptwendepunktes erfolgt nach der Näherungsformel von GUYOT ET AL. (1988) [GBM88] die Berechnung dieses Vegetationsindexes REIP:

$$REIP = 700 + 40\frac{(R_{670} + R_{780})/2 - R_{700}}{R_{740} - R_{700}} \qquad (2.1)$$

LIEBLER (2003) [Lie03] und SCHMIDHALTER ET AL. (2003) [SJBGMM03] stellten gute Korrelationen dieses Vegetationsindexes mit der N-Aufnahme und dem N-Gehalt fest. Ein Nachteil dieses reflexionsoptischen Verfahrens ist, dass das vom Boden reflektierte Licht, der sich im Sichtfenster des Sensors befindet, als Falschinformation miterfasst wird. Daher sollte im Sichtfeld ein weitgehend geschlossener Pflanzenbestand vorliegen. Eine weitere Voraussetzung für erfolgreiche Spektroskopie sonnenlichtbasierter Reflexionsmessungen ist die gleichzeitige Messung der einfallenden Sonnenstrahlung und der Bestandesreflexion, da die Lichtverhältnisse im freien Feld oftmals wechseln. Da Pflanzenbestände keine Lambertschen Reflektoren sind, ändert sich ihr Reflexionsverhalten mit verändertem Zenitwinkel [Reu97]. Als besonders vorteilhaft erwiesen sich in diesem Zusammenhang Sensorsysteme, mit einer Messoptik, die in vier um etwa 90° versetzte Richtungen sieht („oligo-view“) und deren Signale über mehrere Einkoppelungsoptiken gebündelt werden. Die eingehenden Signale werden zu einem Wert verrechnet. MISTELE (2005) [Mis05] konnte damit zeigen, dass die Messungen unabhängiger vom Blickwinkel und dem Sonnenstand werden.

Während im Bereich der Online-Sensoren zur Bestimmung von Pflanzeneigenschaften und –zuständen bereits erste am Markt eingeführte Sensorsystemen existieren und sich auch langsam ein Verständnis über den Wert dieser Sensoren herausbildet, hat der Bereich der Online-Sensoren zur Ermittlung bodenphysikalischer und bodenchemischer Parameter bisher noch nicht diese Stufe erreicht. Ein Großteil der Forschungsarbeiten und Produktentwicklungen ist bisher im Bereich der Bestimmung der stabilen und temporär stabilen Bodeneigenschaften angesiedelt. Die konventionelle Bodenuntersuchung mittels

Beprobung, einem destruktiven Verfahren, ist zeit- und kostenaufwändig. Ziel zahlreicher Forschungs- und Entwicklungsarbeiten ist daher die Anwendung von nicht-destruktiven Verfahren zur Erfassung der Bodenvariabilität, was wiederum eine große Flächenleistung mit hoher Messwertdichte erlaubt. Im Bereich des Präzisions-Ackerbaus liegt ein Schwerpunkt der Untersuchungen auf geophysikalischen Verfahren zur Kartierung von Bodeneigenschaften für die teilflächenspezifische Bewirtschaftung. Vorzugsweise wird dabei das geophysikalische Verfahren der Bodenleitfähigkeitsmessung angewandt. Zwei grundlegend unterschiedliche Funktionsprinzipien kommen zum Einsatz [DT03]. Eine Variante basiert auf einem gleichstromelektrischen Wirkprinzip, das einen Bodenkontakt erfordert. Mit dem „Veris 3100-System“ der Fa. Veris Technologies, Salina, KS, USA wurde eine technische Umsetzung realisiert, bei der über sechs Scheibenseche der nötige Bodenkontakt hergestellt wird. Weitaus häufiger kommt jedoch die zweite Variante, die auf einem elektromagnetischen Verfahren beruht und somit keinen Bodenkontakt erfordert, zur Anwendung. Technisch hat die Fa. Geonics Limited, Mississauga, Ontario, Kanada dies in Form des Sensors „EM38“ umgesetzt. Das verwendete physikalische Prinzip zur Messung der (scheinbaren) elektrischen Leitfähigkeit und die praktische Verwendung dieses Sensors sind bei [CL03] ausführlich beschrieben. Die elektrische Leitfähigkeit des Bodens hängt vom volumetrischen Wassergehalt, der Konzentration gelöster Stoffe in der Bodenlösung und der Art und Menge der Tonminerale im Boden sowie dem Gehalt an organischer Substanz und der Trockenraumdichte des Bodens ab [Sch01].

Für den Standort „Scheyern“ (45 km nordwestlich von München) stellte DURLESSER (1999) [Dur99] fest, dass die Variation der elektrischen Leitfähigkeit wesentlich auf den Tongehalt, den Bodenwassergehalt und die Bodentemperatur zurückzuführen ist. Der Einfluss der Temperatur kann durch eine Korrekturformel auf einen Normwert bei 25°C relativiert werden. Jedoch ändert sich mit der vegetations- und reliefbedingten Änderung des Bodenwassergehaltes nicht nur das Niveau sondern auch die Variation der scheinbaren elektrischen Leitfähigkeit [SDK01]. Das Messergebnis spiegelt somit den Einfluss verschiedener Bodeneigenschaften wider, wobei jedoch die komplexe Interaktion der beteiligten Faktoren bei der Signalzusammensetzung und der Signalausprägung in Abhängigkeit der Bodentiefe noch nicht abschließend geklärt sind. Des Weiteren sind mit der Messung der scheinbaren elektrischen Leitfähigkeit nur indirekte Rückschlüsse auf die Bodeneigenschaften möglich. Das Hauptpotential der scheinbaren elektrischen Leitfähigkeit liegt in der Möglichkeit (einheitliche) Managementzonen abzuleiten. Für eine absolute Skalierung der Messung sind jedoch gezielte Bodenbeprobungen zum Referenzieren

unumgänglich.

Neben der elektrischen Bodenleitfähigkeitsmessung mittels EM-38 spielt für den anwendungsorientierten Teil dieser Arbeit die Zugkraftmessung eine besondere Rolle. Bei diesem Verfahren wird versucht, durch Messungen des Zugkraftbedarfes eines Bodenbearbeitungsgerätes Bodenunterschiede festzustellen. Interessant an diesem Verfahren ist, dass auf bereits in modernen Traktoren eingebaute „*Mature*"-Sensortechnik zurückgegriffen wird. Zur Messung der Zugkraft bei Arbeitsgängen der Bodenbearbeitung werden dabei die Kraftmesssignale der elektronischen Hubkraftregelung aufgezeichnet, um nach anschließender Kartierung Information über die Bodenvariabilitäten zu erhalten. Eine ausführliche Darstellung der technischen Zusammenhänge und des Systemaufbaus hat SCHUTTE (2005) [Sch05] beschrieben. Er konnte wie auch ROTHMUND ET AL. (2003) [RZAD03] eine gute Korrelation zwischen Zugkraftbedarf und Tongehalt im Oberboden nachweisen. Zugleich zeigten ROTHMUND ET AL. (2003) [RZAD03] die Machbarkeit und Vorteile einer automatischen Prozessdatendokumentation für diese Größe unter Nutzung des ISOBUS. Jedoch weist SCHUTTE (2005) [Sch05] darauf hin, dass in der landwirtschaftlichen Praxis bei der Anwendung der Zugkraftkartierung häufig eine störende Beeinflussung des Zugkraftbedarfes durch Schwankungen in Arbeitstiefe, Arbeitsgeschwindigkeit sowie Bodenfeuchte und Bodendichte zu erwarten ist. Mit einem von ihm vorgestellten Ansatz zur Normierung der Zugkraftwerte kann der Einfluss von Arbeitstiefe und -geschwindigkeit gut berücksichtigt werden. Der Einfluss der Bodenfeuchte konnte aufgrund fehlender Sensorik nicht erfasst und berücksichtigt werden. Einflüsse der Bodenverdichtung, hervorgerufen durch die Maschinen, speziell am Vorgewende, konnten ebenfalls festgestellt werden und erschweren die Interpretation der Messwerte.

Erste Ansätze zur Messung der Bodenfeuchte mittels einer Online-Sensorik basierend auf GPR (*Ground Penetrating Radar*) wurden beispielsweise von PAUL UND SPECKMANN (2002) [PS02] untersucht. Je nach gewählter Frequenz können unterschiedliche Eindringtiefen erreicht werden. So lässt sich mit einer Frequenz von 24.1 GHz nur die Feuchtigkeit der Bodenoberfläche erfassen, bei 5.8 GHz sind 5 cm Eindringtiefe erreichbar. Gemäß PAUL UND SPECKMANN (2002) [PS02] sollten sich höhere Eindringtiefen bis zu 1 m mit tieferen Frequenzen (0.1 bis 1.5 GHz) erreichen lassen. Jedoch ist mit vermehrten Störsignalen durch Vegetation und Oberflächenrauhigkeit zu rechnen. SCHULZE LAMMERS ET AL. (2006) [SYM06] erprobten ein horizontales Penetrometer mit integriertem kapazitivem Feuchtigkeitsmesser zur Online-Bestimmung des Wassergehalts der oberen Bodenschicht und des mechanischen Widerstands. Bei einer Fahrgeschwindigkeit von 1 m/s und einer Messtiefe

von 15 cm im Boden wurde eine dynamische Genauigkeit von +/- 3 % für den gravimetrischen Wassergehalt ermittelt. Einen über die Wassergehaltbestimmung hinausgehenden Ansatz verfolgten MALEKI ET AL. (2006) [MMRD06]. Mit Methoden der NIRS erprobten sie die Online-Boden-Nährstoffgehalt-Bestimmung, um einen Sensor-Ansatz zur kleinräumigen Phosphatdüngung zu realisieren.

Hinsichtlich weiterer Forschungsarbeiten und -ergebnisse im Bereich der Sensortechnik für Bodeneigenschaften für Zwecke des Präzisions-Ackerbaus sei auf die Zusammenstellungen bei [AHMU04] und [Sch05] verwiesen. Abschließend sei erwähnt, dass neben den Verfahren aus dem Bereich der Geophysik und -chemie auch Techniken der Nah- und Fernerkundung Anwendung finden. So haben z.B. SELIGE ET AL. (2003) [SNS03] ein Spektrometer sowohl im Feld als auch in der Flugzeug-Fernerkundung eingesetzt und damit gute Korrelationen zum Kohlenstoffgehalt und Tongehalt des Oberbodens herstellen können. Die Autoren sehen ein Potential für die Fernerkundung zur Kartierung der organischen Bodensubstanz und des Tongehaltes im Oberboden.

Der Mangel an am Markt verfügbaren „Online-Bodensensoren“ könnte durch eine weitere Kategorie von Sensoren an Wichtigkeit verlieren. Getrieben durch den Fortschritt mobiler Informationstechnologie eröffnen die drahtlosen Sensornetzwerke, die sogenannten *Wireless Sensor Networks* (WSN) neue Möglichkeiten bei der Zustandsüberwachung von Boden- oder Pflanzenparametern, wenn keine fahrzeuggetragene Sensortechnik oder Fernerkundungssensoren verfügbar sind bzw. deren Kosten in keinem Verhältnis zum erwartbaren Nutzen stehen. ADAMCHUK ET AL. (2011) [ARSS11] führten den WSN-Ansatz als weitere Möglichkeit für Online-Boden-Sensoren an und haben wenige Beispiele dafür zusammengestellt. Der Unterschied zu aufwendigen stationären Beobachtungseinrichtungen (z.B. für Wetter, Boden, Pflanzen), wie sie in Forschungs- und Saatzuchtinstitutionen eingesetzt werden, sind ihr sehr kostensensitiver und energieeffizienter bzw. -autonomer Aufbau, verbunden über Funkkommunikation. Dies ermöglicht ein gesamtes Feld an den die Heterogenität repräsentierenden Stellen mit einem Netzwerk dieser einfachen Sensoren zu überziehen und ein Real-time-Abbild der Messergebnisse zu beispielsweise Mikroklima (Boden- und Bestandestemperatur), Bodeneigenschaften (Wassergehalt) oder Pflanzeneigenschaften (Bestandeshöhe) zu erhalten. Aufgrund der selbstorganisierenden Funkkommunikation (Stichwort: „Ad-hoc-Netzwerke“) und einer entsprechenden Integration in übergeordnete IT-Infrastrukturen entfällt eine teure Kabelinstallation, die stets in Gefahr ist, von Landmaschinen während der Arbeitsvorgänge beschädigt zu werden und damit zum Ausfall des Systems zu führen. Exemplarisch haben VELLIDIS ET AL. (2006) [VTB06] die

Machbarkeit dieses Ansatzes für eine Baumwollfeld-Beregnungsapplikation gezeigt. Dabei ermittelten sie den Wassergehalt und die Temperatur des Bodens über ein WSN bestehend aus 20 Sensorknoten an ausgewählten Positionen im Feld.

Gerade die letzten Jahre haben ein verstärktes Interesse der Agrarsystemtechnik an Lösungen gezeigt, die auf drahtloser Kommunikationstechnik beruhen. Neben den WSN und „virtuellen Referenzstationen“ für GPS stehen dabei vor allem die Funkverbindungen im Bereich der Farm Management Informationssysteme (FMIS) zu den mobilen Systemen und der (Flotten-)Logistik im Vordergrund (vgl. Kap. 2.3.2.5). Diese wachsende Bedeutung greift ein neues ISO-Standardisierungsvorhaben auf. Das ISO-Standardisierungsgremium ISO/TC23/SC19 „*Agricultural Electronics*“ hat die Arbeitsgruppe („*Working Group*“) WG 5 ins Leben gerufen, um einen internationalen Standard für „*Wireless Communication in Agriculture*“, d.h. drahtlose Kommunikationssysteme in der Landwirtschaft, zu erarbeiten. Dieser zukünftige Standard wurde bei der ISO bereits als ISO 16867 [NWIISO16867], [Kra11] registriert. Es ist ein neunteiliger Aufbau der Norm mit dem Titel „*Wireless Communication in Agriculture*; *Standardization of wireless communication for Fleet Management, including implement synchronization*“ vorgesehen [Kra11]:

- Part 1: General; Standards for wireless communication in agriculture,
- Part 2: Long range networks: session, presentation and application layer for fleet management,
- Part 3: Long range networks: session, presentation and application layer for implement synchronization,
- Part 4: Long range networks: communication layer,
- Part 5: Short range networks: application layer,
- Part 6: Short range networks: communication layer,
- Part 7: Ultra short range networks: session and presentation layer,
- Part 8: Ultra short range networks: communication layer,
- Part 9: Webservices for wirelessly obtained agricultural data.

In der Einleitung und Motivation wird die Standardisierung der Methode und des Formats der Datenübertragung in nachfolgend genannten drahtlosen Netzwerken zum Ziel erhoben. In [NWIISO16867] sollen zum einen drahtlose Kommunikationsnetzwerke für kurze Reichweiten zwischen abgesetzten Sensoren und Gateways oder Netzwerken zum anderen für lange Distanzen zwischen Gateways und anderen ECU’s und Basisstationen für landwirtschaftliche Anwendungen spezifiziert werden.

2.3.2.3 Kartierung

Im Präzisions-Ackerbau spielt die georeferenzierte bzw. kartierte Form von Information über

Standortheterogenität als auch der teilflächenspezifisch definierten Handlungsmaßnahmen eine besondere Rolle. Grundsätzlich werden diese, im Folgenden *Precision Farming-Karten* genannt, als Eingangsinformation für *Decision Support*-Systeme, zur Datenspeicherung und als Ausgangsinformation von Entscheidungsprozessen genützt. Der Einsatz der *Precision Farming-Karten* ist weitreichend und erstreckt sich über Ertragskartierung, Unkrautkartierung, Bodeneigenschaftskartierung, Vermessungswesen (z.B. Grenzen, Hindernisse, DTM), Fernerkundungskarten sowie den teilflächenspezifischen Applikationskarten (Düngung, Pflanzenschutz, Aussaat, Beregnung) wie auch der Dokumentation der durchgeführten Behandlungsmaßnahmen (*as-applied maps*). Von einem informationstechnischen Blickwinkel handelt es sich bei den Karten um eine Zusammenstellung von Datenobjekten mit einer räumlichen, evtl. zeitlichen und einem oder mehreren Eigenschaftsattributen.

Die Geoinformatik kennt zwei grundlegend unterschiedliche Modellgruppen zur Beschreibung der Georeferenz, die Rastermodelle und die Vektormodelle. Der folgende Abschnitt fasst hierzu den Vergleich beider Datenwelten durch BARTELME (1995) [Bar95] zusammen.

Ein Rastermodell geht von der Annahme aus, dass der Interessensbereich in Teilflächen mit homogener Thematik aufgeteilt werden kann. Die gängige Fachbezeichnung ist als Mosaik oder Tesselation bekannt. Eine spezielle (und auch häufigste Form) des Mosaiks ist die quadratische oder zumindest rechteckige Aufteilung in Rastermaschen, auch Gittermaschen oder Zellen genannt. Das (konzeptionelle) Raster kann in eine (logische) Matrix abgebildet werden. Thematische Bewertungen schlagen sich in numerischen Werten für die Matrizenelemente nieder. Rastermodelle vertragen sich gut mit einem „*Layer*"-Konzept. Die Überlagerung mehrerer thematischer *Layer* ist in der Regel „einfach" durchzuführen, solange die Rasteraufteilung und damit die Größe, Ausrichtung und Position der einzelnen Maschen in allen *Layer* identisch ist. Der Vergleich und die Verschneidung mehrerer *Layer* ist dann besonders einfach.

Vektormodelle bauen auf Punkten und Linien auf. Flächen werden durch geschlossene Folgen von Linien, falls nötig auch mit Aussparungen, modelliert. Diesen elementaren geometrischen Elementen Punkt, Linie, Fläche (sogenannten Features) werden thematische Charakteristika (Attribute) zugeordnet. Da eine implizite Zuordnung, der Attribute zu den Features, wie im Rastermodell, nicht ausreicht, müssen explizite Verweise gemacht werden. Daher wird das Vektormodell auch mitunter als georelationales Modell oder „*feature-based model*" bezeichnet.

Zusammengefasst folgert BARTELME (1995) [Bar95], dass die Rastermodelle für die Beschreibung flächiger Sachverhalte weit besser geeignet sind als die Vektormodelle, die ihrerseits wieder eine Stärke für linienförmige Verbindungen aufweisen. Eine Gegenüberstellung der wichtigsten Eigenschaften beider Modelle fasst Tabelle 2.5 zusammen [Bil99].

Tabelle 2.5: Gegenüberstellung der Eigenschaften von Rasterdaten und Vektordaten (nach [Bil99])

Rasterdaten	Vektordaten
Pixel als graphische Grundstruktur	Punkt und Linie als graphische Grundstrukturen, Fläche als geschlossener Linienzug
Flächenhafte Betrachtungsweise, dadurch Vorzüge in diesem Bereich	Daten nach Objektlinien geordnet, dadurch linienhafte Betrachtungsweise
Ordnung nur nach der Position der Pixel	Daten nach Objektlinien geordnet
Logische Datenstrukturierung und Objektbezug sehr eingeschränkt	Logische Datenstrukturierung und Objektbezug leicht möglich
Einfache Datenerfassung, kurze Erfassungszeiten	Punktuelle Datenerfassung durch den Einsatz von bewährten Methoden, jedoch hohe Erfassungszeiten
Große Datenmengen, dadurch hoher Rechenaufwand	Geringe Datenmengen, kurze Rechenzeiten

Um dem Ziel der sachgerechten Modellierung der Variabilität eines Feldmerkmales im Rahmen der Kartierung nachzukommen, ist es in der Regel nötig, aus Punktinformationen auf die Situation des gesamten Feldes zu schließen [LSL02]. Dies gilt sowohl für die geringe Anzahl der Analysedaten bei „konventioneller“ Bodenbeprobung, wie auch für die Ertragsdaten oder Daten mit hoher Erfassungshäufigkeit wie die Zugkraftmesswerte, gewonnen im Rahmen der automatisierten Prozessdatendokumentation.

Die Verarbeitungsschritte und Problemfelder bei der Kartierung von *Precision Farming-Karten* soll im Folgenden an der Ertragskartierung aufgezeigt werden. Dabei wird der folgende Absatz nahezu ohne Änderungen aus [AOMSSW06][22] zitiert. Im Präzisions-Ackerbau kommt der Ertragsermittlung und daraus abgeleiteten Ertragskartierung eine zentrale Bedeutung zu. So stellt die Ertragskartierung neben der Bodenanalyse die wichtigste Eingangsgröße beim „Kartierungs-Ansatz“ dar. Für den im anwendungsorientierten Teil dieser Arbeit verfolgten Sensor-Ansatz mit Kartenüberlagerung übernimmt sie sogar eine Doppelfunktion: Zum einen dient sie der Ableitung des Ertragspotentials und damit als Zielgröße für die Online-Steuerung. Zum anderen liefert sie aus langjährigen

[22] A. a. O., S. 60-61

Erfassungsreihen Begrenzungswerte im Hinblick auf maximal erzielbare lokale Erträge. Fehlerquellen und Genauigkeitsklassen der Ertragsmesssensoren und Ortungssysteme wurden bereits im Kapitel 2.3.2.2 erörtert. Um zu einer möglichst weitgehend objektiven Vergleichbarkeit von Ertragskarten zu gelangen, schlägt STEINMAYER (2002) [Ste02] einen Kartierungsalgorithmus vor, der in drei aufeinander folgenden Segmenten eine Vielzahl von Einzelschritten enthält:

1. Datenvorbereitung,
2. Datenbereinigung über den relativen mittleren Ertrag des Schlages mit weiteren acht konkreten Maßnahmenschritten,
3. Ableitung des Kartierungsalgorithmus nach Ertragsklassen im Hinblick auf technisch mögliche und ökonomisch sinnvolle Applikationsdifferenzierungen mit Erstellung der Ertragskartierung in Raster- oder Konturgrafik unter Zuhilfenahme geo-statistischer Softwarewerkzeuge.

Für den Schritt der Datenbereinigung werden von diversen Autoren weitere Korrektur/Filteralgorithmen vorgeschlagen [Bla99], [NMD03]. Für die im dritten Schritt zur Erstellung der Ertragskartierung in der Praxis angewandten (geo-statistischen) Interpolationsverfahren haben (LUDOWICY ET AL., 2002) [LSL02] die Verfahren „*Inverse Distanz*“, „*Kriging*“, „*Nearest Neighbor*“ und „*nichtlineare Regression*“ identifiziert. Die Autoren verweisen weiterhin darauf, dass am häufigsten das *Kriging*- und das *Inverse Distanz*-Verfahren zum Einsatz kommen. Dem *Kriging*-Verfahren schreiben sie dabei theoretisch das größere Potential zu, der Realität am nächsten liegende Ergebnisse zu liefern.

BACHMAIER UND AUERNHAMMER (2005) [BA05] sehen bei diesem Kartierungsalgorithmus die anisotrope Ausprägung der Ertragsmerkmale nicht oder nur unzureichend berücksichtigt. Deshalb schlugen sie den neu entwickelten Kartierungsalgorithmus „*Paraboloides Butterfly Fitting*“ vor. Der Name ist abgeleitet von der grundsätzlich eingesetzten Methode, entlang der Fahrspuren eine Anpassung in Form eines Schmetterlings zu realisieren.

Dieser Ansatz versucht gegenüber dem rein theoretisch begründeten „*Kriging*-Ansatz“ die Ertragsausbildung eines natürlichen Bestandes zu berücksichtigen und die Messcharakteristik des Sensors einzubeziehen. Die Methode beruht auf dem Angleichen von Ertragsstrukturen durch Paraboloide. Der Ertrag zu jedem Punkt (x, y) einer Ertragskarte entsteht darin als der gefittete Wert einer zweidimensional-quadratischen Regression der Messdaten auf den Koordinaten (x, y) einer Umgebung. Als beste Form für diese Umgebung wurde eine „Auswahlregion“ symmetrisch zu einer Fahrspur abgeleitet, deren Form einem Schmetterling ähnelt. Diese Tatsache stand Pate bei der Namensvergabe für diese Methode. Der Vorteil dieses Verfahrens liegt in seinem Potential aus Messdaten unterschiedlicher Bestände und unterschiedlicher Sensoren wirklich vergleichbare Ertragskartierungen erstellen zu können.

2.3.2.4 Aktorik

Grundaufgabe der Applikationstechnik der Düngung ist es, dass mineralische und organische Düngemittel exakt dosiert und gleichmäßig ausgebracht werden. Die Ausbringtechniken für Mineraldünger lassen sich in Technik für feste Mineraldünger und flüssige Mineraldünger unterscheiden. Mit Hinsicht auf den anwendungsorientierten Teil dieser Arbeit sei bezüglich der Ausbringtechniken für flüssige Mineraldünger und flugzeuggetragene Düngetechnik auf die einschlägige Literatur verwiesen [HSS99]. Auch die Applikationstechnik für organischen Dünger ist nicht Gegenstand dieser Arbeit und kann beispielsweise bei [DN06] eingesehen werden.

Die wichtigsten Bauformen von Düngerstreuern zur Applikation von mineralischem (granulösen) Dünger sind die Kastenstreuer, die Wurfstreuer (Schleuderstreuer), die (pneumatischen) Auslegerstreuer und die Großraumstreuer [DN06].

Für die Ausbringung von mineralischem Stickstoffdünger kommen vorrangig die Wurfstreuer in der Form der Zentrifugaldüngerstreuer mit zwei Streuscheiben wie auch die Ausleger-Pneumatikstreuer zum Einsatz.

Beim Funktionsprinzip ist ihnen gemeinsam, dass bei allen Mineraldüngerstreuern der Dünger aus einem Vorratsbehälter zur Dosiervorrichtung gelangt. Diese besteht bei Wurfstreuern (in der Regel) aus einer im Querschnitt verstellbaren Auslauföffnung, beim Ausleger-Pneumatikstreuer ist sie mit Nockenrädern ausgestattet, deren Drehzahl stufenlos verstellt werden kann. Diese „Beschickungsvorrichtungen“ bewirken, dass eine mehr oder weniger exakt dosierte Düngermenge dem Streuorgan zugeführt wird, welche das gleichmäßige Verteilen des Düngers übernimmt.

Beim **Zentrifugalstreuer** sind dies zwei sich drehende Streuscheiben. Dabei wird das daraus resultierende Streubild vor allem durch die Scheibendrehgeschwindigkeit, die Länge und Form der Wurfschaufeln, die Neigung der Scheiben und der Aufgabestelle des Düngers auf der Scheibe bestimmt. Je nach Drehrichtung der Scheiben ergeben sich quer zur Fahrrichtung ein dreiecksförmiges Streubild bei nach innen drehenden Scheiben und ein trapezförmiges bei nach außen drehenden Streuscheiben. Um eine gleichmäßige Düngerverteilung auf der gesamten Fläche zu erreichen, muss daher beim Anschlussverfahren überlappt werden. Arbeitsbreiten bis zu 36 m werden dabei erreicht. Für eine theoretische Darstellung zu Streubildanalyse und den Flugbahnen (Trajektorien) der Düngerstreuerpartikel sei auf [HSS99] verwiesen.

Beim **Ausleger-Pneumatikstreuer**, der ähnlich wie eine Pflanzenschutzspritze mit einzelnen Teilbreiten ausgestattet ist, wird die zugeteilte Düngermenge in die Verteilrohre der einzelnen

Teilbreiten eingespeist. In diesen übernimmt ein Druckluftstrom den Transport der Düngerpartikel zu den einzelnen Pralltellern, die je nach Bauart/Modell mit etwa 50 cm bis 100 cm Abstand gleichmäßig über die gesamte Auslegerbreite angebracht sind. Von dem einzelnen Prallteller aus gesehen, ergeben sich dreiecksförmige Streubilder, die aber über die einzelne Teilbreite oder die gesamte Arbeitsbreite eine nahezu homogene Düngerausbringung ergeben. Im Gegensatz zum Schleuderstreuer ist beim Ausleger-Pneumatikstreuer die Arbeitsbreite somit weitgehend durch die Breite des Auslegers definiert. Das Streubild, das etwas über die Auslegerbreite (bis zu 24 m) hinausgeht, ist durch die beiden äußeren Prallteller verursacht und erzwingt auch hier eine Überlappung im Anschlussverfahren jedoch bei Weitem nicht so ausgeprägt wie beim Schleuderstreuer. Die einzelnen Teilbreiten lassen sich ein- und ausschalten und ermöglichen damit exaktes Grenz- bzw. Parzellenstreuen sowie die Reaktion auf ungleichmäßige Feld- bzw. Managementzonen.

Damit wird ersichtlich, dass sich beide Systeme in wichtigen Punkten unterscheiden und deshalb der Genauigkeit der Applikationstechnik eine große Bedeutung zukommt. So dürfen nach DLG-Prüfungsrichtlinien die Abweichungen vom Sollwert im Mittel nur ± 10 % (maximal ± 30 %) betragen [DN06]. Zur Beurteilung der Streuqualität wird der Variationskoeffizient herangezogen, für die Methode zur Bestimmung wird auf [HSS99] verwiesen. Dabei wird bei einem Wert von kleiner 5 % von sehr guter Qualität, bei 5 % bis 10 % von einem guten Ergebnis, bei 10 % bis 15 % von zufriedenstellender und bei größer 15 % von unzureichender Streuqualität ausgegangen. Für die einzelnen Charakteristika Längsverteilung, Querverteilung, sind verschiedene Fehlerursachen auszumachen [DN06]. Unterschiedliche Fahrgeschwindigkeiten, ungleichmäßige Düngerzufuhr zum Streusystem, Bodenunebenheiten und unterschiedliche Bodenoberflächen wie Hanglagen haben Einfluss auf die Qualität der Längsverteilung. Die Querverteilung wird hauptsächlich durch die Bauart und Funktion des Streusystems, die gewählten Geräteeinstellungen (Kalibrierungen), die physikalischen Eigenschaften des Düngers, wie Korngrößenverteilung, spezifisches Gewicht, Form der Düngerpartikel, und die Umweltbedingungen, wie Bodenunebenheiten, Geländeneigung, Windstärke und -richtung oder Luftfeuchte, bestimmt. Selbst bei optimalen Umgebungsbedingungen und Geräteeinstellungen bleiben immer noch die Fehler, die beim nötigen Überlappen durch Anschlussfahrten oder beim Grenzstreuen gemacht werden.

Durch den Einsatz von Agrarelektronik bei Mineraldüngerstreuern wird der Nutzer in der Anwendung unterstützt. Elektronische Ausbringregelungen ermöglichen eine fahrgeschwindigkeitsabhängige Dosierung der Düngermenge sowie das einfache Einstellen unterschiedlicher Ausbringmengen nach einmaliger Kalibrierung. Wiegeeinrichtungen in den

Düngerstreuern machen die Ermittlung der ausgebrachten Gesamtdüngermenge möglich. Weiterhin ermöglicht damit das Wiegen während der Ausbringung zum einen eine Regelung anstatt einer reinen Steuerung bei der Düngermengendosierung zum anderen die Dokumentation des Ist-Zustands (d.h. *as-applied maps*).

Einfache Plus-/Minustasten an der zugehörigen Mensch-Maschine-Schnittstelle sind üblich, so dass der Nutzer den eingestellten Sollwert um einen vordefinierten Erhöhungs- oder Verringerungswert anpassen kann. Über Teilbreitenschaltungen lassen sich beim Pneumatik-Auslegerstreuer einzelne Teilbreiten schalten.

Schließlich ermöglichen elektronische Schnittstellen die Integration in landtechnische Prozesssteuerungen (vgl. Kap. 2.3.2.6) und somit den Schlüssel zur Dokumentation der Bewirtschaftungsmaßnahme und der teilflächenspezifischen Mineraldüngerausbringung.

Die bisherigen Ausführungen haben die Applikationstechnik für die möglichst einheitliche und gleichmäßige Ausbringung von mineralischem festem Dünger beschrieben, die nur vom Nutzer eventuell an die vorliegende Situation angepasst wird. Im Rahmen der teilflächenspezifischen Bewirtschaftung ist jedoch das Ziel, die Düngermenge auf dem Feld lokal variiert auszubringen. Je nach Ansatz wird der variierende Dünger-Sollwert von einem anderen Entscheidungsprozess zur Verfügung gestellt. Entweder ist dies ein Nutzer bzw. Experte, eine Applikationskarte beim Kartierungsansatz, ein Online-Sensor beim Sensor-Ansatz oder eine Kombination der beiden letztgenannten beim Sensor-Ansatz mit Kartenüberlagerung. Zusätzlich zu den oben beschriebenen Kriterien der Streuqualität erfordert eine teilflächenspezifisch taugliche Applikationstechnik (*VRT* (*Variable-Rate Technology*)) ein geeignetes dynamisches Verhalten bei der Sollwertanpassung in Fahrtrichtung und die Einhaltung einer mindestens befriedigenden Streuqualität bei sich ändernden Sollmengen.

PEDERSEN ET AL. (2001) [PJS01] haben für 3 Zentrifugaldüngerstreuer und einen Auslegerstreuer den Variationskoeffizienten für die Querverteilung bei sechs unterschiedlichen Sollwerten von 40 kg/ha bis 400 kg/ha gemessen. Bei den Zentrifugaldüngerstreuern lag der Variationskoeffizient zwischen 4.7 % und 13,4 %, während er beim Ausleger-Pneumatikstreuer bei 5,5 % bis 7 % lag. Für die drei Schleuderstreuer wurde auch der Variationskoeffizient für die Änderung der Düngermenge in Fahrtrichtung bestimmt. Der Pneumatikstreuer blieb aufgrund von Gesamtsystemproblemen bei der Auswertung außen vor. Zwei der Schleuderstreuer weisen 14 % auf, der dritte 42 %. Speziell bei zwei dieser Streuer wurde auch eine starke Abhängigkeit von der Fahrrichtung berichtet, was auf Probleme bei der Echtzeitfähigkeit der übergelagerten Applikationssoftware (d.h.

Ortung und Zuordnung zur Applikationskarte) bei Sollwertänderungen hinweist.

FULTON ET AL. (2005) [FSHHS05] untersuchten ebenfalls die Streubildgüte für zwei Wurfstreuer (A und B) und für zwei Pneumatik-Auslegerstreuer (C und D) mit einer ähnlichen Versuchsanstellung. Eine um eine Person geänderte Arbeitsgruppe zielte im zweiten Teil der Untersuchung auf die Analyse des dynamischen Verhaltens ab [FSHDS05]. Entscheidend für die Realisierung eines Prozesssteuerungssystem zur variablen Mineraldüngerausbringung ist demnach das Antwortverhalten der Aktorik auf eine (sprungförmige) Sollwertänderung. Dabei wurden positive und negative Sollwertsprünge im Bereich von 56 kg/ha bis 336 kg/ha vorgegeben. Die gefahrenen Geschwindigkeiten reichten von 9,4 km/h (C) bis 20,4 km/h (A), die Arbeitsbreiten von 12,2 m (C) bis 21,3 m (D). Mit den durchgeführten Untersuchungen konnte gezeigt werden, dass das Antwortverhalten auf eine sprungförmige Sollwertänderung durch eine Sigmoid-Funktion (*sigmoidal function*), deren Graph einen s-förmigen Verlauf aufweist, angenähert werden kann. Damit wurden stets zwei das Antwortverhalten charakterisierende Parameter berechnet. Dies ist zum einen die Übergangszeit (*transition time)* und zum anderen eine Verzögerungszeit (*delay time*). Dabei gibt die Übergangszeit größten Teils die Dynamik der Aktorik wider, während auf die Verzögerungszeit vor allem auch Berechnungsvorgänge der übergelagerten Regelungs- und Steuerungssoftware (Kartierung) eine Auswirkung haben. Aus den gefundenen Ergebnissen ist aus dem Blickwinkel dieser Arbeit (vgl. Kap. 4.1) besonders die minimal erreichbare Übertragungszeit von Interesse. So weist der Auslegerstreuer C weitgehend einheitliche Übergangszeiten für die Sollwerterhöhung als auch -verringerung auf. Für den Sollwertsprung von 112 kg/ha auf 336 kg/ha wurden mit 0,4 s und für den Sprung von 336 kg/ha auf 112 kg/ha mit 0,3 s sehr schnelle Anpassungen auf den neuen Düngersollwert erreicht. Bei den anderen Düngerstreuern liegen diese Werte um mindestens einen Faktor 10 höher. Erwähnenswert bei den Verzögerungszeiten sind die negativen Werte bei Schleuderstreuer B. Dies bedeutet, dass die Sollwertänderung bereits vor dem Erreichen der neuen Teilfläche eingeleitet wird. Bei den anderen Düngerstreuern lassen die Ergebnisse darauf schließen, dass der Vorgang eher zu spät eingeleitet wird. Diese Problematik ist weniger der eigentlichen Düngerstreuer-Aktorik als mehr der übergeordneten teilflächenspezifischen Prozesssteuerung zuzuordnen. Denn zwischen der Bestimmung des neuen Sollwertes, beim Kartierungsansatz abhängig von der Position und einer Applikationskarte, beim Sensor-Ansatz durch die vom Online-Sensor interpretierte (reflektionsoptische) Messung, und dem Auftreffen des Düngers am Boden liegt ein bestimmter Zeitversatz, der durch die Vorwärtsbewegung der Traktor-Geräte-Kombination direkt in einem Positionsversatz resultiert. In der praktischen

Anwendung wird über die Einführung eines „Vorhalte-Faktors“ (*look ahead*) zumindest eine teilweise Kompensation angestrebt. Hierzu untersuchten GRIEPENTROG UND PERRSSON (2001) [GP01] anhand eines von ihnen definierten Modells diesen Positionsversatz in Abhängigkeit von Fahrgeschwindigkeit und Arbeitsbreite für zwei Zentrifugalstreuer und einen Pneumatik-Auslegerstreuer. Der Positionsversatz wurde für fünf Fahrgeschwindigkeitsstufen (1 m/s, 2 m/s, 3 m/s, 4 m/s, 5 m/s) bestimmt. Während bei dem Pneumatikstreuer mit einem negativen Positionsversatz von bis zu 22 m bei 5 m/s die Ausbringmenge zu spät innerhalb der Teilfläche geändert würde, treffen die Düngerkörner bei beiden Schleuderstreuer fast immer zu früh auf dem Boden auf, was sich in positiven Positionsversätzen von bis zu 15 m bei 1 m/s ausdrückt. Jeder der drei Düngerstreuer erreicht nur bei einer definierten Geschwindigkeitsstufe einen Positionsversatz von 0 m. Die Autoren schließen aus den Ergebnissen, dass ein konstanter „*look ahead*“-Kompensationsfaktor in der Regelung als erster Schritt unbedingt erforderlich ist, um den Anforderungen der teilflächenspezifischen Düngung zu genügen. Um der Abhängigkeit des Positionsversatzes von Fahrgeschwindigkeit, Arbeitsbreite, Düngertyp und Streuerbauart genügend Rechnung zu tragen, empfehlen sie jedoch die Nutzung eines dynamischen Faktors.

Die aufgezeigte Problematik und Charakteristika des dynamischen Verhaltens der Applikationstechnik ist bei dem Prinzip des Schleuderstreuers noch mehr zu beachten als beim Pneumatik-Auslegerstreuer, dessen beide Ausleger nahezu seine gesamte Arbeitsbreite ergeben, auf der er in einem Arbeitsgang die gewünschte Applikationsmenge ausbringt. Beim Schleuderstreuer hingegen ergibt sich durch das Erzielen der endgültigen Applikationsmenge erst über ein zweistufiges Arbeitsverfahren, d.h. durch Überlappen, eine weitere Fehlermöglichkeit bzw. Komplexität. PEDERSEN ET AL. (2001) [PJS01] wiesen auf diesen Sachverhalt hin und nannten ein Verbesserungspotential von 5 % - 29 % bei der Güte der Querverteilung zweier Schleuderstreuer, falls der Positionsversatz optimal berücksichtigt worden wäre.

2.3.2.5 Farm Management Informationssysteme und Geo-Informationssysteme

Laut BERG (1993) [Ber93] ist der Betriebserfolg in einem entscheidenden Maß nicht nur durch die Betriebsgröße sowie die natürlichen Standortverhältnisse sondern auch durch die Qualität der Betriebsführung bestimmt. Weiterhin beschreibt er die Betriebsführung (*farm management)* als einen Informationsverarbeitungsprozess. Dieser Informationsverarbeitungsprozess ist gekennzeichnet als kontinuierlicher Prozess von Entscheidungsfindungen im Rahmen der verschiedenen Funktionsbereiche des

Unternehmens, der Beschaffung, der Produktion, des Absatzes, der Finanzierung und dem Personalwesen. Eine Aufteilung zu treffender Entscheidungen in die Schritte Problemerkennung durch Beobachtung, Analyse und Planung, Entscheidung, Durchführung der Entscheidung und der Kontrolle ist möglich. Durch Nutzung des Ergebnisses des Kontrollschritts als Eingangsinformation für die Problemerkennung lässt sich der gesamte Entscheidungsablauf als dynamischer Regelkreis auffassen. Somit beinhaltet die Betriebsführung in entscheidendem Ausmaß die systematische Gewinnung und Verarbeitung von Information [Ber93]. Die Fortschritte der Informationstechnologie in den letzten Jahrzehnten unterstützen den Landwirt bei der Bewältigung dieser Aufgaben und schaffen die Voraussetzungen zum Aufbau einer rechnergestützten Betriebsführung. Ein zentrales Element dafür ist ein für die landwirtschaftliche Betriebsführung geeignetes Informationssystem (*information system*).

BALZERT (2000) [Bal00] definiert aus Sicht der Informatik ein Informationssystem als eine Einheit eines organisatorischen Systems zusammen mit sonstigen technischen Einrichtungen, wobei Mitarbeiter in ihrer Rolle als Aufgabenträger einschließlich Anwender und Benutzer ein organisatorisches System bilden. Somit besteht ein Informationssystem aus Menschen und Maschinen, die Informationen erzeugen und/oder benutzen und durch Kommunikationsbeziehungen miteinander verbunden sind. Diese doch sehr allgemein gehaltene Definition schlüsselt SINDIR (2006) [Sin06] nach vier Haupttypen von Informationssystemen für die landwirtschaftliche Betriebsführung auf:

- *Transaction processing systems,*
- *Farm automation systems,*
- *Management information systems (MIS),*
- *Decision support systems (DSS).*

Farm Management Information System (FMIS): Bei der weiterführenden Definition eines FMIS orientiert sich SINDIR (2006) [Sin06] an der Klassifizierung von unterschiedlichen Entwicklungsstufen eines FMIS nach LEWIS (1998) [Lew98]. Im Rahmen dieser Arbeit wird eine derart differenzierte Sicht nicht weiterverfolgt, sondern die Definition eines Farm Management Informationssystems nach ISOBUS-Definition [ISO11783-1] herangezogen:

„*Farm management information system (FMIS) is an office computer system used by a farmer or contractor that includes the software for farm management such as book keeping, payroll, resource management for machines, products, workers, field management, geographical*

information system, decision support systems and task management" [23].

Der Markt bietet diesen weiten Leistungsumfang sowohl als voll integrierte Lösung als auch als Einzelanwendung an, wie beispielsweise eine „elektronische" Schlagkartei für den Ackerbaubetrieb (mit Komponenten für Planung, Dokumentation des gesamten ackerbaulichen Betriebsprozesses, Lagerverwaltung, Pacht- und Flächenverwaltung, Dünge- und Pflanzenschutzplanung, mobile Datenerfassung und Dokumentation).

Ein FMIS mit seinen übergeordneten Aufgaben Planung und Prognose, Überwachung und Auswertung steht in der Mitte zwischen den Ebenen der mobilen Prozesssteuerung/-technik und der überbetrieblichen Ebene wie Administration, Forschung, Dienstleister, Beratung und Abnehmer [Aue91]. Unabdingbare Voraussetzung für den Aufbau und das Funktionieren eines derartigen Gesamtsystems ist der Informationsaustausch sowohl innerhalb der einzelnen Ebenen als auch zwischen den Ebenen. Kommunikation und deren technische Realisierung ist damit von herausragender Bedeutung [Aue91], [MS01].

Während die bisher aufgezeigten Zusammenhänge und der Stand der Technik bei den FMIS unabhängig von der Bewirtschaftungsweise aufgezeigt wurden, soll im Anschluss auf die Besonderheiten bzw. Erfordernisse der teilflächenspezifischen Bewirtschaftung eingegangen werden.

Gerade der Präzise Ackerbau erfordert die Integration eines georeferenzierten, sehr heterogenen Datenbestandes, der mit unterschiedlichster räumlicher, zeitlicher und thematischer Ausprägung vorliegt. Das Informationsmanagement dieser Art von Daten ist eine ureigenste Aufgabenstellung der Fachdisziplin der Geo-Informationssysteme (GIS). BILL (1999) [Bil99] definiert ein GIS als ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden. Für die grundlegende Systemarchitektur eines GIS und der Vielzahl an integrierten Algorithmen sei auf [Bil99] und [Bar95] verwiesen.

Hinsichtlich des Integrationsgrades der Geo-Informationssysteme bzw. deren Funktionalität in die FMIS unterscheidet KORDUAN (2004) [Kor04] die folgenden Gruppen:

- GIS mit Präzisen Ackerbau-Erweiterungen,
- Herstellerspezifische GIS,
- Erweiterte GIS für Schlagkarteien,

[23] A. a. O., S. 3

- Modellier- und Applikationssoftware,
- GIS zur Felddatenerfassung.

Aufgrund der Wichtigkeit von GIS-Funktionalität in einem FMIS bietet der Markt eine Vielzahl der oben aufgeführten Produkte, die einer stetigen Weiterentwicklung unterliegen. Ein Auszug in der Praxis eingesetzter Produkte wird in [Kor04] und [LSL02] gegeben. Der verstärkte Praxiseinsatz von FMIS mit Eignung zur Integration der teilflächenspezifischen Daten seit mehr als einem Jahrzehnt führt zu einem rasanten Anwachsen des sehr heterogenen Datenbestandes. Um diesen heterogen Datenbestand über einen längeren Zeitraum effektiv nutzen und auch eine Austauschbarkeit (*interoperability*) zwischen verschieden Informationssystemen unterschiedlicher Interessenten und Nutzer zu gewährleisten, schlägt KORDUAN (2003) [Kor03] die Datenbeschreibung mittels standardisierter Metadaten vor. Für das deutsche Forschungsverbundprojekt „*Preagro*“ hat er ein entsprechendes Metadaten-Informationssystem basierend auf der CSDGM-Standardisierung modelliert und implementiert.

Eine Grundlage jedes FMIS ist die Erfassung der Arbeits- und Prozessdaten. Auf die damit verbundenen Fragestellungen nach Eigentum, Nutzung und Schutz dieser Daten sowie den Methoden zur Datenerfassung gibt das DLG-Merkblatt 338 „Datenerfassung - Notwendigkeit und Chance“ [PJ07] eine Antwort für die aktuelle Praxis. Es wird zwischen einer rein manuellen, einer halbautomatischen und einer vollautomatischen Datenerfassung unterschieden. AUERNHAMMER (2006) [Aue06] stuft die automatische Prozessdatenerfassung als eine der Hauptsäulen des Präzisen Ackerbaus ein und zeigt den Stand der Technik bei Ertrags- und Qualitätsermittlung auf den selbstfahrenden Erntemaschinen sowie die Ansätze zur voll automatisierten Prozessdatenerfassung für die Traktor-Geräte-Kombination auf. Mit den rasant steigenden Möglichkeiten der weltweiten Vernetzung mittels des Internets tritt bei der Datenhaltung und Datenweiterverarbeitung neben der innerbetrieblichen Lösung auf dem Betriebsrechner verstärkt die betriebsexterne Lösung zu Tage. Vorschläge für Web-basierte Lösungen im Zusammenhang mit der automatisierten Prozessdatenerfassung und -dokumentation finden sich bei [RA04], [SRA06], [NKB07]. Einen Überblick über die allgemeine Nutzung des Internets und der damit verbundenen Web-basierten Lösungen für die landwirtschaftliche Betriebsführung haben FERENTINOS ET AL. (2006) [FAS06] zusammengestellt. Gerade im Zusammenhang mit den Web-basierten Lösungen rückt die flexible drahtlose M2M-Kommunikation immer mehr in das Interesse der Wissenschaft und Industrie [EOBKR10]. Stellvertretend hierfür werden die Forschungsvorhaben zu einem durchgängigen Datenmanagement für Landmaschinen innerhalb der „*iGreen*“-Infrastruktur

[BBMK11] und zu einer drahtlosen Kommunikation auf dem Feld in Anlehnung an ISO 11783 für die autonome Prozessplanung und -steuerung kooperierender mobiler Arbeitsmaschinen [RRGAHG11] angeführt. Die in engen Zusammenhang damit stehenden neuen Normungsbestrebungen zu NWI ISO 16867 „*Wireless Communication in Agriculture*" [NWIISO16867] wurden bereits im Rahmen der WSN im Kapitel 2.3.2.2 beschrieben.
Den Abschluss dieses Kapitels bildet ein Blick auf den Vorschlag für ein (Farm Management) Informationssystem für den Sensor-Ansatz mit Kartenüberlagerung. LINSEISEN ET AL. (2000) [LSHWSD00] stellten ein Datenflussdiagramm mit Daten, Datenströmen und der benötigten Software für ein Gesamtsystem zur Teilflächenspezifischen Pflanzenproduktion auf. Die Autoren kommen mit Blick auf den Sensor-Ansatz mit Kartenüberlagerung jedoch zu dem Schluss, dass für den Softwareteil der mobilen Prozesstechnik, dieser langfristig in der Lage sein muss, in Echtzeit aktuelle und vergangenheitsbezogene Informationslagen zu verknüpfen, um in einem Arbeitsgang die Sensordaten interpretieren, Daten für Entscheidungsmodelle bereitstellen und die Applikationstechnik dementsprechend steuern zu können. In der von LINSEISEN (2001) [Lin01] entwickelten ersten Version des vorgeschlagen Informationssystems ist diese teilweise Verlagerung der FMIS-Funktionalität in den Mobilteil nicht vollzogen, sondern beschränkt sich nur auf eine stationäre Lösung. Im Mittelpunkt steht dabei die Realisierung einer zentralen Datenablage in Form einer gesamtbetrieblichen Datenbank (mit Erweiterungen zur Erkennung georeferenzierter Daten und) mit den Schnittstellen zu einem GIS, zu der automatischen Prozessdokumentationskomponente und zu einer teilflächenspezifischen Leistungs-Kostenrechnungs-Komponente [Aug01].

2.3.2.6 Systemarchitektur

Zum Aufbau von Prozesssteuerungssystemen für die mobilen Applikationstätigkeiten kommen drei grundsätzlich unterschiedliche Systemarchitekturen zum Einsatz bzw. haben sich historisch entwickelt. AUERNHAMMER (1991) [Aue91] unterscheidet:

- Insellösung (Spezialgerät),
- Mobiler Agrarcomputer (Universalgerät),
- Landtechnisches BUS-System (Vernetztes System).

Die gestellten Anforderungen sind eine zentrale Überwachung, Steuerung und Regelung vom Fahrerplatz aus, die vielfache Nutzung wichtiger Sensoren, eine zentrale Anbindung an den Betriebsrechner und ein zentraler Diagnosezugang.
Bei der Insellösung ist der Steuerungs- und Regelungsrechner und nahezu sämtliche Sensorik und Aktorik direkt auf dem Gerät installiert. Am Fahrerplatz ist nur eine meist einfach

gehaltene (Fern-)Bedieneinheit angebracht. Der Datenaustausch mit dem Betriebsrechner ist nicht vorgesehen. Diese Speziallösung ist optimal auf die zu erfüllende Aufgabe zugeschnitten und führt für das vorgesehene Einsatzziel zum Optimum. Die ersten Spritz- und Düngecomputer zählen zu dieser Klasse. Nachteilig an dieser Lösung ist jedoch, dass diese Speziallösung nicht für andere Geräte bzw. Applikationstechniken des Betriebes genutzt werden kann und aufgrund der geringen Nutzungsdauer teuer ist. Der fehlende Datenaustausch mit dem Betriebsrechner schränkt die Eignung für den Präzisionsackerbau sehr ein.

Mit der Einführung der Systemarchitekturvariante Mobiler Agarcomputer bot die landtechnische Industrie dem Landwirt eine Lösung zur Mehrfachnutzung der eingesetzten Elektronik. Beim Mobilen Agrarcomputer wird ein universeller Steuerungs- und Regelungscomputer am Fahrerplatz installiert. Die nötige Sensorik und Aktorik ist dabei auf den Maschinen und Geräten angebracht. Dieser universelle Agrarcomputer ist portabel und kann auf unterschiedlichen Traktoren und Maschinen des Betriebes eingesetzt werden. Auf dieser Einheit sind alle Steuerungs- und Regelungsprogramme für die unterschiedlichen Applikationstätigkeiten gespeichert und ausführbar. Die Erkennung des angeschlossenen Gerätes erfolgt über eine spezifische Belegung eines Steckers. Auch ein Datenaustausch mit dem Betriebsrechner ist über ein portables Speichermedium möglich. Der Mobile Agrarcomputer erreicht hohe Einsatzzeiten je Jahr und ist damit kostengünstiger als eine Betriebsausstattung basierend auf Insellösungen. Um diesen Vorteil nutzen zu können, ist der Betrieb jedoch zur Festlegung auf einen Agrarelektronikhersteller gezwungen. Auch technisch ist diese Systemarchitektur nicht frei von Nachteilen. Das Universalgerät gibt den Rahmen für die Realisierung einer Benutzerschnittstelle vor, was zwangsläufig auf einen Kompromiss hinausläuft. Auch das Vorhalten der gesamten Rechnerleistung und Speicherkapazität auf einem zentralen Universalgerät kann zu Leistungseinschränkungen bei der Realisierung von Steuerungs- und Regelungssystemen sowie Dokumentationslösungen führen.

Eine alternative Systemarchitektur ist der Einsatz eines verteilten bzw. vernetzten Systems, bei dem mehrere elektronische Steuerungs- und Regelungseinheiten bzw. elektronische Systemteilnehmer über ein Kommunikationssystem miteinander verbunden sind und in ihrer Gesamtheit ein Prozesssteuerungssystem bilden. Für das Kommunikationssystem kommt wie in der Automobil- und Prozessleittechnik ein Feldbussystem zum Einsatz.

Diese Systemarchitekturform wird im Bereich der mobilen Agrarsystemtechnik als Landwirtschaftliches BUS-System, im internationalen Sprachgebrauch als „*Mobile*

Agricultural BUS System“, bezeichnet. Ein derartig vernetztes System zeichnet sich dadurch aus, dass es am Fahrerplatz eine zentrale Ein- und Ausgabeeinheit (Benutzerschnittstelle) besitzt und die Geräte mit einer eigenen Steuer- und Regelungsrechnereinheit mit direkt angeschlossener Sensorik und Aktorik versehen sind. Die Geräterechner erzeugen und steuern über die Dienste der zentralen Ein- und Ausgabeeinheit ihre Schnittstelle zum Nutzer. Ein zentraler Systemteilnehmer, vorrangig in der Nähe des Nutzers platziert, bietet auch die Möglichkeit zum Datenaustausch mit dem Betriebsrechner. Gleiches gilt für eine zentrale Diagnoseschnittstelle. Die verteilte Struktur bietet eine sehr gute Erweiterbarkeit und Skalierbarkeit bei der Realisierung von mobilen Prozesssteuerungen, wie auch die Möglichkeit zur Mehrfachnutzung der eingesetzten zentralen Komponenten. Soll diese Lösung nicht nur technisch vorteilhaft gegenüber dem Mobilen Agrarcomputer sein, so ist auch die Bindung an einen Hersteller zu überwinden. Dies kann jedoch nur über eine Standardisierung oder Normung der Kommunikation sowie eines Teils der (logischen) Funktionalität einzelner Systemteilnehmerklassen erreicht werden. Nur dadurch ist gewährleistet, dass der Nutzer weitgehend unabhängig bei der Wahl der gewünschten Systemkomponenten ist.

Auf nationaler Ebene wurde in Deutschland bereits in den achtziger Jahren des letzen Jahrhunderts mit der Normung des Landwirtschaftlichen BUS-Systems (LBS) begonnen [Aue89] und 1997 als DIN 9684 Teil 2-5 [DIN9684] verabschiedet. Die DIN-Normungsgruppe initiierte etwas verzögert auf internationaler Ebene die Normungsbemühungen zu einem „*Mobile Agricultural BUS System“*. Die entsprechende Normung des internationalen Standards ISO 11783 [ISO11783] ist noch nicht in allen Teilen abgeschlossen. Am Markt wurde der Name ISOBUS für ein Landwirtschaftliches BUS-System nach ISO 11783 etabliert. Die Historie der nationalen und internationalen Normungsbemühungen ist bei [AS06] beschrieben. Die Autoren geben auch eine Aufstellung der Gründe, wieso LBS nicht den breiten Markterfolg erringen konnte.

Auf den landwirtschaftlichen Betrieben findet sich somit noch eine heterogene Landschaft hinsichtlich der mobilen Prozesssteuerungssysteme. Gerade im Altbestand oder bei selbstfahrenden Spezialmaschinen finden sich noch Insellösungen und Mobile Agrarcomputer. Vernetzte Systeme, wenn auch nicht unbedingt in der standardisierten Version, sondern mit herstellerspezifischem Busprotokoll, werden immer mehr zum Stand der Technik. Ein Blick auf die maschineninternen Elektroniksysteme bei Traktoren und Selbstfahrern neuerer Bauart für Nordamerika und Europe zeigt, dass vernetzte Systeme bereits Stand der Technik sind. Ein Blick in den Ausstellungskalender der (weltweit größten)

Landtechnikmesse AGRITECHNICA in den Jahren 2009 und 2011 offeriert, dass das Angebot an ISOBUS-Ausstattung und -Komponenten stark wachsend ist.

2.4 Landwirtschaftliche BUS-Systeme

Im vorigen Kapitel wurde als Systemarchitektur für ein Prozessführungssystem die Klasse der landwirtschaftlichen BUS-Systeme überblicksartig eingeführt. In diesem Kapitel sollen die beiden existierenden genormten landwirtschaftlichen BUS-Systeme DIN9684 Teil 2-5 und ISO 11783 mit ihren (technischen) Hauptmerkmalen vorgestellt werden. Das Kapitel schließt mit einem kurzen Ausblick auf zukünftige Trends und Anforderungen.

2.4.1 Landwirtschaftliches BUS-System nach DIN 9684 - LBS

Folgende Zusammenstellung basiert auf dem Inhalt der Norm DIN 9684 und auf den Erläuterungen von [SJ99]. Laut Zieldefinition der Norm liegt dem LBS als Kerngedanke zugrunde, einzelne elektronische Komponenten, die bei Traktoren und Arbeitsgeräten eingesetzt und von verschiedenen Herstellern hergestellt werden können, zu einem Gesamtsystem zu verbinden und eine zuverlässige Datenübertragung zwischen traktor- und geräteseitigen Komponenten sowie die Datenübertragung von und zu dem stationären Betriebsrechner zu gewährleisten. Dazu legt die DIN 9684 für den Anwendungsbereich von Traktor-Geräte-Kombinationen in der Land- und Forstwirtschaft funktionelle, elektrische und mechanische Eigenschaften für ein bitserielles BUS-System fest. Laut Norm ist ein BUS ein System zur Übertragung von Daten zwischen mehreren Einrichtungen (Teilnehmern, Stationen) über ein gemeinsames Leitungsmedium. Dabei gelten für die übertragenen Daten, für den Anschluss der Einrichtungen und für den Austausch der Daten zwischen den Einrichtungen vereinbarte Bedingungen. In der Informations- und Kommunikationstechnik werden die vereinbarten Bedingungen als Protokoll bezeichnet. Diese abstrakte BUS-Definition übersetzen SPECKMANN und JAHNS (1999) [SJ99] für die internationale Diskussion mit Binary Unit System, was wiederum die Topologie eines Netzwerkes zur Datenkommunikation repräsentiert und als Synonym für Netzwerk verstanden werden sollte.
Das Protokoll für die Verbindungen, den Datenaustausch und den Funktionsumfang der Einheiten eines landwirtschaftlichen BUS-System nach DIN 9684 „Schnittstellen zur Signalübertragung“ [DIN9684] wird in den folgenden vier Normteilen definiert:

- DIN 9684 Teil 2: Serieller Daten-BUS,
- DIN 9684 Teil 3: Systemfunktionen, Identifier,
- DIN 9684 Teil 4: Benutzerstation,

- DIN 9684 Teil 5: Datenübertragung zum Management-Informations-System.

Entsprechend dem OSI-Referenzmodell der ISO für offene Kommunikationssysteme [ISOIEC7498-1] definiert Teil 2 vor allem die Bitübertragungsschicht (*Physical layer*) und die Sicherungsschicht (*Data link layer*) sowie einen Teil der Vermittlungsschicht (*Network layer*). Weiterhin werden in Teil 2 Anwendungsbereich, Zielsetzung und grundlegende Definitionen getroffen. LBS setzt auf dem von der Fa. Bosch erfundenen CAN-Protokoll auf. Dieses Protokoll ist in [ISO11898] dargelegt. CAN liegt eine Multi-Master-Architektur mit verteilten Netzknoten zugrunde. Die Datenübertragung erfolgt seriell mit Datentelegrammen (*data frames*), bestehend aus einem *Identifier* und einem Datenfeld. Dies bedeutet, dass bei diesem nachrichtenorientierten Protokoll der Datenaustausch nicht auf der Adressierung des Nachrichtenempfängers, sondern auf Kennzeichnung der übertragenen Information durch eine Nachrichtenkennung, d.h. dem *Identifier* basiert. Bei diesem objektorientierten Ansatz kann jeder Teilnehmer alle Datentelegramme auf dem gemeinsamen Datenübertragungsmedium empfangen und die für ihn relevanten Daten herausfiltern. Weiterhin verfügt CAN über die Möglichkeit zur Priorisierung von Nachrichten, bietet sehr geringe Latenzzeiten durch einen Buszugriff nach CSMA/CA und gewährleistet eine hohe Datenübertragungssicherheit mit sehr kurzer Fehlererholzeit.

Von der reinen objektorientierten Philosophie, dass im Wesentlichen mit jedem Datentelegramm ein Prozessdatum (z.B. Messwert, Sollwert) übertragen wird, wurde bei LBS abgerückt. Nicht einzelne einfache Sensoren oder Aktoren sind die Netzwerkteilnehmer, sondern sogenannte Jobrechner und LBS-Dienste sind die hauptsächlichen Netzwerkteilnehmer. Diese Jobrechner und Dienste repräsentieren die Einheiten eines Landwirtschaftlichen BUS Systems, wie z.B. eines Traktors, eines Gerätes oder einer Benutzerstation (Mensch-Maschine-Schnittstelle) und stellen deren Funktionalität bereit. Dieser Ansatz erweitert die rein nachrichtenorientierte Struktur von CAN auch um einen verbindungsorientierten Anteil. Um diesen Ansatz entsprechend umzusetzen, wurde das CAN-Identifier-Feld, in der gewählten CAN Version V2.A ein 11bit Identifier, in acht Prioritäts- oder Aufgabengruppen aufgeteilt. Teil 3 der Norm [DIN9684] beschreibt diese Identifier-Struktur und die Festlegung der Systemfunktionen ausführlich.

Die erste Aufgabengruppe „LBS-Initialisierung“ definiert den allgemeinen Systemaufbau mit den möglichen Teilnehmerklassen und die Systemverwaltung mit ihren Funktionen wie z.B. einer dynamischen Teilnehmeradressierung, was An- und Abmelden von Teilnehmer im laufenden Betrieb zulässt. Die zweite Aufgaben- bzw. Prioritätsgruppe sind die „Basis-Botschaften“, die jeweils an alle Teilnehmer gesendet werden, also „*broadcast messages”*

sind. Die Basisbotschaften teilen sich wiederum in die beiden Untergruppen Basis-Daten und Prozessdaten auf. Die Basis-Daten stellen grundlegende Information wie Geschwindigkeit, Motordrehzahl, Zapfwellendrehzahl oder Hubwerkstellung dar und werden zyklisch vorzugsweise vom Traktor-Jobrechner allen Teilnehmer zur Verfügung gestellt. Die zweite Untergruppe umfasst die Prozessdaten für Mess-, Steuerungs- und Regelungsaufgaben der Geräte, soweit sie an alle Netzwerkteilnehmer versendet werden sollen. Da es schwierig ist, abzusehen, welche Mess- und Regelungsmöglichkeiten und -anforderungen zukünftig gefordert sind, wurde kein abgeschlossenes Prozessdatenverzeichnis erstellt, sondern ein Ansatz gewählt bei der jeder Prozessgröße eine klassifizierende Beschreibung („*Identifier*“) mitgegeben wird. Dies sind der für die Prozessgröße zuständige Gerätetyp, der logische Typ einer Prozessgröße (z.B. Sollwert oder Istwert) und der physikalische Typ (wie z.B. elektrisch, mechanisch, hydraulisch). Diese Klassifizierung ist noch nicht umfassend und wird in der Norm noch weiter aufgeschlüsselt. Für den groben Überblick bleibt festzuhalten, dass für die 16 möglichen Gerätetypen die Menge der listenspezifischen *Identifier* jeweils über eine Instanz-Wertgruppe-Matrix festgelegt wird. Obwohl die Matrixelemente bereits klassifiziert sind, sind noch viele Matrixelemente nicht „belegt“ und ermöglichen die zukünftige Erweiterung bzw. Anpassung. Für eine tiefer gehende Betrachtung sei auf Teil 3 der Norm [DIN9684] verwiesen.

Die dritte Aufgaben- bzw. Prioritätsgruppe sind die „Gezielten-Botschaften“. Mit dieser Gruppe können die im vorhergehenden Absatz beschriebenen Prozessdaten nicht mehr im „*broadcast”*-Modus sondern gezielt an einen Empfänger verschickt werden, d.h. teilnehmerorientiert. Die vierte und fünfte Aufgaben- bzw. Prioritätsgruppe sind die „LBS-Dienste“, eine Menge von Betriebsmitteln, die im BUS-System allen Teilnehmer zur Verfügung stehen und über diese beiden Gruppen richtungsabhängig definiert sind. Die beiden wichtigen Dienste Benutzerstation (*Virtual Terminal*) und Auftragsbearbeitung (*Task Controller*) werden nachfolgend noch kurz erläutert.

Die sechste Prioritätsgruppe ist für Partnersysteme reserviert, die beiden restlichen Prioritätsgruppen sind nicht genutzt und wurden für zukünftige Erweiterungen vorgesehen.

Der Teil 4 der Norm [DIN9684] standardisiert den bereits angesprochenen LBS-Dienst „Benutzerstation“. Diese Benutzerstation ist die zentrale Schnittstelle zwischen BUS-System und Nutzer, weshalb ihr eine besondere Bedeutung und Aufgabe zukommt und sie einen wichtigen Einfluss auf die Akzeptanz des Systems beim Nutzer hat. Ziel war es, dass der Nutzer mit nur noch einer Benutzerstation herstellerübergreifend alle eingesetzten Anbaugeräte steuern und bedienen kann. Die Aufgaben und der Leistungsumfang werden in

der Norm überwiegend in Form logischer Funktion beschrieben.

Die bisher aufgeführten Normfestlegungen ermöglichen eine Kommunikation zwischen den einzelnen BUS-Teilnehmern, seien es Traktor, Gerät oder Benutzerstation. Weiterhin ist auch die Schnittstelle zwischen dem Nutzer und dem BUS-System definiert. Was noch fehlt, ist eine Möglichkeit zum bidirektionalen Datenaustausch mit dem FMIS, dem stationären Betriebsrechner. Zu diesem Zweck werden in Teil 5 der Norm [DIN9684] die Festlegungen für „Auftragsbearbeitung und Datenübertragung zum Management-Informations-System“ getroffen. Grundsätzlich definiert dieser Normteil zwei Schnittstellen für den bidirektionalem Datentransfer zwischen dem stationärem System und dem mobilen BUS-System. Für die Übertragung und Interpretation der auszutauschenden Daten wurde einerseits eine standardisierte Datentransfer- oder Auftragsdatei definiert, andererseits der LBS-Dienst „Auftragsbearbeitung“. Bewusst wurde auf eine Standardisierung des Datentransfermediums, z.B. Speicherkarte oder eine Funkverbindung, verzichtet, um mit den teilweise rasanten Entwicklungen der modernen Informationstechnik Schritt halten zu können. Der LBS-Dienst „Auftragsbearbeitung“ kann die am FMIS vorbereiteten Auftragsdaten, die in der LBS-Auftragsdatei abgelegt sind, interpretieren und mittels der Prozessdaten in Mess-, Steuerungs- und Regelungsaufgaben für die Anbaugeräte umsetzen. Ebenso kann er die durchgeführten Arbeiten und Messwertsammlungen in der Auftragsdatei dokumentieren. Auf die Bedeutung der Auftragsbearbeitung bzw. des *Task Controllers* im Zusammenhang mit teilflächenspezifischer Bewirtschaftung wird im Kapitel des ISOBUS nochmals gesondert eingegangen.

2.4.2 Landwirtschaftliches BUS-System nach ISO 11783 – ISOBUS

Während der nationalen Standardisierungsarbeit zur DIN 9684 wurde bei der internationalen Normungsorganisation ISO mit der Standardisierung eines Landwirtschaftlichen BUS-Systems als ISO 11783 „*Tractors and machinery for agriculture and forestry -- serial control and communications data network* --“ begonnen. Wie schon bei der DIN 9684 mit der Kurzbezeichnung LBS wurde auch für diese ISO-Norm nach einer schlagkräftigen und marktwirksamen Umschreibung gesucht und in dem Ausdruck ISOBUS gefunden. Laut Definition der grundlegenden Zielsetzung umfasst der Anwendungsbereich des ISOBUS Traktoren und Geräte für den landwirtschaftlichen und forstwirtschaftlichen Einsatz. Für diesen Anwendungsbereich ist ein serielles Datennetzwerk für Regelungs- und Kommunikationsaufgaben zu spezifizieren. Dabei steht die Standardisierung der Methode und des Formats des Datenaustausches zwischen Sensoren, Aktoren, Steuer- und

Regelungseinheiten, von Informationsspeichereinheiten und von Mensch-Maschine-Schnittstellen im Mittelpunkt.

Laut AUERNHAMMER und SPECKMANN (2006) [AS06] wurden Grundgedanken und Funktionalität der DIN 9684 gerade in Bezug auf die landwirtschaftlichen Belange ohne fundamentale Änderungen in die internationale Norm integriert bzw. als Ausgangspunkt für die Ausgestaltung der Norm übernommen. Die beiden Autoren weisen jedoch auch darauf hin, dass die deutlich umfangreichere und komplexere ISO-Norm sich in wesentlichen Definitionen von der DIN 9684 unterscheidet:

- Verwendung von CAN Version V 2.0B mit dem erweiterten 29 bit Identifier,
- Eine Verdoppelung der Übertragungsrate auf 250 kbit/s,
- Eine Strukturierung der Norm in deutlich mehr Einzelteile,
- Anlehnung an das OSI-Referenzmodell der ISO für offene Kommunikationssysteme [ISOIEC7498-1], wann immer möglich,
- Definition der Norm, so dass sie interoperabel mit dem Standard SAE J1939 [SAEJ1939] ist,
- Definieren der Aufgaben der Traktor ECU und Klassifizierung unterschiedlicher Traktor ECU-Klassen,
- Einführung der Möglichkeit zur Definition von herstellerspezifischen (*proprietary*) Nachrichtentypen und –austausch.

Die Philosophie des ISOBUS und die Art und Weise, wie einerseits die Elemente der SAE J1939 und anderseits Festlegungen und Ideen der DIN9684 als die beiden Grundpfeiler in die Norm eingeflossen sind, wird bei [SMFB99] ausführlich dargelegt. Abbildung 2.6 gibt eine typische Traktor-Geräte-Kombination mit ISOBUS-Netzwerk wieder, das untergliedert ist in einen Traktor BUS (*Tractor Bus*), einen Geräte BUS (*Implement Bus*) und einen Geräte-Subnetzwerk BUS (*Implement Subnetwork Bus*). Die Möglichkeit zum Datenaustausch mit dem FMIS über einen sogenannten *Task Controller* oder ein *Management Computer Gateway* ist ebenfalls angedeutet.

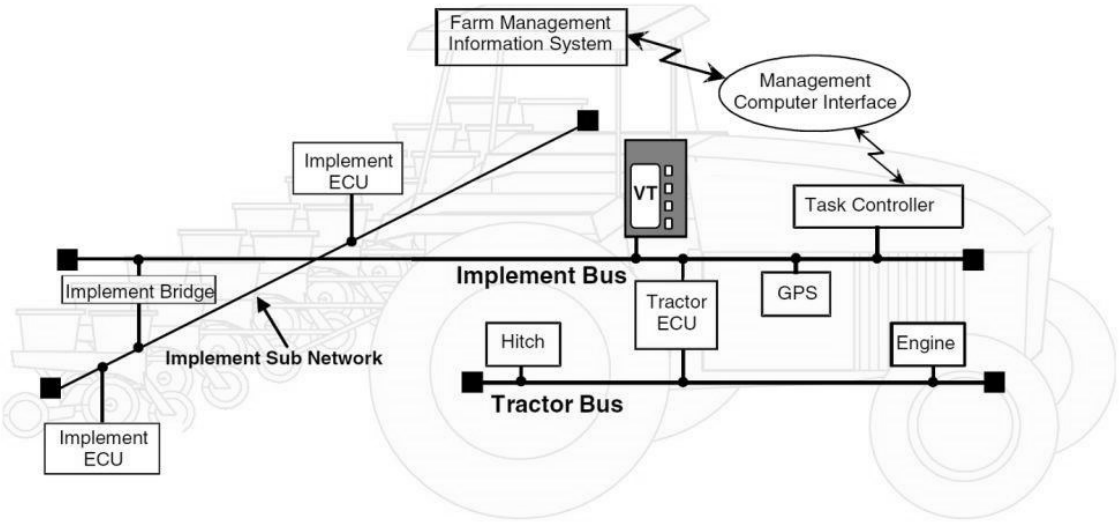


Abbildung 2.6: ISOBUS-Konfiguration mit Traktor und Anbaugerät (nach [SMFB99], [SAEJ1939-02])

Das Ergebnis der nahezu zwei Jahrzehnte dauernden und noch laufenden Normungstätigkeit ist ein umfangreiches Normenwerk mit 14 Teilen. Tabelle 2.6 führt die einzelnen Normteile und ihre Zielsetzung auf. Der für diese Arbeit herangezogene Status ist in der ersten Spalte mit aufgeführt. Ausgangspunkt ist die letzte gültige IS-Version zum August 2011. Nur wenn für einen Teil bis dahin überhaupt kein Internationaler Standard verabschiedet wurde, folgt eine Entwurfsversion. Auf eine detaillierte Erläuterung aller Normteile wird hier aus Gründen des Umfangs verzichtet. Neben dem Studium der einzelnen Normtexte werden dem interessierten Leser auch die Kurzbeschreibungen der ISO 11783 Teile 1-13 von [AS06] empfohlen. Bevor am Ende dieses Kapitels noch auf einige für den Präzisen Ackerbau wichtigen Aspekte der Norm eingegangen wird, folgt ein kurzer Absatz zum Stand der Normungsaktivität, Implementierung und Markeinführung.

Tabelle 2.6: Teile der ISO 11783 und ihre Zielsetzungen (Stand August 2011) [ISO11783]

Norm-Teil und Status	Titel	Ziel
ISO 11783-1:2007(E) (2007-06-15) (IS)	General standard for mobile data communication	Überblick über die gesamte Norm und Beschreibung der Zusammenhänge der einzelnen Normteile
ISO 11783-2:2002(E) (2002-04-15) (IS)	*Physical layer*	Spezifikation der Verkabelung, der Steckverbinder und der physikalischen Signaldarstellung auf dem Datenbus / Datenübertragungsmedium
ISO 11783-3:2007(E) (2007-10-01) (IS)	*Data link layer*	Spezifikation der Datenstruktur der CAN-Nachrichtentelegramme und einer Methode zur Übertragung von Informationen, die größer als ein CAN-Nachrichtentelegramm sind
ISO 11783-4:2001(E) (2001-05-01) (IS)	*Network layer*	Spezifikation der Vernetzung verschiedener Sub-Netzwerke
ISO 11783-5:2011(E) (2011-04-01) (IS)	*Network management*	Spezifikation von Methoden zur Netzwerkinitialisierung sowie zur eindeutigen Benennung und Identifizierung der Netzwerkteilnehmer
ISO 11783-6:2010(E) (2010-10-15) (IS)	*Virtual terminal*	Spezifikation einer Teilnehmereinheit als universelle Mensch-Maschine-Schnittstelle
ISO 11783-7:2009(E) (2009-05-01) (IS)	*Implement messages application layer*	Definition von Nachrichten zum Datenaustausch mit und zwischen Traktor und Geräten
ISO 11783-8:2006(E) (2006-02-01) (IS)	*Power train messages*	Definition von Nachrichten zur traktor- bzw. antriebsstranginternen Kommunikation
ISO 11783-9:2002(E) (2002-07-01) (IS)	*Tractor ECU*	Spezifikation des Traktorrechners, der zugleich eine *Gateway*-Funktion zwischen Traktor- und Geräte-Bus einnimmt
ISO 11783-10:2009(E) (2009-12-15) (IS)	*Task controller and management information system data interchange*	Spezifikation der Kommunikation des *Task Controllers* (Auftragsbearbeitung) mit den elektronischen Busteilnehmern (ECU) wie auch des Datenaustausches zwischen dem *Task Controller* und dem FMIS
ISO 11783-11:2011(E) (2011-07-01) (IS)	*Mobile data element dictionary*	Datenbank (online auf www.isobus.net) zur Definition der Datenelemente, die im Teil 10 ausgetauscht werden.
ISO/DIS 11783-12 (2006) (ISO/DIS)	*Diagnostics services*	Definition der Diagnosefunktionalität des BUS-Systems
ISO 11783-13:2011(E) (2011-04-01) (IS)	*File Server*	Definition einer von allen BUS-Teilnehmern nutzbaren Datenspeichereinheit und der zugehörigen Funktionen
ISO/DIS 11783-14 (2008) (ISO/DIS)	*Sequence control*	Definition automatisiert ablaufender Funktionssequenzen bei Traktor-Geräte-Kombinationen

Hierzu erklären HIERONYMUS UND HENNIGER (2009) [HH09], dass die Definition der Normenprozesse weitestgehend abgeschlossen und die Produktprogramme vieler Hersteller bereits auf ISO 11783 eingestimmt sind. Die beiden Autoren weisen jedoch auch auf die Herausforderungen bei der Implementierung und der praktischen Anwendung hin. Der ISOBUS ist als offenes System konzipiert, dessen Voraussetzung eine uneingeschränkte

system- und herstellerunabhängige Kompatibilität ist. In der aktuellen Praxis zeigt sich, dass eine uneingeschränkte Kompatibilität der ISOBUS-Produkte unterschiedlicher Hersteller nicht immer gegeben ist. Dies kann einerseits durch unterschiedliche Interpretation der komplexen Normteile bedingt sein, zum anderen existieren mittlerweile unterschiedliche Ausbaustufen und Varianten des ISOBUS seit der Markteinführung zur AGRITECHNICA 2003. Als eine Lösung für diese Situation wurde die Umsetzung der ISOBUS-Norm in einer Software-Bibliothek als „Freie Software“ (*Open Source)* vorgeschlagen und als „ISOAgLib“ implementiert [SAD01]. Diesem Ansatz liegt die Grundidee zugrunde, dass wenn diese Form der Software von allen ISOBUS-Entwicklern eingesetzt wird, dann sind die damit erstellten Systeme immer kompatibel, weil sie letztendlich auf die gleiche, einheitliche Software aufbauen. Gleiches gilt für Erweiterungen und Ergänzungen. Von der ISOBUS-Gemeinschaft wurde jedoch ein anderer Weg eingeschlagen. HIERONYMUS und HENNIGER (2009) [HH09] beschreiben diesen Weg, der auf Zertifizierungstests in offiziell akkreditierten Laboren (schon bei LBS realisiert), regelmäßigen „*Plug-fest*“-Treffen, d.h. entwicklungsorientierte Komponenten- und Schnittstellentests der verschiedenen Hersteller, sowie der Spezifikation von „ISOBUS *Implementation Level* (IL)“ basiert. Aus applikationsorientierter Sicht wird bei der Definition eines ISOBUS-IL eine Teilmenge aus den bereits vorhandenen wie auch neuen Normteilen mit dem Ziel spezifiziert, eine einheitliche applikationsorientierte Implementierung in einem festgelegten Zeitrahmen zu erreichen. Bei der Definition dieser IL’s wird auf Rückwärtskompatibilität geachtet, so dass auch ältere ISOBUS-Produkte mit denen neuer Bauart auf der Basis des „kleinsten gemeinsamen Nenners“ kommunizieren können, auch wenn dadurch das volle Funktionsspektrum der neuen Produkte nicht voll abgerufen werden kann.

Zum Abschluss dieses Kapitels soll auf den für die teilflächenspezifische Prozessführung wichtigen Teil 10 der Norm [ISO11783] mit den Festlegungen für Auftragsbearbeitung und den Datenaustausch mit dem FMIS eingegangen werden. In der nachfolgenden Erläuterung wird der Stand ISO 11783-10:2009(E) [ISO11783-10] herangezogen. Laut der Zielsetzung dieses Normteils wird darin die Auftragsbearbeitung, im Englischen das „*Task Management*“, für den ISOBUS spezifiziert. Laut ISOBUS-Nomenklatur werden dafür zwei unterschiedliche Systemeinheiten unterschieden. Einerseits das mobile Traktor-Geräte-Gespann zur Prozessführung, abgekürzt mit MICS, und das stationäre betriebliche Informationssystem, das FMIS. Der Stand der Technik im Bereich FMIS wurde in Kapitel 2.3.2.5 bereits erläutert. Die zentrale Schnittstelle auf dem MICS zum Datentransfer mit dem FMIS als auch zur Auftragsausführung durch die Mittel bzw. Entitäten des MICS ist der *Task Controller*, in etwa

vergleichbar dem LBS-Dienst Auftragsbearbeitung. Im Rahmen des „*Task Management*“ wird eigens auf die beiden Säulen des Präzisen Ackerbaus, die Dokumentation mit der automatischen (Prozess)Datenerfassung und die Teilflächenbewirtschaftung, eingegangen.

Die automatische Prozessdatenerfassung, in der Norm mit dem Schlüsselwort „*Data logging*“ ([ISO11783-10] Kapitel „6.6.2 Data logging“) bezeichnet, wird folgendermaßen umschrieben. Der *Task Controller* kann zu diesem Zweck Prozessdaten-Variablen-Werte über das ISOBUS-Netzwerk anfragen, die gelieferten Werte sammeln und zum weiteren Transfer ans FMIS aufbereiten. Die Dokumentation erfolgt in der Regel mit Positions- und Zeitreferenz. Die Auslösebedingung bzw. das Aufzeichnungsverhalten ist über sogenannte „*Data log triggers*“ spezifiziert. Diese Triggerbedingungen können sowohl zeit- oder wegbasiert, ereignisgesteuert (z.B. Über- oder Unterschreiten von Werteober-/untergrenzen) als auch Integralwertbasiert („*Totals”*) sein. In Kombination mit den geplanten Auftragsdaten lassen sich somit die geplanten und kommandierten Sollwerte zusammen mit den Istwerten („*as applied*“) einer Applikationstätigkeit dokumentieren. Sind im ISOBUS-System integrierte (Online)Sensoren einer Geräteklasse zuzuordnen oder sind sie eine Funktion eines ISOBUS-Jobrechners bzw. *Working-Sets*, so können damit Messwerte als Ausgangsmaterial für Kartierungen aufgezeichnet werden. Der ISOBUS bietet daher gute Voraussetzungen für die automatische Prozessdatenerfassung.

Die Teilflächenbewirtschaftung wiederum wird im [ISO11783-10] Kapitel „6.6.1 Site-specific application“ spezifiziert. Der ISOBUS-Ansatz sieht dazu Folgendes vor:

Teilfächenspezifische Applikation erfordert, dass der *Task Controller* entsprechend der aktuellen Position die gültigen vorab geplanten Prozessdaten im MICS aussendet. Für diese Aufgabenstellung werden in den Auftragsdaten die entsprechenden geometrischen, positionsrelevanten Definitionen getroffen. ISOBUS kennt dafür sowohl das Rasterformat („*gridcell*“) als auch für unregelmäßige Formen das Vektorformat („*polygon*“). „*Gridcell*“ oder „*polygon*“ nehmen Bezug auf eine Teilfläche, die homogen behandelt werden soll, die „*TreatmentZone*“. Bewegt sich ein „*DeviceElement*“ in eine neue „*TreatmentZone*“, dann soll der *Task Controller* den neuen Applikationssollwert, der laut Auftragsdaten der „*TreatmentZone*“ zugeordnet ist, an den für die Applikation zuständigen *Working-set master* kommandieren.

Aus diesen Festlegungen ist zu schließen, dass der Kartierungsansatz zur Prozessführung gut unterstützt wird und bisher im Mittelpunkt der ISOBUS-Normungsbemühungen hinsichtlich der Teilflächenbewirtschaftung stand. OSTERMEIER ET AL. (2003) [OAD03] haben die fehlende Unterstützung für den reinen Sensor-Ansatz und erst recht für den Sensor-Ansatz mit

Kartenüberlagerung sowohl für LBS als auch für ISOBUS frühzeitig aufgezeigt. Nachdem mittlerweile Normteile wie das „*Virtual Terminal*“ oder die „*Automated Sequences*“ als größtenteils abgeschlossen betrachtet werden, gewinnt die Definition des Teils 10 an Bedeutung. Auch das wachsende Angebot an Online-Sensorik zur N-Düngung nach dem Sensor-Ansatz, wie z.B. Yara N-Sensor und -ALS, Greenseeker, CropSpec, ISARIA oder Crop Circle [Rec10], haben das Interesse an Integration des Sensor-Ansatzes bei der weiteren Normierung des Teil 10 geweckt. Das Normungsgremium hat dementsprechend auf den Bedarf nach einer umfassenderen Definition für teilflächenspezifische Applikationstätigkeiten mittlerweile mit einem Ergänzungsvorschlag, dem „ISO Draft Amendment ISO 11783-10:2009 DAM1 (Date: 2010-09-29)“ [ISO11783-10DAM1], reagiert. In dem Kapitel „6.6.3 Real time sensor based control“ wird ein Vorschlag für den Sensor-Ansatz vorgestellt und auch eine Lösung für den Sensor-Ansatz mit Kartenüberlagerung gegeben. Dabei wird eine Fusion rein auf Entscheidungsebene vorgeschlagen, d.h. die Fusion von Applikationssollwerten, die von externen Quellen, z.B. einem Online-Sensor, oder von Nutzereingaben oder einer Applikationskarte stammen können. Somit ergibt sich für die Hersteller und Nutzer von Online-Sensoren eine neue Implementierungsperspektive. Bisher lösten die am Markt befindlichen Online-Sensorsysteme die Problematik sehr pragmatisch und wendeten z.B. bereits „fertige“ Normteile wie das *Virtual Terminal* nach ISOBUS-Definition an und implementieren zur Ansteuerung der Applikationstechnik und für den Datenaustausch mit dem FMIS (für Dokumentation) eine herstellerspezifische (*proprietary*) Lösung, teilweise mit eigener Hardware, vergleichbar einem Mobilen Agrarcomputer, oder mittels der herstellerspezifischen ISOBUS-Nachrichtendefinitionen.

2.4.3 Ausblick

Wie bereits im Kapitel 2.3.2.6 angedeutet, gab es während der nationalen und internationalen Normungsbemühungen auch firmenspezifische landwirtschaftliche BUS-System-Lösungen, die teilweise nur die Netzwerkstruktur und das CAN-Bus-Protokoll mit den Normen gemeinsam hatten oder bereits Teile der Normen verwendeten oder in abgeänderter Form anwandten. Da wie von HIERONYMUS und HENNIGER (2009) [HH09] aufgezeigt, die Produktprogramme vieler Hersteller, auch aller „*Global Player*“ bereits auf ISO 11783 eingestimmt sind und sowohl auf Normungs-, Entwicklungs-, Konformitätsprüfungs-Seite aber auch im Bereich Marketing, Vertrieb und Service umfangreiche Bemühungen unternommen werden, ist davon auszugehen, dass der ISOBUS eine Art De-facto-Standard für die Landtechnik in der Gegenwart und näheren Zukunft sein wird. Dabei gehen

AUERNHAMMER und SPECKMANN (2006) [AS06] davon aus, dass die Hersteller für rein herstellerspezifische Belange, sei es zur Pflege historischer Lösungen oder um eigenständige fortschrittliche Systemlösungen mit einem Wettbewerbsvorteil zu implementieren, wahrscheinlich auf die Nutzung der im ISOBUS auch möglichen proprietären Nachrichtentypen zurückgreifen werden.

Es deuten sich aber bereits auch neue Anforderungen für tiefgreifende zukünftige Erweiterungen des ISOBUS bzw. neuer landwirtschaftlicher BUS-Systeme an. So weist der ISOBUS strukturbedingt durch Verwendung des CAN-Protokolls kein absolut deterministisches Übertragungsverhalten auf. Lösungen, wie sie auch für die Automobiltechnik vorgeschlagen bzw. erprobt werden, wurden bereits in [RFS07], [SLMPW05] diskutiert. EHRL UND AUERNHAMMER (2006) [EA06] sprechen auch den Aspekt der Datensicherheit an und schlagen Verschlüsselungsstrategien als Abhilfe vor. Beim Blick auf die Übertragungsrate offenbart sich eine weitere Schwachstelle. Die 250 kbit/s sind für Anwendung mit verteilten Bildverarbeitungssystemen höchstwahrscheinlich zu gering bzw. schränken die Möglichkeiten stark ein. Auch ein Blick auf die vorgestellten Lösungen zur Elektrifizierung von Traktor-Geräte-Kombinationen, lässt erahnen, dass ein BUS mit höherer Bandbreite und Deterministik nötig ist [PPT10]. Die Autoren empfehlen hierfür eine Real-time Ethernet-Lösung. In der industriellen Prozessautomatisierung wird für vergleichbare Applikationen bereits auf derartige Industrial Real-time Ethernet-Systeme vertraut [Ahl08].

Von Vorteil bei der Integration dieser neuen Anforderungen könnte sein, dass bei der ISOBUS Definition, wann immer möglich, das OSI-Referenzmodell der ISO-Norm für offene Kommunikationssysteme [ISOIEC7498-1] als Modell angewandt wurde. Theoretisch sollten somit einzelne Kommunikationsschichten (*Layer*) durch eine andere Technik ersetzt werden können, solange dies transparent für die unter- und übergelagerte Schicht bleibt, d.h. die entsprechenden Schnittstellendefinitionen sich nicht ändern.

2.5 Entwicklungsprozesse und Vorgehensmodelle

In den Traktoren und landwirtschaftlichen Maschinen mit ihren elektronischen, vernetzten Kommunikationssystemen werden Steuerungen, Regelungen und Benutzerschnittstellen zu einem großen Teil über Software realisiert [HH07].

Eine wirtschaftliche und qualitätsbewusste Entwicklung dieser elektronischen Systeme in landwirtschaftlichen Arbeitsmaschinen erfordert einen festgelegten organisatorischen Rahmen. Das Fachgebiet der Software-Technik (*Software Engineering*) spezifiziert solch einen Rahmen als Prozessmodell, auch Vorgehensmodell genannt. Darin wird festgelegt,

welche Aktivitäten in welcher Reihenfolge von welchen Personen erledigt werden und welche Ergebnisse dabei entstehen und wie diese in der Qualitätssicherung überprüft werden [Bal00]. BALZERT (1998) [Bal98] listet folgende Vorgehensmodelle für die Software-Technik auf:

- Wasserfall-Modell,
- V-Modell,
- Prototypen-Modell,
- evolutionäres/inkrementelles Modell,
- objektorientiertes Modell,
- Spiralmodell.

Einige dieser Modelle werden in diesem Kapitel noch etwas ausführlicher dargestellt, die geeigneten Modelle für die Entwicklung wissensbasierter Systeme werden im zweiten Teil dieser Arbeit noch vertieft (vgl. Kap. 5.1.3).

Auch wenn der Softwareanteil in den landwirtschaftlichen Traktoren und Maschinen einen immer größeren Anteil einnimmt, darf der Hardwareanteil nicht außer Acht gelassen werden. Entsprechend fordert MARTINUS (2004) [Mar04], dass für die Entwicklung elektrisch/elektronischer/programmierbar elektronischer Systeme (E/E/PES) von mobilen Arbeitsmaschinen eine Trennung der Entwicklungsprozesse für Software- und Hardware-lastige Systemelemente vermieden werden sollte. Diese Sichtweise deckt sich auch mit Vorgehensmodellen zur Realisierung nachrichtentechnischer Systeme, bei der die Entwicklung und Herstellung von Software oft auch eine definierte Hardware erfordert, deren Entwicklung häufig Teil des Gesamtprojektes ist [Tor94]. Als Orientierung für den landtechnischen Bereich verweist MARTINUS (2004) [Mar04] auf die Entwicklungskonzepte der Pkw- und Nutzfahrzeugindustrie. Dabei werden typischerweise die einzelnen Teilsysteme entsprechend einer genauen Schnittstellenvorgabe getrennt voneinander entwickelt und in späteren Schritten zu einem Gesamtsystem integriert. Die Entwicklung der Hardware und Softwareeinheiten erfolgt dabei in der Regel parallel.

2.5.1 Wichtige Vorgehensmodelle

Das ursprüngliche und weitverbreitete Vorgehen lässt sich mit dem Wasserfall-Modell der Software-Technik sehr gut beschreiben, erweitert um parallele Prozessschritte zur Hardwareentwicklung und mit nachgeschalteten Integrationsschritten. Das Wasserfall-Modell ist so benannt, da die Ergebnisse einer Phase bzw. eines Prozessschrittes wie bei einem Wasserfall in die nächste Phase fallen. Die einzelne Schritte bzw. Phasen sind angelehnt an [Mar04], [Tor94]:

- Systemanforderungen,
- Software/Hardware-Anforderungen,
- Analysephase,
- Software/Hardware-Entwurf,
- Codierung/Prototypenbau,
- Software/Hardware-Test,
- Integration von Software und Hardware,
- Fertigung,
- Betrieb und Wartung.

Charakteristisch ist dabei, dass jede Aktivität in der richtigen Reihenfolge und vollständig durchzuführen ist, also ein streng sequentieller Entwicklungsablauf, mit Orientierung am „*top down*“-Vorgehen. Phasenübergreifende Iterationen sind dabei möglichst zu vermeiden.

Ursprünglich entwickelt für die Bundeswehr, etabliert sich mittlerweile jedoch das V-Modell zum Stand der Technik im Bereich der E/E/PES für mobile Arbeitsmaschinen bzw. Agrarsystemtechnik [HH07]. Das V-Modell ist eine Erweiterung des Wasserfall-Modells und integriert die Qualitätssicherung in Form der Verifikation und Validation in den Prozess. Unter Verifikation wird die Überprüfung der Übereinstimmung zwischen einer Software/Hardware/Systemeinheit und seiner Spezifikation verstanden. Unter Validation wird die Eignung bzw. der Wert eines Systems bzw. Produktes bezogen auf seinen Einsatzzweck verstanden [Bal98]. Die Validation und Verifikationsschritte werden den entsprechenden Spezifikations-, Entwurfs- und Implementierungsschritten gegenübergestellt, was dazu führt, dass das entstehende Prozessmodell sich in Form eines V darstellen lässt. Abbildung 2.7 verdeutlicht dies:

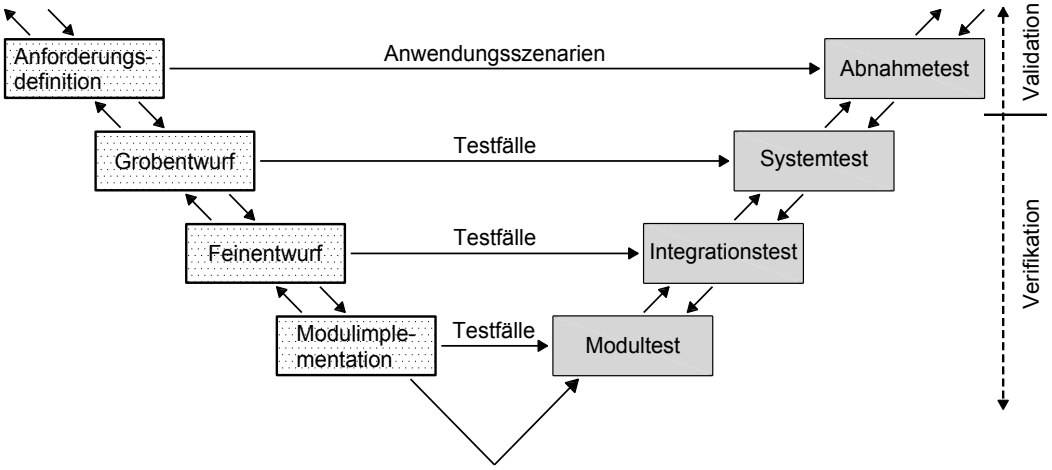


Abbildung 2.7: V-Modell [Bal98]

Gerade diese explizite Integration von Verifikation und Validation in das Prozessmodell spielt eine immer größere Rolle bei der Entwicklung sicherheitsrelevanter Systeme in landwirtschaftlichen Traktoren und Arbeitsmaschinen. MARTINUS (2004) [Mar04] hat die speziellen Anforderungen an die funktionale Sicherheit zukünftiger mechatronischer Systeme von mobilen Arbeitsmaschinen analysiert und ein entsprechendes Entwicklungskonzept aus Vorgehensmodell, Methoden und Werkzeugen erarbeitet. Hierfür greift er auf das V-Modell zurück und zeigt dies beispielhaft an einem Vorgewendemanagement einer Traktor-Geräte-Kombination. HIERONYMUS ET AL. (2008) [HHG08] stellten die Normungsarbeiten der Landmaschinenindustrie für eine eigene Norm zur funktionalen Sicherheit vor. Im Jahr 2010 wurde die entsprechende ISO 25119 - Part 1-4, *Tractors and machinery for agriculture and forestry - Safety-related parts of control systems,* [ISO25119] verabschiedet.

Einen dem evolutionären/inkrementellen Modell folgenden Ansatz „*Agile Software Development*“ fordern LENZ UND MÜNCH (2005) [LM05], um die Adaption und Integration von (neuen) Kundenanforderungen zu beschleunigen. Bei dem Wasserfall- und V-Modell kann es dazu kommen, dass das Produkt alle Anforderungen fehlerfrei erfüllt, ausgiebig getestet ist, aber beim ersten Praxiseinsatz die Kundenwünsche und -erwartungen verfehlt bzw. enttäuscht. Dies liegt normalerweise daran, dass in der Anforderungsphase die Kundenwünsche nicht ausreichend bekannt waren oder nicht klar formuliert werden konnten. Zum anderen entwickeln sich Anforderungen und neue Ideen zur Verbesserung und Erweiterung erst bei den ersten Einsätzen bei den Anwendern. Jedoch liegt bis zur Auslieferung des Produkts bzw. von neuen Merkmalen an den Kunden bedingt durch Entwicklungsphasen und Produktlebensdauerzyklen oftmals ein längerer Zeitraum. Um genau diese Innovationsfähigkeit und Agilität, das schnelle Reagieren und Integrieren von neuen

(inkrementellen) Produktmerkmalen, zu verbessern bzw. zu erreichen, schlagen genannte Autoren die Anwendung der Agilen Softwareentwicklung vor. Jedoch unterscheiden sie zwischen zwei Produktkategorien bzw. Reifeklassen von Anforderungen. Ist das Produkt mit seinen Anforderungen bestens bekannt und gereift, so sollte mit dem Fokus auf Fehlerfreiheit oder -vermeidung bei den bewährten Wasserfall- oder V-Modellen geblieben werden. Stehen neue Produktfelder, Innovationen und schnelle Reaktion auf Erfahrungen und Verbesserungswünsche des Kunden im Mittelpunkt, so empfiehlt sich das evolutionäre/inkrementelle Modell.

Die aufgezeigten Modelle fokusieren sich jedoch auf Produkte für eine Serienfertigung. Hingegen ermöglicht der begrenzte Umfang an Ressourcen im Rahmen eines Forschungsprojektes nicht die vollständige und umgreifende Realisierung eines Systems bzw. zielt auch nicht auf industrielle Serienfertigung ab. Jedoch sollen Wirkprinzipien untersucht und ein entsprechender „*proof of concept*“ der Kernfunktionalität gezeigt werden. Für einen derartigen Ansatz eignet sich insbesondere das Prototypen-Modell. Nach BALZERT (1998) [Bal98] unterstützt dieses Modell auf systematische Weise die frühzeitige Erstellung ablauffähiger Modelle (Prototypen) des zukünftigen Produkts, um die Umsetzung von Anforderungen und Entwürfen in Software zu demonstrieren und damit zu experimentieren. Im Englischen wird dies auch als *Prototyping* umschrieben. Zwischen einem Prototyp und fertigen Systemen können unterschiedliche Beziehungen bestehen. Die Prototypen sind Teil der Produkt/Anforderungsdefinition. Oder die Prototypen werden inkrementell bis zur Marktreife weiterentwickelt. Oder die Prototypen werden implementiert, um Ideen zu überprüfen, Probleme zu untersuchen bzw. zu klären oder neue Erkenntnisse zu gewinnen. Ohne das Ziel ein marktfähiges Produkt zu entwickeln, ist dies meist der Ansatz, der an Universitäten oder Forschungsinstituten zu finden ist.

2.5.2 Integration der MSDF

Neben dieser Einschränkung muss im Hinblick auf das Untersuchungsziel auf die speziellen Belange bei der Implementierung von MSDF-Systemen bzw. -Funktionen eingegangen werden.

In der Literatur wird dazu grundsätzlich festgestellt, dass es im eigentlichen Sinne kein reines MSDF-System gibt, sondern dass es eine Teilfunktionalität eines größeren Systems ist, das zur Erfüllung seiner Dienste und Aufgabenstellungen MSDF-Funktionalität einsetzt bzw. erfordert [HM04], [WH01]. Folglich ist die Anforderung nach einer derartigen Funktionalität ein Ergebnis der Anforderungsanalyse für ein übergelagertes System, wie. z.B. für ein

landwirtschaftliches Prozessführungssystem. Für WALTZ UND HALL (2001) [WH01] ist nahezu jedes *Data Fusion* System eine eigenständige Aufgabenstellung der Systemtechnik (*System Engineering*), aber einige Fragestellungen spielen fast immer eine Rolle. Dies sind die Anforderungsanalyse (*requirement analysis*), die Sensorauswahl (*sensor selection*), die Architekturauswahl (*architecture selection*), die Algorithmusauswahl (*algorithm selection*), die Software-Implementierung (*software implementation*), der Test (*test*) und die Evaluierung (*evaluation*). Eine Aufgabenbeschreibung und eine Einordnung der Systemtechnik im Rahmen des Projektmanagements finden sich bei [Mad00]. Nach MADAUSS (2000) [Mad00] löst die Systemtechnik Aufgaben prinzipiell durch eine globale Betrachtungsweise, indem eine Sache aus einer alles überblickenden Vogelperspektive betrachtet wird, um dann weiter ins Detail vorzustoßen. In der einschlägigen Literatur wird dieser Ansatz als „*top down approach“* benannt.

Dieser systemtechnische Ansatz mit den typischen Aktivitäten der Anforderungsdefinition, des Subsystem-Designs und Design-Synthese wurde von WALTZ UND LLINAS (1990) [WL90] überblicksartig dargestellt und lässt sich folgendermaßen zusammenfassen. Die Anforderungen der Gesamtanwendung werden in funktionale Leistungsanforderungen auf der Systemebene konvertiert. Dies erfordert oftmals die Dekomposition des Systems in funktionale Anforderungen und Zuordnung zu Subsystemelementen (z.B. Sensor-Subsysteme, Kommunikations-Subsysteme, Datenverarbeitungs-Subsysteme, Datenbank-Subsysteme, MMI-Subsysteme und Applikationssteuerungs-Subsystem).

Die primäre Aufgabe des Subsystemdesigns ist es, die funktionalen Anforderungen an Sensor, Datenverarbeitung (*Processing*), Kommunikation, MMI und der Applikationssteuerung zu spezifizieren, die durch das physikalische Systemdesign erfüllt werden müssen. Das Hauptaugenmerk liegt dabei auf der Auswahl einer zentralen, verteilten oder hybriden Architektur und der Definition der räumlichen und zeitlichen (Auflösungs-)Fähigkeiten der individuellen Sensoren.

Die anschließende Phase ist die Implementierung oder Design-Synthese, in der die Anforderungen realen Hardware- und Software-Komponenten zugeordnet werden. Dieser Segmentierungsprozess teilt weiterhin funktionale Anforderungen zwischen Hardware und Software auf, was wiederum neue Schnittstellendefinitionen zwischen diesen beiden nach sich zieht. Abschließend sind die Hardware und Software-Designs für Sensorkommunikation, Verarbeitung, MMI, Applikationssteuerung zu implementieren. Entsprechende Hardware- und Software-Integrationsschritte auf Subsystemebene als auch auf übergeordneten Systemebenen schließen die Implementierungsphase ab.

Dieses Vorgehen ist identisch mit dem bereits vorgestellten Wasserfall-Modell, falls auch die Verifikations- und Validationsaktivitäten nach Abschluss der Implementierungsphase durchgeführt werden. Jedoch lässt sich auch das V-Modell zur Anwendung bringen, falls die Test- und Evaluierungsschritte bereits parallel zu den entsprechenden Ebenen der Definitions-, Design- und Implementierungsphasen ausgeführt werden.

Dieser systemtechnische Ansatz zur Implementierung eines (Multisensor) *Data Fusion*-System wird auch von BOWMAN UND STEINBERG (2001) [BS01] unterstützt. Jedoch weisen WALTZ UND HALL (2001) [WH01] auch auf Nachteile bzw. Einschränkungen und Voraussetzungen dieses Vorgehens hin. So ist dieser Ansatz vor allem für umfangreiche und komplexe Hardware- und Softwaresysteme geeignet, wie sie bei großen Unternehmen, Behörden und dem Militär vorkommen. Für eine erfolgreiche Anwendung bzw. zur Rechtfertigung der aufzubringenden Aktivitäten und Mittel sind jedoch auch einige Voraussetzungen zu beachten. Das System muss groß und komplex genug sein, um nicht mit weniger formalen Mitteln entwickelt werden zu können. Die Anforderungen müssen relativ stabil sein bzw. bleiben und die grundlegenden Techniken sollten über die Entwicklungsdauer sich nur langsam ändern bzw. weiterentwickeln. Weiterhin muss ein großer Teil des Systems von Grund auf selbst entwickelt werden, da es nicht als Standardprodukt auf dem Markt erworben werden kann.

Diesen Zusammenhängen stehen jedoch die rasanten Fortschritte in der modernen Informationstechnologie gegenüber, welche einen alternativen Ansatz oder auch ein teilweise anderes Vorgehen ermöglichen, bzw. erfordern, um mit den raschen Entwicklungen Schritt halten zu können. Dieser „*Enterprise Architecture Approach",* auch „*System Architecting*" umschrieben, wurde in [WH01] beschrieben. Demzufolge ermöglicht die verfügbare Informationstechnologie die Realisierung von komplexen Computernetzwerken, die zusammen mit einer großen Anzahl von Menschen (Nutzern) Unmengen an Daten analysieren und verarbeiten können und zwar in einer Umgebung, die mit „*enterprise*" umschrieben wird. Für die Entwicklung einer entsprechenden „*enterprise architecture*" sind die funktionalen Anforderungen (*functional operations*) zu betrachten und diese Funktionen auf ein Netzwerk von Personen (kognitiv), auf Hardware (physikalisch) oder Software-Komponenten aufzuteilen. Für eine tiefer gehende Betrachtung dieses Vorgehensmodells sei auf die oben angegebene Literaturstelle verwiesen, jedoch soll die unterschiedliche Sichtweise nochmals aufgezeigt werden. Während beim systemtechnischen Ansatz die eine (Einzelfall)spezifische Lösung gesucht und implementiert wird, wird beim „*Enterprise Architecture Approach"* versucht mit bereits bestehenden bzw. am Markt erhältlichen Lösungen- und/oder

Standardkomponenten zu beginnen und zu sehen, inwieweit die Anforderungen („*use cases"*) damit bereits erfüllt werden können bzw. ob die „*use cases*" sich entsprechend modifizieren lassen. WALTZ UND HALL (2001) [WH01] weisen abschließend jedoch auch darauf hin, dass sich beide Ansätze nicht komplett ausschließen, sondern komplementär sind und somit Systementwickler mit zunehmender Verfügbarkeit von Standardkomponenten („*COTS tools"*) eher zu einer hybriden Vorgehensweise tendieren werden. HALL UND MCMULLEN (2004) [HM04] haben eine Übersicht von am Markt befindlichen MSDF-Software-Werkzeugen und -Programmpaketen zusammengestellt. Sie verweisen jedoch darauf, dass trotz des wachsenden Angebotes an COTS-Software für einzelne *Data Fusion*-Fragestellungen noch kein umfassendes MSDF-Software-Paket erhältlich ist.

2.6 Schlussfolgerung

Die Zielsetzung dieser Arbeit weist einen stark multidisziplinären Charakter auf und erfordert, Wissen verschiedener Fachdisziplinen zur Anwendung zu bringen. Daher wurde in dem Kapitel zum Stand des Wissens ein großer Bogen von der Theorie der *Multisensor Data Fusion*, den Grundlagen des Präzisen Ackerbaus, den Prinzipien der Regelungs- und Steuerungstechnik, dem Stand der Technik bei Landwirtschaftlichen BUS-Systemen bis hin zu ausgewählten Aspekten der Systemtechnik gespannt. Aus dem Blickwinkel des Ergebnisteils zum zweiten spezifischen Teilziel (vgl. Kap. 5.2.2.1) relativiert sich der große Umfang der Ausführungen, da die Unterkapitel zum Präzisen Ackerbau auch als Ergebnisbeschreibung interpretierbar sind. Auch wenn die Agrarsystemtechnik bisher nicht zu den treibenden Kräften bei der *Multisensor Data Fusion*-Disziplin gehört hat, ist ihr das Gebiet nicht fremd und vor allem *Sensor Fusion*-Lösungen aus Teilsystemen von Serienprodukten nicht mehr wegzudenken. Bisher lag der Fokus dabei größtenteils auf dem Teilsystem und der Auswahl eines genau darauf abgestimmten Fusionsalgorithmus. Eine allgemeine Analyse- und Entwurfsmethode wurde für die Agrarsystemtechnik dabei aber nicht dokumentiert bzw. vorgeschlagen, geschweige denn auf Landwirtschaftliche BUS-Systeme abgestimmt. Daher soll diese Arbeit an dieser Stelle zur wissenschaftlichen Diskussion und Aufarbeitung beitragen, um die bestehende Lücke für die MSDF in der Agrarsystemtechnik zu schließen.

3 MSDF-Framework und Zielsetzung

Die bisherigen Ausführungen zeigen, dass ein durchgängiger Spezifikations- und Entwicklungsprozess eine herausragende Rolle bei der erfolgreichen Umsetzung von Anforderungen in eine physikalische Implementierung spielt, wenn technische Systeme effizient und zielorientiert implementiert werden sollen. Dazu ist ein entsprechendes Vorgehensmodell für mobile landtechnische BUS-Systeme erforderlich, wenn diese den Einsatz eines MSDF-Systems erfordern bzw. davon profitieren würden. Daraus leitet sich die allgemeine Zielsetzung ab und diese soll am aussagekräftigen Prozessführungsbeispiel Sensor-Ansatz mit Kartenüberlagerung als eine Art Obermenge für Ansätze in mobilen Applikationssystemen in einem spezifischen Teilziel gezeigt werden.
Dazu gilt es, durch ein entsprechendes Vorgehensmodell, im Weiteren auch MSDF-Framework genannt, das Verständnis und die Kommunikation unter Theoretikern, Entwicklern, Testern und Nutzern von derartigen Systemen zu erleichtern. Anderseits soll es helfen, ein geeignetes und angepasstes Problemlösungs-Paradigma auszuwählen, das dem zugrundeliegenden Problem adäquat entspricht. Für die Umsetzung in ein reales System/Produkt gilt es den aktuellen Stand der Technik bei der mobilen landtechnischen Prozessführungstechnik zu beachten, jedoch auch das nötige Erweiterungspotential für zukünftige Lösungen mitzubringen bzw. aufzuzeigen.

3.1 MSDF-Framework für landwirtschaftliche BUS-Systeme

Im Stand des Wissens wurde deutlich, dass sich ein weitgehend systemtechnischer Entwicklungsprozess nach dem Wasserfallmodell oder V-Modell, wenn funktionale Sicherheit eine Rolle spielt, als Stand der Technik für mobile agrartechnische Systeme etabliert hat. Wird in der Gesamtsystemanalysephase die Forderung nach einer MSDF-Lösung oder der Einsatz von Techniken aus diesem Fachgebiet für Teilsysteme abgeleitet, wird für die weiteren Phasen das folgende MSDF-Framework vorgeschlagen.

- Grundsätzlich sollte ein durchgängiges MSDF-Framework unterschiedliche Abstraktionsebenen besitzen und eine „*top down*"-Dekomposition der Anforderungen, sowie einen anschließenden strukturierten Systementwurf erlauben.
- Ein funktionales Modell sollte auf der höchsten Abstraktionsebene beschreiben und festlegen, welche Analysefunktionalität oder -prozesse durchgeführt werden müssen. Demgegenüber beschreibt ein prozedurales bzw. Prozessmodell auf hoher Abstraktionsebene, wie diese Analysen geleistet werden können.

- Auf der Basis dieser abstrakten Sichtweise von Anforderungen, Spezifikationen und Problemlösungs-Paradigma bzw. -Paradigmen gilt es eine Systemarchitektur (Abstraktion der Hard- und Software-Implementierung) zu entwerfen.
- Bei der anschließenden Transformation dieser Systemarchitektur in eine konkrete technische Realisierung in Hard- und Software müssen etablierte und geeignete Systemtechnik (*System Engineering*)-Methoden Anwendung finden.
- Falls die weitere Verbreitung von MSDF-Technik in der Landtechnik es möglich macht, können dann auch Methoden des „*System Architecting*“ für einen hybriden Ansatz mit einbezogen werden.

In den vorausgehenden Abschnitten wurde ausführlich dargelegt, dass es im Bereich der funktionalen und prozeduralen Modelle sowie der Systemarchitekturen bereits Lösungen gibt. Es ist daher keine komplette Neuerfindung nötig, sondern es kann auf Elemente zurückgriffen werden, die sich in anderen Anwendungsbereichen bereits bewährt haben. Jedoch gilt es die Eignung für die Agrarsystemtechnik zu beachten und, falls nötig, dementsprechend anzupassen.

Auf funktionaler Ebene wird für die Anwendung des „*Revised JDL data fusion model*“ nach STEINBERG und BOWMAN (2001) [SB01] (vgl. Kap. 2.1.4.1) vorgeschlagen. Zwar ist das *„JDL data fusion model“* das am weitesten verbreitete und etablierte funktionale Modell, jedoch ist es zu stark auf den militärischen Einsatzbereich fokussiert. Im Gegensatz dazu ist das „*Revised JDL data fusion model*“ auch für zivile Anwendungsbereiche geeignet und hat eine größere Abdeckung der möglichen (*Multisensor*) *Data Fusion*-Funktionen und -Anwendungsbereiche als das „*Functional Model*“ nach DASARATHY (1994) [Das94]. Sollte ein „*human in the loop*“-Entscheidungsfinder im System von Nöten sein, so lässt sich das empfohlene Modell um ein „*Level 5 Processing (cognitive refinement)*“ nach BLASCH UND BLANO (2002) [BP02] erweitern.

Auf der Ebene des Prozessmodells wird eine Anwendung des „*Process Model“* nach ANTONY (1995) [Ant95] (vgl. Kap. 2.1.5.4) vorgeschlagen. Bereits im Vorwort zu seinem Lehrbuch zu „*Principles of Data Fusion Automation*“ verweist ANTONY (1995) [Ant95] darauf, dass im Idealfall ein *Data Fusion*-System mit einem „*top down*”- und anforderungsgetriebenen Ansatz zu entwickeln sei. Durch das Fehlen einer übergreifenden Theorie zur Algorithmen-Auswahl, ist die Auswahl der technischen Lösung weitgehend durch die individuelle Erfahrung des Designers bestimmt oder der Expertise und Historie einer Firma oder Forschungseinrichtung geschuldet. Auch wenn das prozedurale Modell nach Antony nicht als formaler

informationstheoretischer Ansatz bewertet werden kann, ist es nach Wissen und Einschätzung des Autors dieser Arbeit der umfassendste und strukturierteste Ansatz zur Ableitung und Auswahl eines geeigneten Problemlösungsparadigma bzw. *Data Fusion*-Algorithmus. Die grundsätzlichen Ideen der beiden Prozessmodelle „*Boyd's Decision Loop*" (vgl. Kap. 2.1.5.2) und „*Omnibus Process Model*" (vgl. Kap. 2.1.5.3) finden sich auch im vorgeschlagen prozeduralen Modell wieder. Der Weg über die Taxonomie von Algorithmen nach HALL UND MCMULLEN (2004) [HM04] (vgl. Kap. 2.1.5.1) bleibt immer noch für den Fall, dass mit dem prozeduralen Modell nach Antony kein geeigneter Ansatz bestimmt werden kann, oder wenn für einzelne Algorithmus-Typen weitere Hintergrundinformation eingeholt werden sollen.

Für die Ableitung und den Entwurf einer Systemarchitektur gilt es zuerst auf die für MSDF-Systeme bekannten und bewährten Systemarchitekturvarianten zurückzugreifen. Die Anpassung an die agrarsystemtechnischen Anforderungen erfolgt über die Beachtung der Rahmenbedingung, dass Landwirtschaftliche BUS-Systeme vorrangig in ihrer standardisierten Form nach ISO 11783 oder DIN 9684 Stand der Technik sind, wobei in der Praxis und der Forschung nur die internationale Norm von Bedeutung ist und daher als Rahmenbedingung für die Systemarchitektur vorgeschlagen wird. Dennoch gilt es im Rahmen dieser Arbeit auch den Aspekt der Skalierbarkeit und zukünftigen Erweiterbarkeit der abzuleitenden Systemarchitektur nicht zu vernachlässigen.

Für die anschließenden Phasen des Feinentwurfes und der Implementierung wird auf die Anwendung von dem Stand der Technik entsprechenden bzw. im Unternehmen vorgegebenen Systemtechnik-Methoden, Richtlinien und Qualitätssicherungsmaßnahmen verwiesen.

3.2 Zielsetzung

Als Antwort auf die allgemeine übergeordnete Zielsetzung wurde im vorausgegehenden Unterkapitel das **MSDF-Framework** für landwirtschaftliche BUS-Systeme, vorangig für den ISOBUS, vorgeschlagen. Wie in der Problemstellung dargelegt, stehen auf **MSDF basierende Prozessführungen in landwirtschaftlichen BUS-Systemen** im Mittelpunkt des Interesses in dieser Arbeit. Um die Anwendbarkeit des MSDF-Frameworks für dieses Anwendungsgebiet zu überprüfen und das Verständnis für die praktische Umsetzung zu fördern, wird zweistufig vorgegangen:

Dazu wird das MSDF-Framework im ersten spezifischen Teilziel auf ein **konkretes landwirtschaftliches Verfahren** angewandt werden. Hierzu bietet sich eine ***Real-time* Prozessführung in sensorgestützten Düngesystemen nach dem Sensor-Ansatz mit**

Kartenüberlagerung (*Real-time approach with map overlay*) für intensive Stickstoffdüngung an. Gerade dieser Ansatz erscheint besonderes geeignet, die Breite des Anforderungs- und Lösungsspektrums abzudecken, da er sowohl Online-Sensortechnik als auch Kartenmaterial und somit unterschiedliche Informationsquellen und Wissensarten zur Prozessführung vereint. Im Grunde stellt dieser Ansatz die Obermenge aller Prozessführungen für teilflächenspezifische Applikationssysteme dar und ermöglicht somit durch gezieltes Ein- und Ausblenden von Systemkomponenten bzw. -aspekten die Überprüfung auf Skalierbarkeit der vorgeschlagenen Lösung. Hinsichtlich der landwirtschaftlichen BUS-Systeme soll dabei auch betrachtet werden, ob die vorliegenden Normen bereits ausreichende Definitionen bereitstellen oder ob Erweiterungen anzuraten sind, die für Systeme der Zukunft bereits von Beginn an Bestandteil sein sollten.

Im zweiten spezifischen Teilziel soll dann die abgeleitete Lösung für den Sensor-Ansatz mit Kartenüberlagerung für die intensive Stickstoffdüngung als **Software-Simulation mit intuitiver Interaktionsmöglichkeit** aufgebaut und getestet werden. Dabei ist der Fokus auf die MSDF realisierenden Systemelemente sowie auf eine angepasste Entwicklungsmethodik zu legen. Aufgrund des nötigen interdisziplinären Fachwissens und des Bedarfs an Feldtestdaten soll hierfür auf die Arbeiten der IKB-Dürnast Forschungsgruppe zurückgegriffen werden.

4 MSDF ISOBUS-Lösung für die (kleinräumige) N-Düngung

In diesem Kapitel wird die vorgeschlagene Analyse- und Entwurfsmethode für eine auf *Multisensor Data Fusion* basierende Prozessführung in einem landwirtschaftlichen BUS-System auf den Sensor-Ansatz mit Kartenüberlagerung für intensive Stickstoffdüngung angewandt. Die Realisierung einer Real-time Prozessführung in sensorgestützten Applikationssystemen ist prinzipiell eine Aufgabenstellung der Steuerungs- und Regelungstechnik. Abbildung 4.1 stellt den Gesamtzusammenhang bildlich dar.

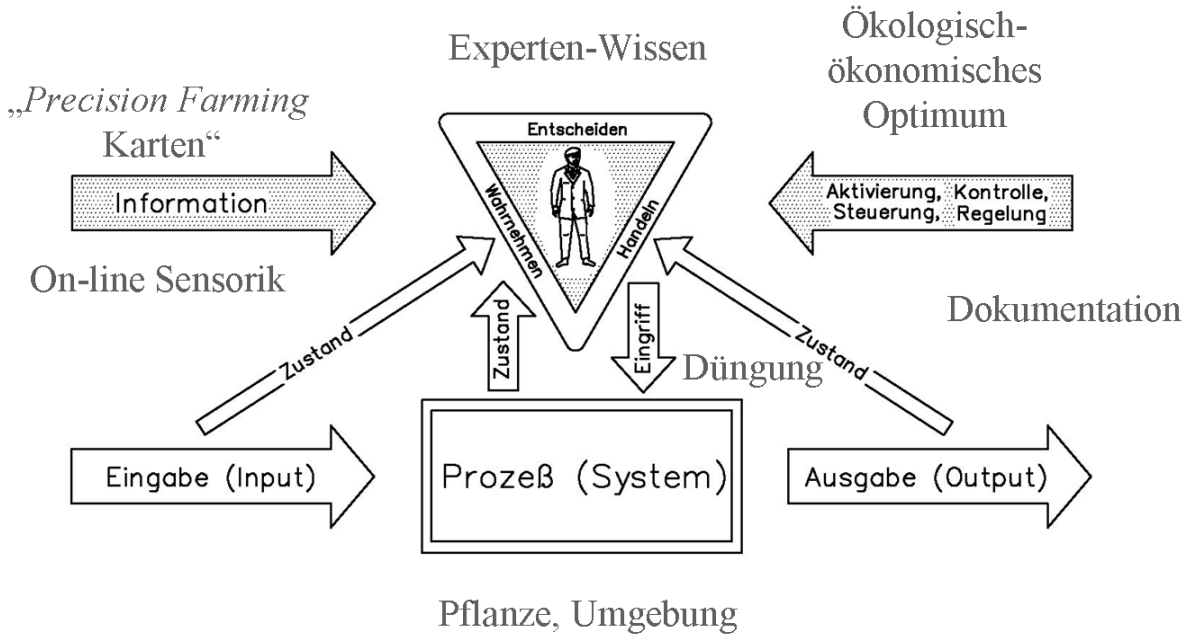


Abbildung 4.1: Real-time Prozessführung in sensorgestützten Applikationssystemen aus der Sichtweise der Steuerungs- und Regelungstechnik (abgeändert nach Vorlesungsunterlagen AUERNHAMMER)

Es ist notwendig einen Prozess oder ein System, hier Pflanzen und ihre nähere Umgebung, zu einem ökologischen und ökonomischen Optimum zu führen. Dies erfordert Information über den aktuellen Zustand des Prozesses und seine Eingangsgrößen (*Input*), d.h. *Precision Farming-Karten* und Online-Sensorik-Prozessdaten. Die Möglichkeit zur Beeinflussung des Prozesses ist die Düngung. Dabei wird der Applikationssollwert anhand von analytischem und/oder heuristischem Wissen und der vorliegenden Information über die Prozess-Eingangsgrößen abgeleitet. Dokumentation vervollständigt das Verfahren. Zusammengefasst bedeutet dies, dass die aktuelle Situation für jede einzelne Teilfläche untersucht und bewertet werden muss, woraufhin eine Aktion (Applikationssollwert) abgeleitet werden muss.

Diese Abstraktion auf eine Situationsbewertung mit abgeleiteter Aktion sowie das Vorliegen

unterschiedlicher Informationsklassen lassen den Schluss zu, dass sich für diese Aufgabenstellung die Anwendung von MSDF-Techniken eignen. Bevor im Folgenden das funktionale und prozedurale Modell sowie die Systemarchitektur entworfen werden, sollen noch einige wichtige Rahmenbedingungen für das übergeordnete Gesamtsystem festgelegt werden.

4.1 Rahmenbedingungen

Als Rahmen („*(in) scope*“) für die zu entwerfende Real-time Prozessführung in sensorgestützten Düngesystemen soll eine für Mitteleuropa typische Traktor-Geräte-Kombination mit einem landwirtschaftlichen BUS-System vorausgesetzt werden. Dabei steht nicht die Grunddüngung mit Kalk, Phosphor, Kalium und Magnesium im Mittelpunkt, sondern die in Europa wichtige (mineralische, erstragsbildende) Stickstoffdüngung. Während eine teilflächenspezifische Grunddüngung in der Regel mit dem Kartierungsansatz realisiert wird, können für die kleinräumige Bestandesführung bei der Stickstoffdüngung alle drei Systemansätze in Betracht gezogen werden. Um der kleinräumigen Bewirtschaftungsweise möglichst optimal gerecht zu werden, wird nicht der weitverbreitete Schleuderstreuer als Applikationstechnik, sondern ein Ausleger-Pneumatikstreuer angenommen. Wie im Kapitel 2.3.2.4 aufgezeigt wurde, weist dieser Typus besseres bzw. einfacheres Verhalten bei der Überlappung auf und zudem lassen sich Teilbreiten gezielt schalten. Hinsichtlich der Echtzeitbedingung stellt er höhere Anforderungen, da er durch fehlende Wurfweite nach hinten etwas weniger Zeit für die Anpassung an neue Sollwerte hat. Nicht zu unterschätzen ist auch, dass der Ausleger-Pneumatikstreuer der Applikationstechnik des Pflanzenschutzes und der Saat zumindest im geometrischen Sinne ähnelt. Dies erleichtert den Transfer des zu entwerfenden Systems auf weitere teilflächenspezifische Aufgabenstellungen.

Hinsichtlich der Prozessführungsstrategie liegt eine modellbasierte Optimierung vor. Das anzustrebende Optimum ist in diesem Fall sowohl in ökonomischer als auch ökologischer Richtung vorgegeben und zielt auf die Einstellung eines optimalen Betriebspunktes für jede einzelne Teilfläche ab. Der entsprechende Steuerungs- und Regelungsalgorithmus wird in den folgenden Unterkapiteln abgeleitet und im Folgekapitel wird für einen konkreten Fall die entsprechende Implementierung in Form einer Simulation demonstriert.

Eine wesentliche Anforderung einer Real-time Prozessführung ergibt sich aus dem geforderten Echtzeitverhalten. Unter der Voraussetzung eines stets korrekten Applikations-Sollwertes, können, aus dem ökonomischen Blickwinkel betrachtet, bei einer nicht Schritt haltenden Sollwertänderung zusätzliche Kosten in Form von Minderertrag oder

Düngermehrverbrauch mit Folgekosten durch Lagerfrucht und/oder Qualitätsverlust entstehen. Diese Kosten tendieren eher zu einem Grenzwert. Dies spricht mehr für *weiche Echtzeitbedingungen* für diese Prozessführung. Aus dem Blickwinkel der Ökologie ist bei den Vorgaben von festen Vorschriften des Gesetzgebers, Kunden oder Verbänden auszugehen. Die Nichteinhaltung von Grenzwerten durch nicht Schritt haltende Sollwertänderung kann daher in letzter Konsequenz zu enormen Kosten in Form von Strafen, Aberkennung einer Zertifizierung, Auftragsstornierung und Vertrauensverlust führen. Dies legt den Schluss nahe, dass *harte Echtzeitbedingungen* einzuhalten sind.

Unabhängig von der später gewählten Systemarchitektur und dem konkreten Einsatzfall soll und kann an dieser Stelle bereits der grobe Rahmen für die einzuhaltende Zeitbedingung bestimmt werden. Der gesamte Prozess mit den Schritten Eingangsdatenerfassung, Datenübertragung, Datenverarbeitung zur Entscheidungsfindung und Änderung der Ausbringrate an der Aktorik (pneumatischer Auslegerstreuer) muss in einer endlichen Zeit erfolgen. Somit ist auch der für die MSDF und Entscheidungsfindung zur Verfügung stehende Zeitraum begrenzt. Die gesamte verfügbare Zeit für den oben beschriebenen Gesamtprozess kann über Gleichung 4.1 berechnet werden:

$$t_T = \frac{s}{v} \tag{4.1}$$

mit:

t_T = Zeitdauer für Gesamtprozess

s = Distanz zwischen dem Sensorblickfeld und dem Bereich, wo der Dünger auf dem Boden auftrifft

v = Fahrgeschwindigkeit der Traktor-Düngerstreuer-Kombination

Angenommen das Sensorblickfeld liegt vor dem Traktor, dann ist eine Distanz s von 6,5 m durchaus realistisch. Wird weiterhin eine Fahrgeschwindigkeit von 4,0 m/s vorausgesetzt, so berechnet sich laut Gleichung 4.2 eine Zeitdauer für den Gesamtprozess von 1,6 s:

$$t_T = \frac{s}{v} = \frac{6.5\,m}{4.0\,m/s} = 1.6\,s \tag{4.2}$$

Der Zeitanteil für die Signalverarbeitung in der Online-Sensorik wird im vernachlässigbaren ms-Bereich angenommen, die Datenübertragungszeit auf dem ISOBUS wird mit vier Online-Sensorprozessdaten und einem Prozessdatum für den Applikationssollwert zu je 6 ms angesetzt. Für letztgenannten Wert wird auf die Untersuchungen zu Übertragungs- bzw.

Latenzzeiten[24] in einem ISOBUS-System von HOFSTEE UND GOENSE (1999) [HG99] zurückgegriffen. Damit beträgt der addierte Zeitanteil $t_{Sensor\&ISOBUS}$ für die Sensorsignalverarbeitung und die mittlere ISOBUS-Kommunikation etwa 30 ms. Weitaus mehr fällt der Zeitbedarf für die Durchführung der Sollwertänderung am Düngerstreuer ins Gewicht. Ausschlaggebend hierfür ist die Übergangszeit (*transition time)* der Aktorik *(*Auslegerstreuer), da die Verzögerungszeit (*delay time*) bereits in dem Teilprozess der Datenverarbeitung zur Entscheidungsfindung enthalten ist bzw. dort berücksichtigt wird.
Die von FULTON ET AL. (2005) [FSHDS05] an einem Serienprodukt ermittelten minimal möglichen Übergangszeiten für die Sollwerterhöhung als auch -verringerung in der Größenordnung von 0,4 s und 0,3 s werden etwas abgeschwächt angewandt. Nach einer in der Vergangenheit liegenden Rücksprache mit einem ehemaligen Kollegen am Fachgebiet Technik im Pflanzenbau der TU München[25] wird ein etwas höherer Wert von 0,65 s angenommen. Werden diese Übergangszeit und der Zeitanteil $t_{Sensor\&ISOBUS}$ addiert, so ergibt sich ein Zeitabschnitt von 0,68 s, der von der Zeitdauer des Gesamtprozesses subtrahiert die maximal verfügbare Zeitdauer Δt für den Teilprozess MSDF und Entscheidungsfindung nach Gleichung 4.3 ergibt:

$$\Delta t = 1.6\ s - 0.68\ s = 0.92\ s \tag{4.3}$$

4.2 Funktionales Modell

Aus dem funktionalen Blickwinkel lässt sich der Sensor-Ansatz mit Kartenüberlagerung für intensive Stickstoffdüngung entsprechend dem „*revised JDL data fusion model*“ folgendermaßen beschreiben und analysieren.

4.2.1 Informationsquellen

Mögliche Informationsquellen wurden bereits im Kapitel 2.3.2 vorgestellt. Hervorzuheben sind im Bereich der lokalen Informationsquellen im mobilen Prozessführungssystem

[24] A. a. O., S. 383 „…. *The simulations showed that the transfer time of messages across the network remains less than 6 ms (maximum observed value) for a typical agricultural machinery configuration. Increase in load on the implement bus to more than 80 % of full capacity can result in transfer times of about 70 ms when the priorities are not properly assigned. ….When time critical messages are assigned a proper priority, the transfer times of the messages remain less than a few milliseconds, even when the load on the implement bus is high. …* “

[25] Die Zeitangabe resultiert aus einem persönlichen Gespräch im ersten Halbjahr 2006 mit dem damaligen Kollegen M. Ehrl, der im Rahmen seiner Forschungsarbeiten die Ansteuerung eines Auslegerstreuers der Fa. Rauch versuchsweise modifiziert hat und dabei Übergangswerte von 650 ms erzielen konnte.

integrierte Sensoren zur Online-Ermittlung des Ernährungszustandes der Pflanzen und künftig von Bodeneigenschaften, zur Positionsbestimmung der Traktor-Gerätekombination sowie zur Ermittlung technischer Größen der Aktorik (Düngerstreuer). Auch das Bedienpersonal der mobilen Einheit stellt eine lokale Informationsquelle dar. Hinsichtlich der externen Informationsquellen ist an erster Stelle das betriebseigene FMIS zu nennen. Ergänzt wird diese Informationsklasse durch die Informationssysteme von Behörden und Drittanbietern, wie zum Beispiel einem Wetterinformationsdienst. Trotz der Fortschritte bei der Funkübertragung erscheint es gerade für eine Real-time Prozessführung mit Kartenüberlagerung vorteilhaft, dass für die geplante Applikationstätigkeit nötige kartierte Informationen, die auf dem FMIS vorliegen, zu einer lokalen Informationsquelle gewandelt werden. Ertragskarten, Fernerkundungsdaten oder „*as applied*"-Karten vorangegangener Düngergaben sollten daher auch auf dem MICS vorgehalten werden. Bis Sensorik zur Umfelderfassung künftig am Mark verfügbar ist, sollten auch bekannte Hindernisse oder Sperrflächen als kartierte Information auf dem MICS vorliegen.Wie im Stand der Technik dargelegt wurde, haben Sensoren zur Online-Ermittlung bodenphysikalischer und bodenchemischer Parameter noch nicht die Reife erreicht wie die Online-Sensorik zur Ermittlung des Ernährungszustandes der Pflanzen. Die Informationen, die sich damit gewinnen lassen, liegen daher nach Kartierung als externe Information auf dem FMIS vor und sollten, falls benötigt, ebenfalls auf dem MICS als lokale Informationsquelle verfügbar gemacht werden.

Mit den ersten Realisierungen von WSN zur Bestimmung von Bodenparametern oder Witterungsdaten könnten dem System auch verteilte Informationsquellen bereitgestellt werden, die zumindest punktuell eine der Online-Sensorik angenäherte Leistung erbringen können.

4.2.2 Level 0 Processing

Wie bereits im entsprechenden Grundlagenkapitel erläutert, sollen auf der Ebene des „*Level 0 Processing - Sub-Object Assessment*" Signale und messbare Merkmale einer Entität entdeckt, extrahiert, gefiltert und räumlich und zeitlich auf Signal/Pixel-Ebene ausgerichtet und abgestimmt werden. Für den Sensor-Ansatz mit Kartenüberlagerung ist dies zum Beispiel für Messungen von reflektiertem Licht oder elektrischer Leitfähigkeit, für GPS-Signale (*SNR, multipath* Effekte) oder zur Verbesserung von Kontrast und Tiefenschärfe in Fernerkundungskarten durchzuführen. Von genereller Bedeutung ist die Analog-Digital-Wandlung der Sensormesswerte, die in der Regel bei der Erfassung als analoges Signal

vorliegen, für die Weiterverarbeitung jedoch in digitalisierter Form erforderlich sind.

4.2.3 Level 1 Processing

Die Ebene „*Level 1 Processing - Object Assessment*“ umfasst das *Data Alignment*, die Daten/Objekt-Korrelation, die Einschätzung der spezifischen Eigenschaften, der Position und der Kinematik eines Objektes sowie die Objektidentifizierung. Die einzelnen Themen wurden in dem entsprechenden Grundlagenkapiteln näher charakterisiert, nachfolgend werden Funktionalitäten des Sensor-Ansatzes mit Kartenüberlagerung den einzelnen Themen zugeordnet.

Data Alignment:
Daten von verschiedenen Informationsquellen müssen im Raum und in der Zeit oder in einem anderem Merkmalsraum oder -referenzsystem ausgerichtet und aneinander angepasst werden.
Eine der wichtigsten räumlichen Angleichungen ist die Einordnung in ein gemeinsames Koordinatensystem. Eine Option ist die Verwendung eines globalen Koordinatensystems wie beispielsweise WGS 84. Ist der räumliche Einsatzbereich beschränkt, so kann jedoch auch auf ein einfaches Rastersystem, wie z.B. in ISO 11783 definiert, zurückgegriffen werden. Die eigenen Koordinatensystemdefinitionen der Traktor-Geräte-Kombination oder der Sensoren sind in das übergeordnete System zu transformieren bzw. daran anzugleichen. Da bei einer Traktor-Düngerstreuer-Kombination nicht stets von einer starren Geometrie ausgegangen werden kann, fällt die Berücksichtigung von Knickbewegungen in diese Rubrik.
Gerade im Zusammenhang mit der teilflächenspezifischen Bewirtschaftung spielt die Berücksichtigung der teilweise stark differierenden räumlichen Auflösung von Online-Sensorik-Messwerten oder kartierter Information eine große Rolle. Je nach Anforderung einer nachfolgenden *Data Fusion*-Technik oder Anforderungen der Applikationstechnik sind die Messwerte und *Precision Farming*-Karten in eine entsprechende Teilschlagrasterung, Konturkarte oder Ausrichtung entlang von Fahrspuren mit statistischen Verfahren umzurechnen.
Neben der räumlichen ist auch eine zeitliche Anpassung der verschiedenen Informationen notwendig. Die Messungen der Online-Sensoren zur Ermittlung des Pflanzenernährungszustandes, der Positionsbestimmung und von Messgrößen der Aktorik sollten hinsichtlich ihrer Differenzen in der Beobachtungsfrequenz und den Zeitverzögerungen zwischen der physikalischen Beobachtung und dem Auswertezeitpunkt eingeordnet bzw. angeglichen werden. Wenn auch nicht mit so hohen

Genauigkeitsanforderungen wie bei der Online-Sensorik sind auch die *Precision Farming*-Karten zeitlich einzuordnen, wie z.B. erste oder zweite Stickstoffgabe auf dem vorliegendem Feld oder die Ertragskarten der einzelnen vorhergehenden Jahre.
Bei unterschiedlichen Informationsquellen kann nicht davon ausgegangen werden, dass die Einheiten der Messwerte/Daten bereits einheitlich vorliegen. Eine Umrechnung der Einheiten mit vorzugsweise absoluter Skalierung ist anzustreben.

Daten/Objekt-Korrelation:
Laut Definition werden bei der Daten/Objekt-Korrelation Beobachtungen mit anderen Beobachtungen in Verknüpfung gebracht. Im Allgemeinen werden verschiedene Sensoren genutzt, um mehrere Objekte und Teile oder verschiedene Attribute eines Objektes zu untersuchen bzw. zu beobachten. Für einen „Pflanzensensor“ gilt es, zu bestimmen, welche Beobachtung zu welchem Objekt/Objektattribut gehört. Stammen die Messwerte von einem einzelnen Punkt auf einem Blatt wie z.B. bei dem „MiniVegN-System“ oder wird stets ein Mischsignal einer größeren Bestandesfläche erfasst wie z.B. beim Yara N-Sensor. Auch stellt sich die Frage, ob bei einem Signal unterschieden werden kann, ob es vom Boden anstatt von einer Pflanze reflektiert wird. Bei einem „Bodensensor“ ist die Zuordnung der Signale zu verschiedenen Bodenschichten wünschenswert.
Die Korrelation neuer Daten mit zeitlich weiter zurückliegenden Daten lässt sich nutzen, um zeitliche oder räumliche Veränderungen beim Fahrweg der Traktor-Geräte-Kombination entdecken und segmentieren zu können. Dies könnte zumindest theoretisch hilfreich sein, um eine Fahrtrichtungsabhängigkeit von Messwerten detektieren zu können. Eine derartige Abhängigkeit könnte beispielsweise durch unterschiedliche Schattenbildung oder den Einfluss des Windes auf die Blattstellung hervorgerufen werden.

Einschätzung der spezifischen Eigenschaften, der Position und der Kinematik von Objekten:
Die Fusion von Sensordaten und Kontextdaten soll die Ableitung von Entität/Objekt-Eigenschaften sowie der Objektposition und -geschwindigkeit ermöglichen. Repräsentative Objekte für die teilflächenspezifische Applikation sind:

a) Pflanzen mit ihren zugehörigen Attributen (z.B. Vegetationsindizes wie REIP oder NDVI oder Indikatoren von Wasser- oder Krankheitsstress),
b) Boden mit seinen Attributen (z.B. Typ, Nährstoffe, Zugkraft, Feuchte),
c) aktuelles Wetter (z.B. Wind, Temperatur, Intensität des Umgebungslichts),
d) Traktor-Geräte-Kombination und seine Position, Geschwindigkeit und Zustand.

Objektidentifizierung:

Die Bestimmung der Identität beteiligter Objekte wie z.B. die Sorte oder die Unterscheidung zwischen kultivierter Pflanze oder Beikraut, aber auch der Typ der eingesetzten Applikationstechnik wie auch der Düngertyp, mit seiner Wirkstoffzusammensetzung, spielt nicht nur für nachfolgende *Data Fusion*-Prozesse, sondern auch für die Dokumentation und Rückverfolgbarkeit eine wichtige Rolle.

4.2.4 Level 2 Processing

Gemäß der Definition umfasst das „*Level 2 Processing - Situation Assessment*“ die Abschätzung, Beurteilung und Vorhersage von Entitätszuständen auf der Basis abgeleiteter organisatorischer, kausaler, biologischer und räumlich-zeitlicher Beziehungen zwischen Objekten. Übertragen auf den Sensor-Ansatz mit Kartenüberlagerung lässt sich die Funktionalität folgendermaßen darstellen:

a) Basierend auf den Pflanzen- und Bodeneigenschaften, abgeleitet durch *Level 1 processing*, unter Einbeziehung des aktuellen Wetters und des aktuellen Zeitpunktes, wird der aktuelle Pflanzen- und Bodenzustand auf der vorliegenden Teilfläche „diagnostiziert“.
b) Weiterer Kontext wie das grundsätzliche Ertragspotenzial basierend auf korrigierten/bereinigten Ertragskarten und den im gleichen Jahr bereits ausgebrachten Düngergaben (*as applied maps*) vergrößern den Lösungsraum und repräsentieren eine detailgetreuere „Abbildung der Wirklichkeit“.
c) Die Integration von Umweltschutzanforderungen, priorisierter Eingaben des Bedienpersonals sowie der Vermeidung von wiederholter Applikation auf der gleichen Teilfläche (z.B. Überlappung) berücksichtigt Grenzen bzw. Einschränkungen.
d) Der Zustand der Traktor-Geräte-Kombination impliziert technische Echtzeit- und Genauigkeits-Schranken bzw. -Einschränkungen für die Applikationstätigkeit.

In einem weiteren Schritt sollten all diese Diagnosen, Beziehungen und Beschränkungen mit einer modellbasierten Vorstellung eines ökologischen und ökonomischen Optimums unter Berücksichtigung der technischen Machbarkeit verglichen werden und in der Empfehlung für einen Applikationssollwert resultieren.

4.2.5 Level 3 Processing

Die Definition des „*Level 3 Processing - Impact Assessment*“ formuliert eine Erweiterung des *Level 2 Processing* in Richtung einer Projektion der aktuellen Situation in die Zukunft, um Schlussfolgerungen über mögliche Handlungsoptionen und -änderungen ziehen zu können.

Angewandt auf den Sensor-Ansatz mit Kartenüberlagerung könnte das Heranziehen von Hypothesen über die Effekte von zukünftigem Wetter auf die Ertragsbildung oder die

Bodenauswaschung zu geänderten Empfehlungen für den Applikationssollwert führen. Auch das Erfahrungswissen -„Ich weiß, dass diese Stelle anders als erwartet reagiert!“- der Bedienperson ließe sich so integrieren. In gleicher Weise könnten auch Hypothesen über die zukünftige Preisentwicklung an den Warenterminbörsen die ökonomische Bewertung für die vorliegende Teilfläche ändern und ebenfalls in einer abgeänderten Applikationssollwert-Empfehlung resultieren. Die Erstellung von Hypothesen über zukünftige Fahrwege der Traktor-Geräte-Kombination bietet Möglichkeiten die Echtzeit-, die Genauigkeits- und die Glättungsfähigkeit[26] bei der Applikationstätigkeit zu verbessern.

4.2.6 Level 4 Processing

Das „*Level 4 Processing - Process Refinement*“ ist als ein Metaprozess definiert, der den gesamten *Data Fusion*-Prozess überwacht, bewertet und stets nachjustiert, um die spezifizierte Leistungsfähigkeit und Effizienz einzuhalten oder zu erreichen.
Die Übertragung auf den zu betrachtenden Anwendungsfall umfasst dabei die Funktionen der Sensorkoordination und -überwachung, der Bestimmung der Sensor-Blickrichtungen (*sensor cueing*) sowie der Ressourcenoptimierung hinsichtlich der Datenübertragungszeiten, der Berechnungszeitenzuteilung für die einzelnen *Data Fusion*-Prozesse oder der Zugriffe auf Speicherressourcen. Werden auch externe oder verteilte Informationsquellen mit herangezogen, so gehören die Etablierung und das Aufrechterhalten von weitgehend transparenten Kommunikationswegen zum Funktionsumfang.

4.2.7 Level 5 Processing oder die Mensch-Maschine-Schnittstelle

Im MSDF-Framework-Vorschlag wurde die Möglichkeit angesprochen, dass bei Bedarf ein „*Level 5 Processing (cognitive refinement)*“ nach BLASCH UND PLANO (2002) [BP02] mit aufgenommen werden könnte. Da bei einer Real-time-Prozessführung für Applikationstätigkeit ja gerade die Automatisierung der Aufgabe und die Entlastung des Nutzers im Vordergrund steht, wird diese Erweiterung als nicht nötig erachtet. Im Vordergrund sollte eher eine möglichst einfache Mensch-Maschine-Schnittstelle stehen, daher hält der Autor für diese Arbeit die Beschäftigung mit dem Modellelement HCI als ausreichend. Um jedoch dem interessierten Bediener nicht jede „*human in the loop*“-Option gänzlich zu versperren, bietet sich für das HCI bzw. MMI folgende Aufteilung an:

a. Ein Automatikmodus mit Informationsanzeige, wenigen Eingabefunktionen und einer

[26] Dies gilt für den Fall, dass kein abrupter Wechsel der Applikationsrate zwischen benachbarten Teilflächen gewünscht ist, sondern ein fließender Übergang erzielt werden soll.

Applikationssollwert-Übersteuerungsfunktion,

b. Ein (evtl. geschützter) Expertenmodus mit erweiterten Eingabefunktionen, Möglichkeiten zur Parametereinstellung und Diagnosemenüs.

Ein manueller Modus ist nicht notwendig, da in diesem Fall auf die als gegeben vorausgesetzte MMI des Düngerstreuers zurückgegriffen werden kann. Der Automatikmodus sollte über einen Start-, Stopp-, Pause- und Fortsetzungs-Schalter[27] verfügen. Der aktuelle abgeleitete Applikationssollwert sollte numerisch und/oder graphisch dargestellt werden, ebenso wie der Istwert (*as applied*). Weiterhin soll die Bedienperson über den herrschenden Zustand der Sensorik, der Steuerungs- und Regelungsfunktion sowie der Applikationstechnik unterrichtet sein. Dabei bietet sich die Unterteilung in drei Zustände mit graphischer Repräsentation in den „Ampelfarben“ an:

a. Grün: Voll funktionsfähig,
b. Gelb: Eingeschränkt funktionsfähig; zeigt das Vorliegen von kurzfristiger Fehlfunktion (z.B. Grenzüber- oder Unterschreitung von Messwerten der Sensorik bzw. das Fehlen von Karteninformation oder vereinzelter Fehlfunktion der Aktorik) an,
c. Rot: Nicht funktionsfähig aufgrund von anhaltender Fehlfunktion (wie z.B. Ausfall von Sensoren oder Aktoren).

Bereits jetzt ist der Nutzer gewohnt, dass er bei Prozesssteuerungen zur Applikationstätigkeit den Sollwert in abgestuften prozentualen Schritten nachjustieren kann. Diese Funktion sollte auch beim Sensor-Ansatz mit Kartenüberlagerung geboten werden. Wichtig dabei ist jedoch, dass derartige Nutzereingriffe im Sinne der Rückverfolgbarkeit dokumentiert werden.

Der Expertenmodus sollte dem Nutzer die Möglichkeit geben, ausgewählte Parameter des *JDL Level 0-4 Processing* einzustellen, als auch die Entscheidungsfindung der Datenfusionsprozesse nachvollziehen oder auf Fehlfunktion analysieren (*debugging)* zu können. Je nach Realisierungstiefe dieser Funktionalitäten ist die anfangs getroffene Annahme nicht mehr haltbar und es lohnt sich, über die Integration des „*Level 5 Processing (cognitive refinement)*“ nachzudenken. Für diese Arbeit soll der Expertenmodus jedoch beschränkt bleiben. Sinnvoll erscheint die Anzeige von Sensormesswerten, Kartendaten der aktuellen Teilfläche, die Bereitstellung einer Debugging-Möglichkeit der *JDL Level 2 Processing Data Fusion*-Prozesse sowie die Auswahl der genutzten Informationsquellen und Fusionsalgorithmen. Weiterhin soll noch die Auswahlmöglichkeit bestehen, was für Dokumentationszwecke dauerhaft abgespeichert wird.

[27] Anm.: Ein Schalter kann in Hardware oder Software realisiert sein.

4.2.8 Datenbank-Management-System

Im Rahmen des Datenbank-Management-System lassen sich zwei Gruppen von Datenbanken unterscheiden. Die in der Definition „*Support Database*“ benannte Funktionalität hält die für den aktuellen Auftrag benötigten kartierten Daten über beispielsweise Feldtopographie, historischen Ertrag, Bodenbeschaffenheit, Zugkraftwerte, Werte der elektrischen Leitfähigkeit des Bodens oder Daten der teilflächenspezifischen Kosten- und Leistungsrechnung vorrätig. Das zweite Standbein des Datenbank-Management-Systems ist die laut Modelldefinition „*Fusion Database”* bezeichnete Funktionalität. Diese dient zur Datenspeicherung und -bereitstellung der Zwischenprodukte und des Endergebnisses des bzw. der *Data Fusion*-Prozesse. Somit stellt diese „*Fusion Database”* die natürliche Schnittstelle für die Dokumentationsfunktionalität der Prozessführung dar. Sowohl das jeweilige Fusionsergebnis, der teilflächenspezifische Applikationssollwert, liegt dokumentiert vor als auch der Entscheidungsweg kann von Experten für spätere Analysetätigkeiten nachvollzogen werden.

4.3 Prozessmodell

Die theoretischen Grundlagen des prozeduralen Modells nach Antony wurden in Kapitel 2.1.5.4 vorgestellt. In diesem Kapitel wird anhand dieses Prozessmodells der Sensor-Ansatz mit Kartenüberlagerung analysiert und ein geeignetes Problemlösungsparadigma abgeleitet.
Wie bereits an vorangegangen Stellen dieser Arbeit angemerkt wurde, erfordert eine Real-time Prozessführung automatisches Schlussfolgern und eine menschliche Interaktion nur in Ausnahmesituationen. Obwohl das Prozessmodell nach Antony alle *JDL functional Levels* umfasst, liegt das Hauptaugenmerk in diesem Kapitel auf den *Data Fusion*-Prozessen des funktionalen *Level 2 Processing*. Diese Fokussierung wurde in der dieser Arbeit zugrundeliegenden Forschungsarbeit im Rahmen der IKB-Forschergruppe Dürnast getroffen. Denn was ist der Sensor-Ansatz mit Kartenüberlagerung im Kern anderes als eine umfassende Situationsbewertung, d.h. eine Abschätzung der aktuellen Online-Sensorik-Messwerte mit einer kontextsensitiven Interpretation. Diese Tatsache unterscheidet diesen Ansatz entscheidend vom reinen Sensor- oder Kartierungs-Ansatz. Jedoch nicht nur diese Motivation und Einschätzung, worin die Neuheit des vorgestellten Ansatz liegt, dienen als Begründung für diese Einschränkung, auch der Stand der Technik bei der Sensorik und dem Kartenmaterial sowie den externen und verteilten Informationsquellen dienen als weitere Begründung. Im Kapitel zum Stand des Wissens wurde auf den kritischen Kommentar von Steinberg und Bowman hinsichtlich einer Missinterpretation ihres funktionalen Modells hingewiesen. Demnach wäre es ein Fehler das Modell als ein Prozessmodell zu verstehen und

dabei anzunehmen, der Informationsfluss im Fusionsprozess muss strikt vom *Level 0* zum *Level 1*, dann *Level 2* und anschließend *Level 3* stattfinden. Dieser „*bottom up*“ Fusionsprozess gilt nur, wenn einerseits Sensorbeobachtungen jeweils in einzelne Messergebnisse aufgeteilt werden können, die von einer echten Entität stammen, anderseits die komplette Information, die für die Einschätzung eines Entitätszustand nötig ist, in den Messungen von einer individuellen Entität enthalten ist. Diese vereinfachende Annahme kann für aktuell vorgesehene Online-Sensoren gelten, deren Beobachtungen über den Pflanzenzustand als Vegetationsindizes berechnet und weiteren Datenfusionsschritten bereitgestellt werden. Auch die Verwendung von einem GPS-Sensor und einer IMU zur Bestimmung der Position und der Kinematik der Traktor-Gerätekombination kann weitgehend als *Level 1 Processing*-Funktionalität angesehen werden und ist vielfach veröffentlichter Stand der Technik.

Jedoch könnten gerade Online-Sensoren für Pflanzenzustandsbewertung von dem Aspekt der Kontextsensitivität mittels *Data Fusion* profitieren. Zum Beispiel könnte die Lösung des Zuordnungsproblems, welcher Teil des gemessenen Lichts von der Pflanze und welcher vom Boden emittiert oder reflektiert wird, erleichtert bzw. erst zugänglich gemacht werden, indem *Level 1 Processing*-Funktionalität mit auf der *Level 2 Processing*-Ebene gewonnenem oder vorhandenem Kontextwissen über den vorherrschenden Pflanzenbedeckungsgrad unterstützt wird. Weiterhin kann davon ausgegangen werden, dass die kartierte Information schon auf dem FMIS vorverarbeitet wurde, so dass bereits die für das *Level 2 Processing* benötigten Werte vorliegen. Eine Umsetzung der *Level 3 Processing*-Funktionalität wurde im Rahmen der IKB-Forschungsarbeit nicht realisiert und wird auch in dieser Arbeit nicht verfolgt. Somit konzentrieren sich die weiteren Ausführungen dieses Kapitels auf das prozedurale Modell für das *Level 2 Processing* und der Bestimmung einer geeigneten Problemlösungsform.

Die nachfolgende Ableitung zeigt jedoch auf, an welchen Stellen des Verfahrens zum Aufstellen des Prozessmodells Unterschiede auftreten würden, wenn ein alle funktionale Ebenen umfassendes Herangehen zum Einsatz kommt.

Das im Folgenden vorgestellte Modell für den Sensor-Ansatz mit Kartenüberlagerung mit Fokus auf dem *Level 2 Processing* geht von einem „*bottom up*“-Ansatz aus, während funktionale Ebenengrenzen übergreifende Fusionsalgorithmen eine Mischung von „*top down“*- und „*bottom up“*-Ansätzen erfordern, um über mehrere Abstraktions-Ebenen hinweg schlussfolgern zu können. Wobei hier die Definition von ANTONY (1995) [Ant95] gelten soll, der jeden funktionalen *JDL processing level* als eine eigene Abstraktions-Ebene definiert.

Abbildung 4.2 stellt das in den theoretischen Grundlagen bereits beschriebene Verfahren zur

Aufstellung des Prozessmodells und der Ableitung eines geeigneten Problemlösungsparadigma für die zugrundeliegende Aufgabenstellung in erweiterter Form dar.

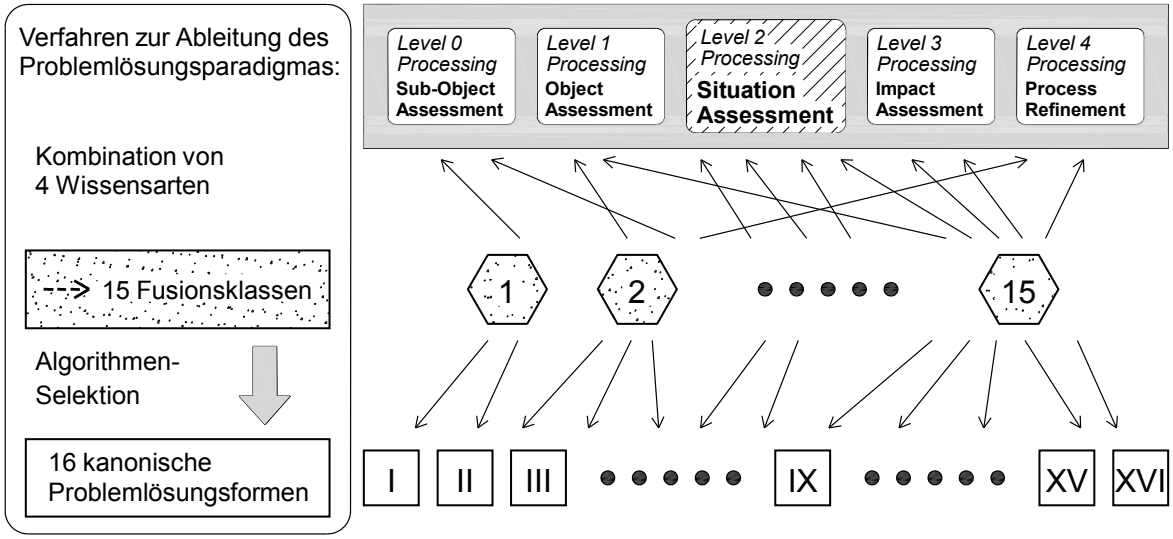


Abbildung 4.2: Verfahren zur Ableitung des Problemlösungsparadigmas

In einem ersten Schritt werden die unterschiedlichen Arten des relevanten Wissens, das für die Erfüllung der *Level 2 Processing*-Funktionalität ausgeschöpft werden soll, bestimmt. Damit lässt sich eine zugehörige Fusionsklasse aus der Menge der 15 verschiedenen Klassen identifizieren. Gemäß der Definition nach Antony stehen die 15 Fusionsklassen mit den 16 kanonischen Problemlösungsformen in einer definierten Beziehung, die vor allem durch die erforderliche Algorithmus-Robustheit und Kontext-Sensitivität bestimmt wird. Somit gilt es, in einem zweiten Schritt (für die vorliegende Anwendung) für diese beiden Kriterien die Festlegungen zu treffen, die dann zu der Auswahl der bzw. einer geeigneten kanonischen Problemlösungsform führen.

4.3.1 Zuordnung der Wissensarten

Den ersten Schritt, die Analyse und die Zuordnung der im funktionalen Modell auf *Level 2 Processing* -Ebene spezifizierten Daten, Information und Fusions-(Zwischen)Ergebnisse zu den einzelnen Wissensarten, zeigt Abbildung 4.3.

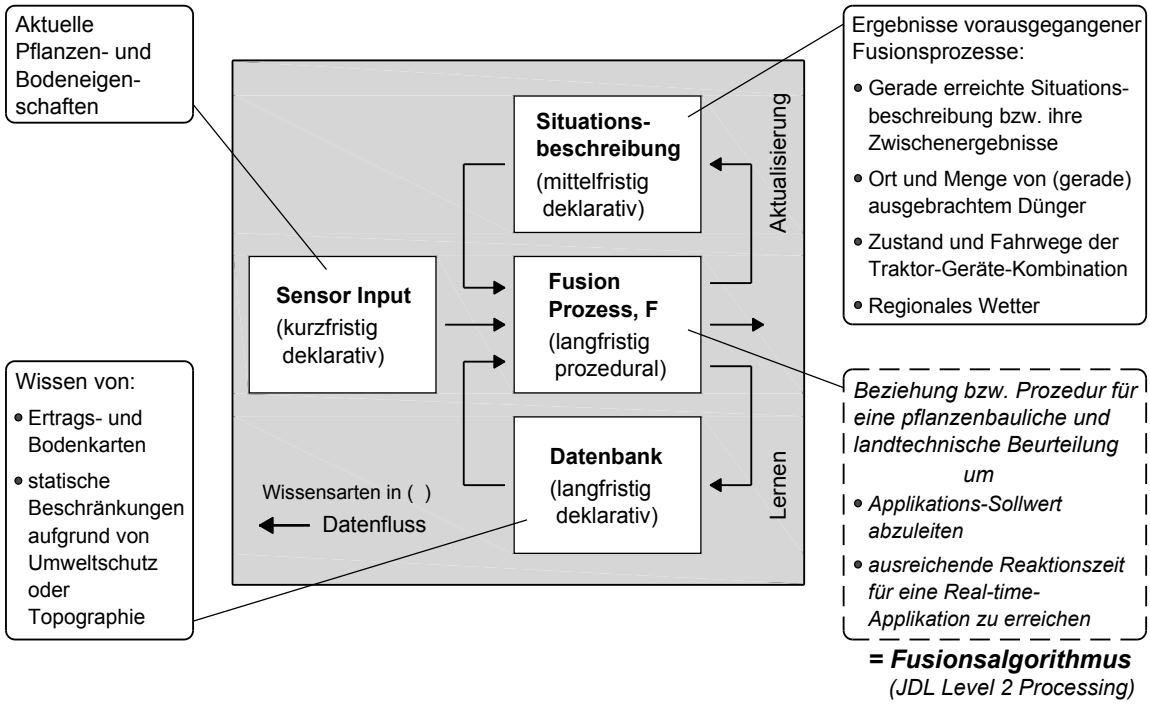


Abbildung 4.3: Wissensarten und Zuordnung zum Prozessmodell

Die aktuellen Pflanzen- und Bodenattribute, ermittelt bzw. beobachtet durch die Online-Sensorik oder durch WSN bereitgestellt, stellen *kurzfristiges deklaratives Wissen* dar.

Dem *mittelfristigen deklarativen Wissen*, kann die Historie der aktuellen Situation auf der vorliegenden Teilfläche, wie die Position und die Menge von (gerade) ausgebrachtem Dünger (*as applied*), der Zustand und der Fahrweg der Traktor-Geräte-Kombination sowie das für die Region gültige Wetter zugeordnet werden. Im weitesten Sinne kann die bereits erreichte und stets aktualisierte Situationsbeschreibung dieser Wissensart zugerechnet werden.

Das *spezifische langfristige deklarative Wissen* ist für den betrachteten Fall sehr einfach zu identifizieren. Diese Art von Wissen wird repräsentiert durch die Ertragskarten, die Kartierung von Bodeneigenschaften, Auflistung von Düngerarten oder Pflanzensorten oder von Maschinenkennwerten. Zu dieser Klasse von Wissen gehören auch die *statischen Domänen-Beschränkungen*, die Umweltschutzvorgaben- oder -zonen ausweisen oder als topographische Karten (DGM) vorliegen.

Hinsichtlich einer MSDF-Lösung fehlt noch das *langfristige explizite prozedurale Wissen*, das den eigentlichen Fusionsprozess darstellt. Für die zu betrachtende Anwendung ist dies eine Beziehung bzw. Prozedur zur pflanzenbaulichen und agrarsystemtechnischen Situationsbewertung, um einen geeigneten Applikationssollwert abzuleiten und dabei ausreichend Reaktionszeit für die eigentliche Applikationstätigkeit zu erreichen. Genau für diese Prozedur, den gesuchten Fusionsalgorithmus, gilt es im zweiten Schritt ein geeignetes

Problemlösungsparadigma aus der Gruppe der kanonischen Problemlösungsformen abzuleiten.

4.3.2 Bestimmung des Problemlösungsparadigmas

Zufolge der Analyse und der Zuordnung zu den verschiedenen Wissensarten erfordert diese Aufgabenstellung die Komposition von kurz-, mittel- und langfristigem deklarativen und langfristigem expliziten prozeduralen Wissen. Eine Aufgabenstellung, die diese Kombination aller vier Wissensarten erfordert, ist laut dem angewandten prozeduralen Modellansatz ein Kandidat für die Fusionsklasse 15. Selbst nach Antony's Taxonomie gibt es nicht nur einen generischen Problemlösungsansatz, der für eine Fusionsklasse 15 in Frage kommt. Bei der weiteren Einengung auf ein geeignetes Problemlösungsparadigma hilft die Betrachtung der Kriterien Robustheit, Kontext-Sensitivität, Erweiterbarkeit, Wartbarkeit und Effizienz sowie die Charakterisierung von *langfristigem expliziten prozeduralen Wissen* nach der Modelldefinition. Der nachfolgend beschriebene Selektionsprozess ist in Abbildung 4.4 grafisch aufbereitet.

Wie in den Grundlagen (vgl. Kap. 2.1.5.4) bereits angeführt, lässt sich prozedurales Wissen wiederum aufteilen in einen Teil langfristig deklarativen Wissens und dem zugehörigen Kontrollwissen (*control knowledge*) als zweiten Teil.

Das deklarative Wissen ist entweder spezifisch oder generell. Sind alle folgenden Voraussetzungen gegeben, und zwar dass der Entscheidungsprozess nur eine begrenzte Menge an Freiheitsgraden hat, die Parameter und Eigenschaften relativ unabhängig voneinander sind, d.h. keine komplexen Zusammenhänge zwischen den Attributen existieren, und das relevante Schlussfolgerungswissen statisch ist, dann können Fusionsalgorithmen basierend auf einem spezifischen langfristigen Wissen sinnvoll sein [Ant01]. Für Aufgabenstellungen, die jedoch mehrere Freiheitsgrade, komplexe Abhängigkeit der Variablen und eine inhärente Dynamik aufweisen wird die Leistungsfähigkeit des Fusionsprozesses durch spezifisches Wissen eingeschränkt und ein Algorithmus, der auf generellem langfristigen deklarativen Wissen fußt, wird vorteilhafter sein. Für den Sensor-Ansatz mit Kartenüberlagerung sind die Voraussetzungen für spezifisches Wissen nicht gegeben, da z.B. die Variablen Pflanzenzustand, Bodeneigenschaften oder Ertragspotential sicher nicht als unabhängig angesehen werden können. Auch im Hinblick auf Erweiterbarkeit und zukünftiger Entwicklung von Online-Sensoren und *Precision Farming*-Karten ist eine Beschränkung an dieser Stelle nicht anzustreben. Damit sind die kanonischen Problemlösungsformen I-VIII ausgeschlossen.

Die Ableitung des geeigneten Paradigma erfolgt somit anhand der Unterscheidung des Kontrollwissens in rigid oder flexibel und der Eignung für eine oder mehrere Abstraktionsebenen. Durch die getroffenen Annahmen für diese Modellbildung ist die Wahl eines Algorithmus, der sich auf eine Abstraktionsebene beschränkt, vorgegeben bzw. ausreichend. Dies schränkt die möglichen Problemlösungsformen auf die Klassen IX, X, XIII und XIV ein.

Laut ANTONY (2001) sind die kanonischen Formen höherer Ordnung (IX-XVI) potentiell kontextsensitiv, da sie durch ihr generelles deklaratives Wissen sensitiv gegenüber Domänenwissen sind, das nicht von einem Sensor abgeleitet wird [Ant01]. Jedoch unterstützten nur die Formen XIII-XVI sowohl kontextsensitives deklaratives Wissen und kontextsensitives Kontrollwissen. In der vorgesehenen Ausbaustufe des Sensor-Ansatzes mit Kartenüberlagerung wird davon ausgegangen, dass kontextsensitives deklaratives Wissen ausreichend ist, d.h. die Fusionsalgorithmen sollen nicht zur Laufzeit in ihrem grundsätzlichen Wirkmechanismus geändert werden. Die kontextsensitive Bewertung von Online-Sensorwerten wird auch durch das damit gewählte rigide Kontrollwissen gewährleistet. Dieser Auswahl liegt außerdem der Gedanke zugrunde, dass die Komplexität der Algorithmen mit kanonischen Formen höheren Grades ansteigt.

Folglich bleiben noch die beiden Problemlösungsformen IX und X übrig. Diese beiden kanonischen Formen sind charakterisiert, dass sie auf generellem deklarativen (Kontroll-)Wissen basieren, eine rigide Kontrollstruktur, geeignet für eine Abstraktionsebene, aufweisen und einen Ansatz repräsentieren, der eine einfache modellbasierte Transformation darstellt.

Da die Funktionalität des *Level 2 Processing* fundamental datengetrieben ist, ist ein *generation-based* Algorithmus am besten geeignet. Eine g*eneration-based* bzw. datengetriebene Problemlösung bedeutet, dass bei einer „*Black Box*"-Betrachtung eine Menge von Eingangszuständen in eine Menge von Ausgangszuständen transformiert wird. In der Konsequenz wurde damit die Entscheidung zwischen der hypothesengetriebenen (*hypothesis-based)* kanonischen Problemlösungsform X zugunsten des *generation-based* Problemlösungsansatzes „*canonical form IX*" getroffen. Ein typischer Vertreter dieses Ansatzes ist ein konventionelles Expertensystem mit seinem Produktionssystem-Paradigma und vorwärtsverkettetem Inferenzmechanismus (vgl. Kap. 5.1).

Im Vertrauen auf Antony's Einschätzung, dass sich modellbasierte Ansätze in der Regel effizienter und robuster als nicht modellbasierte Ansätze erweisen, fällt auch eine Prüfung auf die Kriterien Robustheit und Effizienz zufriedenstellend aus. In Anbetracht des Einsatzes von

generellem deklarativen (Kontroll-)Wissen sollte sich auch die Erweiterbarkeit und Wartbarkeit der Problemlösung einfacher gestalten als bei Einsatz eines rein spezifischen Kontrollwissens.

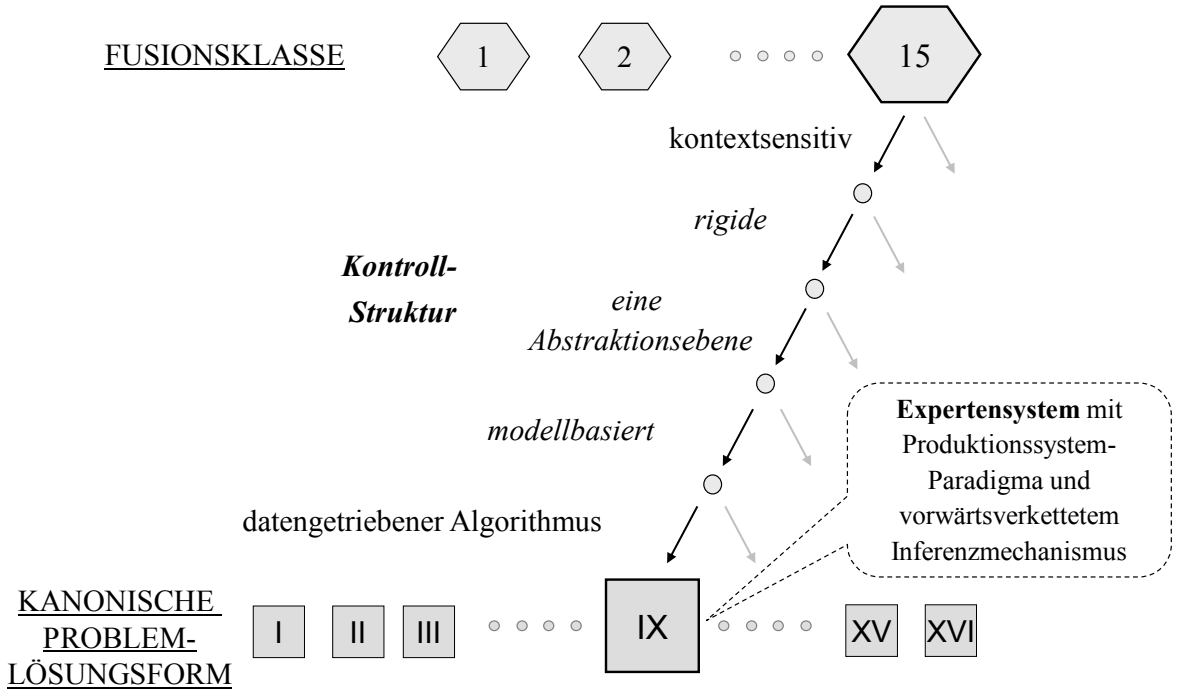


Abbildung 4.4: Ableitung des Problemlösungsparadigmas

4.4 Systemarchitektur

Wie in dem MSDF-Framework vorgeschlagen, wird für die Ableitung und den Entwurf einer Systemarchitektur zuerst eine der bekannten und bewährten Systemarchitekturvarianten für MSDF-Systeme gewählt. Die grundsätzliche Anpassung an das Landwirtschaftliche BUS-System ISO 11783 erfolgt über den Zwischenschritt der Zuordnung der unterschiedlichen Wissensarten zu Systemelementen/-komponenten der Norm. An Stellen, an denen keine ausreichende Normdefinition vorliegt, wird eine Erweiterung vorgeschlagen.

Es liegt auf der Hand, dass die Rohdaten eines Spektrometers, einem Online-Sensor zur Beobachtung des Pflanzenzustandes, die Daten einer Ertragskarte oder die Daten einer EM38-Messung aus der Beobachtung sehr ungleichartiger physikalischer Phänomene stammen. Daher liegen im Fall des Sensor-Ansatzes mit Kartenüberlagerung *noncommensurate* Datenquellen als Eingangsgrößen vor, was eine Fusion auf Rohdatenebene ausschließt. Stattdessen ist eine MSDF auf Merkmals/Zustands- oder Entscheidungsebene gefordert.

Prinzipiell kann eine derartige Datenfusion mit einer Systemarchitektur des Typs „*Central Fusion Processing*“ realisiert werden. Dabei würden alle Sensorrohdaten, Kartendaten und

Fusionszwischenergebnisse einem zentralen (*Data Fusion*)-Prozessor zugeführt und dort verarbeitet werden. Unter Beachtung der grundsätzlichen Randbedingungen des ISOBUS erscheint diese Architektur wenig geeignet und auch nicht erweiterungsfähig. So ist der ISOBUS entgegen der Insellösung und dem Mobilen Agrarcomputer als verteiltes und nicht als zentrales System konzipiert. Gerade die direkte Anbindung der Sensor-(Analog)werte an den zentralen Steuerungs- und Regelungsrechner ist ein Kennzeichen der Insellösung oder des Mobilen Agrarcomputers und nicht eines Landwirtschaftlichen BUS-Systems, denn ein landwirtschaftliches BUS-System stellt allen Systemteilnehmern ein gemeinsames Kommunikationssystem zur Verfügung. Die beschränkte Übertragungsgeschwindigkeit beim ISOBUS (250 kbit/s) schränkt die Übertragung von Sensorrohwerten oder bandbreitenhungrigen Datentypen, wie sie z.B. für bildverarbeitende Systeme typisch sind, stark ein. Weitere grundsätzliche Vorgaben des ISOBUS sind, dass die einzelnen Systemkomponenten von unterschiedlichen Herstellern stammen können und das Gesamtsystem je nach Anwendung konfigurierbar und skalierbar sein soll.

Von den repräsentativen Systemarchitekturen nach WALTZ UND LLINAS (1990) [WL90] ermöglicht der Systemarchitekturtypus „*Distributed Sensor/Fusion*“ die Umsetzung des Sensor-Ansatz mit Kartenüberlagerung in einem mobilen landtechnischen BUS-System. So sind die einzelnen Sensorknoten physikalisch verteilt an der Traktor-Geräte-Kombination angebracht. Diese Sensorknoten sind beispielsweise die Messköpfe eines Spektrometers oder das *Frontend* eines GPS-Empfängers oder die Beschleunigungs- und Drehratensensoren einer IMU. Dabei hat jeder Sensorknoten einen lokalen, zentralen Kontenprozessor (NP), der die lokale *Data Fusion* (*JDL Level 0* und/oder *1 Processing*) durchführt. So fasst der Knotenprozessor für einen „Online-Sensor Pflanze“ die Beobachtungen der einzelnen Sensormessköpfe zusammen und berechnet einen Vegetationsindex. Der Knotenprozessor für das GPS-Empfänger-*Frontend* und die IMU berechnet aus den Messungen die Position und Kinematik der Einheit. Jeder Knotenprozessor leitet seine aufbereitete Information über das gemeinsame Kommunikationssystem, dem Bussystem nach der ISO 11783 Definition, an einen oder mehrere globale Prozessoren (P) weiter, die die *JDL Level 2 und 3 Processing*-Datenverarbeitung durchführen. Für den vorliegenden Fall ist dabei das *JDL Level 2 Processing* ausreichend.

Für die Darstellung des Ergebnisses auf einem Display (D) ist im ISOBUS die Funktionalität eines *Virtual Terminal* vorgesehen. Eine schematische Zusammenfassung obiger Verschläge zeigt Abbildung 4.5 mit einer leicht modifizierten Systemarchitektur „*Distributed Sensor/Fusion*“. Die Modifikation betrifft die Einführung des gemeinsamen CAN-Bussystems

nach ISO 11783 zur Datenkommunikation.

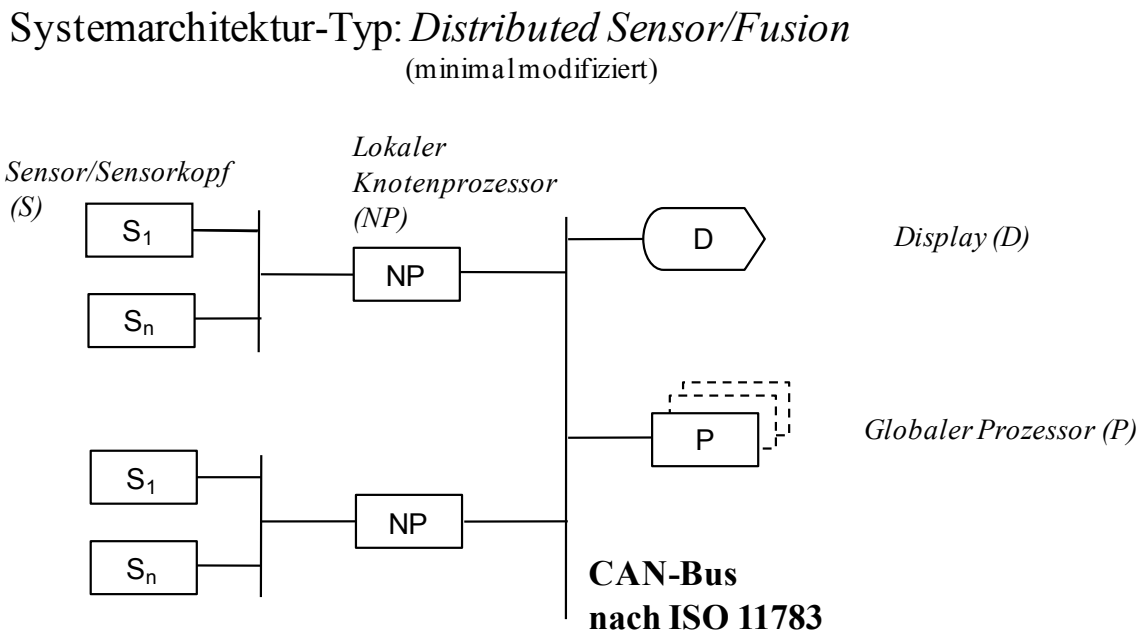


Abbildung 4.5: Systemarchitektur gemäß dem Typ „*Distributed Sensor/Fusion*“

4.4.1 Zuordnung der Wissensarten & Prozessmodell-Elemente in einem ISOBUS-System

Bei der Bestimmung der Fusionsklasse wurde analysiert, dass kurzfristiges deklaratives Wissen, mittelfristiges deklaratives Wissen, spezifisches langfristiges deklaratives Wissen und langfristiges explizites prozedurales Wissen erforderlich sind und sich somit auch in einer ISOBUS-konformen Systemarchitektur wiederfinden sollten. Eine derartige Übersetzung der *„Distributed Sensor/Fusion*“-Systemarchitektur in ein Prozessführungssystem nach ISOBUS zeigt Abbildung 4.6. Gleichzeitig wurden die benötigten Wissensarten ISOBUS-Systemkomponenten zugeordnet bzw. werden durch diese letztendlich in Hard- und Software umgesetzt.

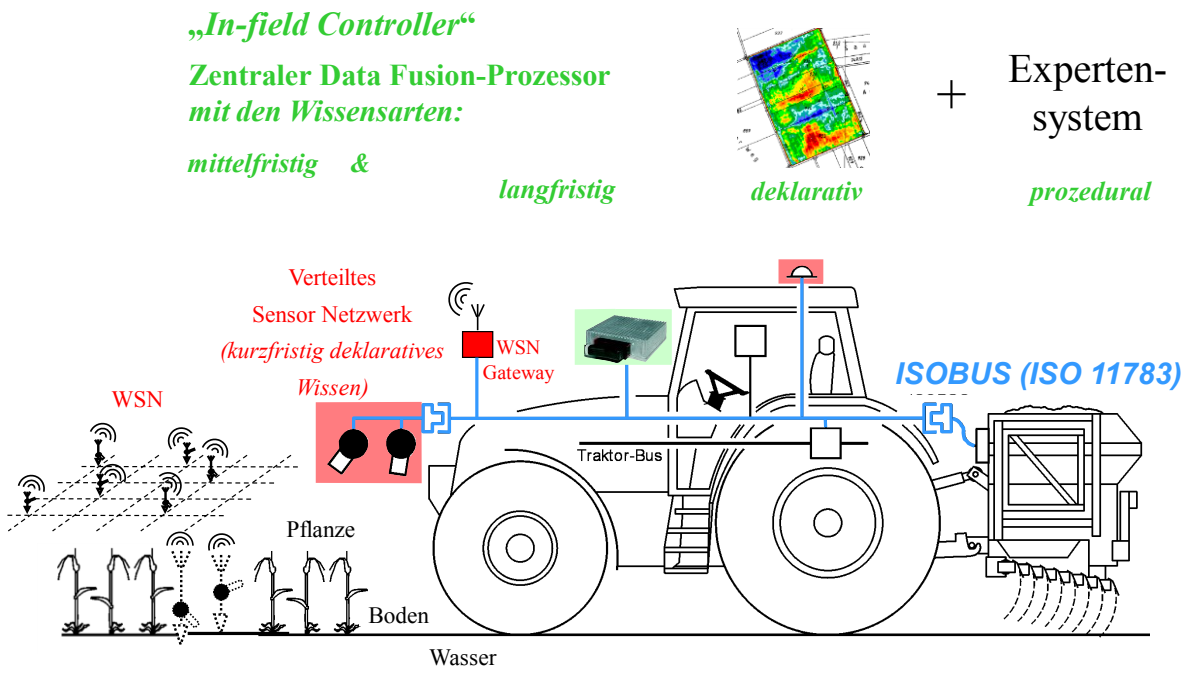


Abbildung 4.6: Systemarchitektur mit „*In-field Controller*“ und den zugeordneten Wissensarten

Das *kurzfristige deklarative Wissen* ist der Online-Sensorik und dem WSN für die Ermittlung der aktuellen Pflanzen- und Bodenattribute sowie der Sensorik des Traktors und der Applikationstechnik zugeordnet. Die Funktion eines Knotenprozessors wird dabei durch einen Geräterechner (z.B. TECU), ein Gerät (*device*) (z.B. Navigation, GPS) oder eine „Funktion“ (*control function*) bereitgestellt. Das Thema des Knotenprozessors für die „Pflanzen- oder Boden-Onlinesensoren“ sowie die Thematik der Anbindung der WSN wird nachfolgend noch weiter erörtert (vgl. Kap. 4.4.3).

Ein zentraler *Data Fusion*-Prozessor hält das *mittelfristige deklarative Wissen* und das *spezifische langfristige deklarative Wissen* vor und führt mittels dem *langfristigem expliziten prozeduralen Wissen* die *JDL Level 2 Processing*-Datenverarbeitung durch.

OSTERMEIER ET AL. (2003) [OAD03] haben diesen zentralen *Data Fusion*-Prozessor bzw. dessen Funktionalität für ein mobiles landtechnisches BUS-System die Bezeichnung „*In-field Controller*” gegeben. Erstmals wurde diese Namensgebung in der Antragstellung für die 2. Phase des IKB-Projektes von AUERNHAMMER (2001) [Aue01b][28] für einen Teilnehmer eines Landwirtschaftlichen BUS-Systems, der alle „*in field*“-Aktivitäten überwachen, steuern und dokumentieren kann, eingeführt.

Wie bereits im Stand der Technik ausgeführt worden ist, wird die vorgeschlagene Struktur

[28] A. a. O., S. 2

noch nicht vollständig von der ISOBUS- (und LBS-) Norm unterstützt. Die Teilflächenbewirtschaftung war nur im Kontext des Kartierungsansatzes in diesen Normen definiert. Erst in jüngerer Zeit hat die Integration von Online-Sensorik im Rahmen des Sensor-Ansatzes Fortschritte bei der Definition im ISOBUS-Standard erfahren. Dabei wurde zumindest eine Möglichkeit zur Überlagerung mit Karten auf Entscheidungsebene mit aufgezeigt. Einer umfassenden Behandlung des Themas *Data Fusion* wird jedoch keine explizite Aufmerksamkeit geschenkt. In den nachfolgenden Unterkapiteln sollen von der vorgeschlagenen Systemarchitektur betroffene Normteile kommentiert bzw. nötige Anpassungen der Norm besprochen werden, die ein „*In-field Controller*" bedingt, der mittels eines integrierten Expertensystems Daten der Online-Sensorik (z.B. Vegetationsindex: REIP) und Daten der *Precision Farming*-Karten (historischer Ertrag, EM38, Zugkraft, applizierte Düngermenge des gleichen Jahres) sowie Beschränkungen aufgrund des Umweltschutzes in Echtzeit im Feld fusionieren kann, um schließlich den geeigneten Düngermengensollwert abzuleiten.

Abbildung 4.7 fasst nun die angestellten Überlegungen zur Systemarchitektur in einem Systemdiagramm zusammen. Schematisch dargestellt ist dabei die Skalierbarkeit des Prozessführungssystems mit den dafür benötigten ISOBUS-Komponenten zur Realisierung der Systemansätze Kartierungsansatz, Sensor-Ansatz und Sensor-Ansatz mit Kartenüberlagerung (für Applikationstätigkeiten), die im Diagramm farblich gekennzeichnet sind. Die Komponenten Geräterechner (*Implement Controller*), TECU (*Tractor ECU)* und *Virtual Terminal* sind allen Ansätzen gemeinsam. Das GPS-System wäre für den reinen Sensor-Ansatz nicht von Nöten, da jedoch auch hier (lokalisierte) Dokumentation der Istwerte erstrebenswert ist, ist seine Existenz zumindest wünschenswert. Aus diesem Grunde macht auch die Anbindung des FMIS an das MICS nicht nur für den Kartierungs- und Sensor-Ansatz mit Kartenüberlagerung Sinn. Die Einbeziehung der *File Server*-Funktionalität ist optional, sie bietet jedoch *Data Fusion*-Aktivitäten einen Vorteil.

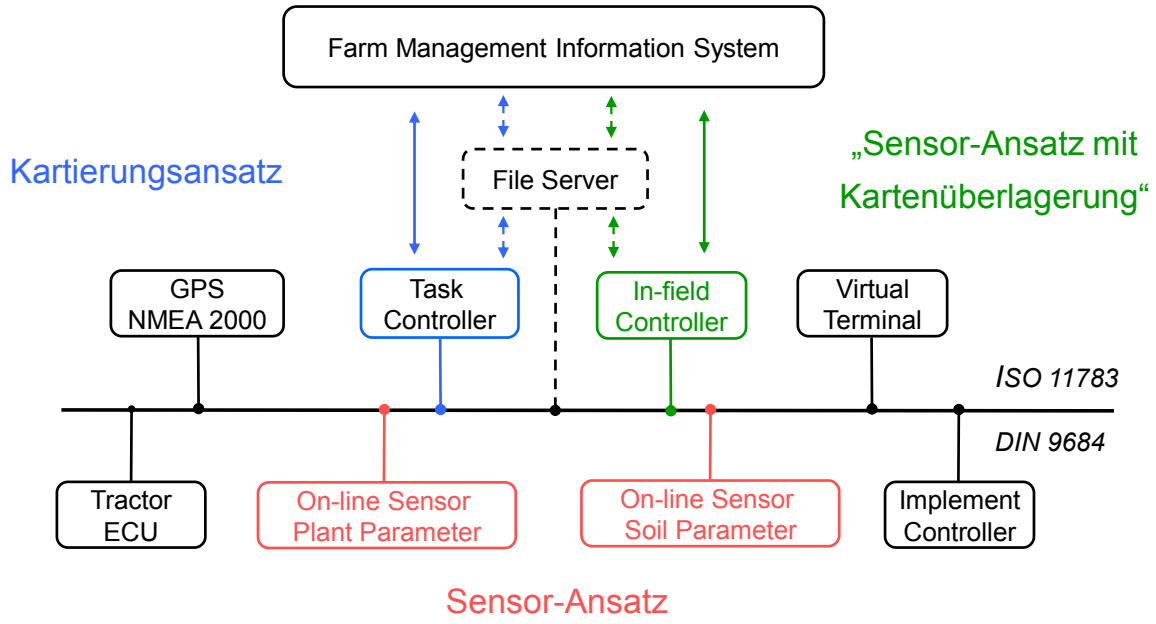


Abbildung 4.7: Systemdiagramm mit den Systemansätzen für die Teilfächenbewirtschaftung

4.4.2 Zentraler Data Fusion-Prozessor „*In-field Controller*“

Der offensichtliche Weg wäre die Erweiterung des *Task Controllers* um die *Data Fusion*-Fähigkeiten des „*In-field Controllers*“. Dies könnte auch im Rahmen einer unterschiedlichen IL-Definition erfolgen. Da dies noch nicht absehbar ist und im Sinne einer zukünftigen Erweiterungsfähigkeit, Skalierbarkeit und Definition eines zukünftigen landtechnischen BUS-Systems wird der „*In-field Controller*“ als eigenständiger Teilnehmer definiert und zeigt auch auf diesen Weg, welche Erweiterung des *Task Controllers* wünschenswert wäre.

Die Realisierung eines „*In-field Controller*“ erfordert eine eindeutige Netzwerkidentität als Basis einer systemkonformen Teilnahme am landwirtschaftlichen BUS-System und somit der Möglichkeit Prozessdaten mit den anderen Teilnehmern und Auftrags-, Dokumentations- und Kartierungsinformation mit dem FMIS standardisiert austauschen zu können. Die ISOBUS-Netzwerk-Identität wird durch eine „*Device Class*“ und „*Function*“-Definition gemäß der Festlegungen zur Namensgebung („*NAME*“) im ANNEX E der [ISO11783-1] gewährleistet. Wie der *Task Controller* sollte der „*In-field Controller*“ innerhalb der „*Industry Group*“ „*Agricultural and Forestry equipment*“ unter „*Device Class*“ „*Non-specific systems*“ als Funktion „*In-field Controller*“ definiert werden. Wie bereits einleitend beschrieben, wären die weiteren Normfestlegungen zwangsläufig eine Kopie der Definitionen zur Auftragsbearbeitung des *Task Controllers*, die an einigen Stellen erweitert bzw. angepasst werden müssten.

Der „*In-field Controller"* muss in der Lage sein, das *spezifisch langfristige deklarative Wissen* mit dem FMIS austauschen zu können. Dazu sollten die bereits in [ISO11783-10] definierten Strukturen für den Datenaustausch zwischen *Task Controller* und FMIS soweit möglich übernommen und ansonsten adaptiert werden. Die Übertragung der kartierten Informationen und deren Speicherung erfordern Grundfunktionen eines GIS. Die Daten sollten dabei sowohl im Rasterformat oder Vektorformat mit räumlichen und zeitlichen Zuordnungen ausgetauscht werden können. Die Unterstützung eines 2D-System ist notwendig, 3D-Systemunterstützung wünschenswert. [ISO11783-10] unterstützt die beiden Formate, das Rasterformat (*„Grid (GRD)"*) und das Vektorformat (*„Polygon (PLN)"*). Es existiert bereits eine Definition für die Repräsentation von Applikationskarten und deren Austausch mittels des zentralen XML-Elements „*TreatmentZone (TZN)*" ([ISO11783-10] 6.6.1 Site-specific application). Die sinngemäße Ausrichtung zielt auf den Prozessdatenaustausch von Applikationssollwerten ab. Auch für die georeferenzierte Datenerfassung, dem „*Data logging*", wurden bereits Definitionen getroffen ([ISO11783-10] 6.6.2 Data logging) mit den zentralen XML-Elementen „*DataLogTrigger (DLT)*" und „*DataLogValue (DLV)*". Über die XML-Elemente „*Time (TIM)"* und *„TimeLog (TLG)"* lässt sich neben der Georeferenzierung auch eine zeitliche Zuordnung herstellen ([ISO11783-10] 6.2 Logging of time and position). Zugleich ist über *„TimeLog (TLG)"* auch die Möglichkeit definiert, auf externe Dateien im Binärformat zu verweisen und dort prozessdatenbezogene Datenerfassung abzulegen. Jedoch liegt bei dem „*Data-Logging*" bisher das Hauptaugenmerk auf der Datenerfassung während der Ausführung eines Auftrages („*Task (TSK)*"), sozusagen auf der (Betriebsdaten-)Dokumentation. In Grunde handelt es sich zum Zeitpunkt der Speicherung noch um kurzfristiges deklaratives Wissen, das auf dem MICS für das FMIS gesammelt wird und dort nach optionaler Weiterverarbeitung als langfristig deklaratives Wissen vorgehalten wird. Für den „*In-field Controller"* ist es jedoch entscheidend, dass nicht nur die Stammdaten („*Coding Data*") und Applikationskarten als *spezifisch langfristiges deklaratives Wissen* bereitgehalten werden können, sondern auch eine breite Klasse an kartierter Information. Somit sollte ein neues XML-Element „*Überlagerungskarte"* (*„Overlay-Map (OMP)"*) definiert und von dem Element „*Task (TSK)*" aus gesehen in einer 1-n Relation angebunden werden. Die vorher aufgeführten XML-Elemente und Beziehungsstrukturen dienen als Vorlage für die Detaildefinitionen. Vorrangig sind dabei die Definition und Relationen der „*TreatmentZone (TZN)*". Zusätzlich ist aber minimal noch ein XML-Element für Metainformation zu definieren, das eine zeitliche Referenz des Kartenmaterials vorhält sowie Angaben zu den in der Kartierungsphase angewandten Bereinigungsalgorithmen, den (geo-statistischen)

Interpolationsverfahren und deren Kenngrößen macht, im Speziellen zu Genauigkeit (, Werteklassen und gewählter Rasterbildung).

Mit der Definition der „*Überlagerungskarte“* ist die wichtigste Art von *spezifischen langfristigen deklarativen Wissen* abgedeckt. Dennoch sollte auch eine Möglichkeit zur Bereitstellung dieser Wissensart in weitgehend format-ungebundener Form integriert werden. Dazu bietet es sich an, freien Text zu nutzen, mit dem beispielsweise aus Erfahrung bekanntes Mikroklima eines Feldes beschrieben werden kann. Hierfür ist nicht unbedingt eine Neudefinition nötig, da bereits XML-Elemente zur Kommentierung (*comment)* spezifiziert wurden. Neben bereits vorab definierbaren Kommentaren („*CodedComment (CCT)*“) besteht über das XML-Element „*CommentAllocation (CAN)*“ auch die Möglichkeit freien Text als Kommentar zu übergeben.

Für eine Art von langfristigem Wissen wurden in der Norm [ISO11783] bisher keinerlei Festlegungen zum Datenaustausch getroffen. Das *langfristige explizite prozedurale Wissen* wird vielmehr als integraler Teil der *Task Controller*-Implementierung betrachtet und erfordert hierfür auch keine Datenaustauschmöglichkeit außerhalb der standardmäßigen Programmwartung und -pflege. Für das mittels des prozeduralen Modells abgeleitete Problemlösungsparadigma in Form eines Produktionssystems (mit vorwärtsverkettetem Inferenzmechanismus) besteht jedoch sehr wohl die Möglichkeit und evtl. auch der Bedarf auftrags- bzw. betriebsspezifisch den deklarativen Teil des Kontrollwissens, d.h. die Wissensmodule des Regelwerks, zu wechseln und auszutauschen. XML wurde für ISO 11783 bereits als Standardaustauschformat gewählt und ist auch für die Repräsentation von Regeln (*rules*) geeignet. FRIEDMAN-HILL (2003) [Fri03] beschreibt ein Verfahren zur XML-Definition, verweist jedoch auch auf zwei Initiativen, die sich eine entsprechende Standardisierung zum Ziel gesetzt haben. Dies ist zum einem „*RuleML*“, zum anderen ist es „*DAML*“. Beide Initiativen haben bisher zu keinem weltweit akzeptierten Standard geführt. Jedoch haben Mitglieder der RuleML-Initiative auch an der Definition des „*W3C Rule Interchange Format (RIF)*“ mitgewirkt. Im Juni 2010 hat RIF den Status einer „*W3C-Recommendation*“, sozusagen den Status eines technischen Standards für das weltweite Internet, erreicht und hierin einen „Dialekt“ speziell für Produktionssysteme, den „*RIF Production Rule Dialect*“, definiert [W3C10]. Auf Basis dieses weltweiten Standards sollte der Austausch von *langfristigen expliziten prozeduralen Wissen* auch für ISOBUS betrieben werden.

Entsprechend der Wissensklassenzuordnung hält nur der „*In-field Controller”* das *mittelfristige deklarative Wissen* vor. Dadurch ergibt sich im Prinzip keine Notwendigkeit für

eine Definition zum Datenaustausch, außer eine Nachweispflicht oder ein Analysebedarf erfordert die Dokumentation der Lösungsfindung mit ihren „Zwischenergebnissen“, wie z.B. der Position und der Menge von (gerade) ausgebrachtem Dünger oder einer Momentaufnahme der Situationsbeschreibung, z.B. in Form von Merkmals- und Zustandswerten. Die obigen Festlegungen zum Datenaustausch für *spezifisches langfristiges deklaratives Wissen* zeigen bereits eine Lösungsoption auf. Anzumerken bleibt, dass gerade diese Art von Dokumentation des *mittelfristigen deklarativen Wissen* der Anteil an der Betriebsdokumentation ist, der durch die *Task Controller*-Definition bisher nicht abgedeckt ist und als „*In-field Controller”*-Funktionalität erbracht wird.

4.4.3 Integration Online-Sensorik

Gemäß den Festlegungen im Teil 7 der Norm [ISO11783-7] wird bereits eine umfangreiche Sammlung von Messwerten der Traktorsensorik als Basisbotschaften gesendet bzw. sind auf Abruf verfügbar. Ein Beispiel ist die PGN-Nachricht „*Ground-based speed and distance*“ (PGN 65097 ($00FE49_{16}$)) mit den Parametern Geschwindigkeit, zurückgelegte Wegstrecke und Angabe zur Bewegungsrichtung (vorwärts oder rückwärts) der Traktor-Geräte-Kombination. Auch die Funktionalität eines Online-Sensors „Position“ ist bereits gut in den ISOBUS eingebunden. So ist in [ISO11783-7][29] definiert, dass die Positions- und Navigationsinformationen gemäß den (Parameter)Festlegungen der Norm IEC 61162-3 (NMEA 2000) genutzt werden sollen. Hierfür existieren mehrere PGN’s zum Nachrichtenaustausch der unterschiedlichsten Daten, z.B. PGN 129029 „*GNSS Position Data*“. Auch einfache Information zur Zustandsbeschreibung des Anbaugerätes sind als Parameter „*Implement transport state*“ (SPN 1869), „*Implement park state*“ (SPN 1879) und „*Implement work state*“ (SPN 1871) bereits definiert.

Vor der Einführung der Geräteklasse (*device class*) „*Sensor systems*“ in den ISOBUS ([ISO11783-1][30] stellte sich die Situation zur Integration von Online-Sensoren und WSN wie bereits auch bei LBS nicht einfach dar. Um Sensoren und ihre Prozessdaten, vor allem Messwerte, in den ISOBUS zu integrieren und zu nutzen, ist die Zuordnung bzw. Verbindung mit einer Geräteklasse von Nöten. Während sich diese Zuordnung für Sensoren der

[29] A. a. O., S. 104: B.5 Navigation location system messages:
“ISO 11783 networks shall use the navigation location messages specified in IEC 61162-3 (NMEA 2000). The preferred (minimum) messages for ISO 11783 are GNSS position data, position, rapid update and GNSS pseudo-range noise statistics. Messages requiring multiple data frames shall use the NMEA fast packet protocol rather than the transport protocol specified in ISO 11783-3.”

[30] A. a. O. S. 66

definierten Geräteklassen, wie z.B. Traktor, Pflanz- und Sämaschine, Düngerstreuer oder Pflanzenschutzspritze, einfach gestaltet, ist die Situation bei einem Online-Sensor „Pflanze“ oder „Boden“ schwieriger. So kann der Online-Sensor „Pflanze“ sowohl im Rahmen einer Applikationstätigkeit mit einem Düngerstreuer als auch mit einer Pflanzenschutzspritze Sinn ergeben. Jedoch benötigt er für eine eindeutige Identifizierung im ISOBUS-Netzwerk eine eindeutige *NAME*-Zuordnung in Verbindung mit einer Festlegung auf eine Geräteklasse und mit einer Instanznummer. Im Fall einer Zuordnung des Online-Sensors zur Klasse Düngerstreuer wäre er ein Teil dieses Düngerstreuers und seine Messwerte werden auch nur im Zusammenhang mit diesem erfasst, gespeichert und weitergeleitet. Lange Zeit war der gültige Status der Norm für die *NAME*-Festlegungen [ISO11783-5a] und hatte als einzige Möglichkeit einer dynamischen (, d.h. zur Systemlaufzeit,) Neukonfigurierung von *NAME*-Teilen die Änderung der „*Self-configurable address*“ jedoch nicht die Änderung der Zuordnung zur Geräteklasse oder der Instanznummer vorgesehen. Mit der neuen Version [ISO11783-5b] des Jahres 2011 wurde ein neues „*NAME-Management*“[31] eingeführt, das auch dies optional ermöglicht. Auch wenn dies eine Zuordnung eines Online-Sensors zur Laufzeit zu der gerade aktuellen Applikationstechnik ermöglicht, wird es dem generischen Charakter eines Online-Sensors nicht gerecht, der für verschiedene, auch eigenständige Dokumentations-, Steuerungs- und Regelungsaufgaben in das ISOBUS-System integriert werden soll. Mit der Einführung der Geräteklasse „*Sensor systems*“ besteht jetzt jedoch die Möglichkeit, Online-Sensoren als eigenständige Busteilnehmer, wie von OSTERMEIER ET AL. (2003) [OAD03] bereits gefordert, zu definieren. Verglichen mit der vorgeschlagenen Systemarchitektur lässt sich der einen Online-Sensor repräsentierende Knotenprozessor über die Geräteklasse „*Sensor systems*“ realisieren. Bei dieser Klasse sollten wiederum zwei Funktionsklassen unterschieden werden, ein (generischer) Knotenprozessor „*internal sensor system*“ und ein Knotenprozessor „*external distributed sensor system*“. Dabei deckt der „*internal sensor system*“-Typ die an der Traktor-Geräte-Kombination montierten Online-Sensoren ab, wohingegen der „*external distributed sensor system*“-Typ, die WSN-Netzwerke mittels seiner Gateway-Funktion in den ISOBUS integrieren hilft.

Mit diesen Grundlagen ist eine eindeutige Identifizierung im Netzwerk und der damit möglichen Systemteilnahme vorhanden. Folglich ist der Blick auf die Möglichkeiten zum Datenaustausch im mobilen System (MICS) und zwischen dem MICS und dem FMIS zu richten. Für den Datenaustausch im MICS bestehen zwei grundsätzliche Optionen. Zum einen

[31] A. a. O., S. 10-18

eignet sich die Definition der Prozessdaten-Nachrichten *process data message PGN: 51968 ($00CB00_{16}$)*) ([ISO11783-10][32]), zum anderen ließe sich eine neue PGN bzw. eine PGN-Sammlung nach dem Muster der Definitionen für die Positions- und Navigationsinformationen ([ISO11783-7][33]) neu definieren. Die Prozessdaten-Nachricht ist geeignet zum Austausch von Soll- und Istwerten, den Prozessvariablen (*process variable value*), mit nur einer CAN-Botschaft. Dies passt sehr gut zu der vorgeschlagenen Systemarchitektur mit Knotenprozessoren, die eine *Data Fusion* auf Merkmals- oder Zustandsebene vornehmen und nur die Ergebnisse in sozusagen komprimierter Form an die nächsthöheren Fusionsebenen im *„In-field Controller"* weiterleiten. Sollten sich jedoch die *JDL Level 1 Processing* -Fusionsergebnisse nicht in einer CAN-Botschaft darstellen und übertragen lassen, so führt kein Weg an einer neuen PGN-Definition vorbei. Längere zusammenhängende Datenpakete könnten mit dem ISO 11783 Transport Protokoll oder dem NMEA *fast packet protocol* wie schon bei den Positions- und Navigationsinformationen übertragen werden.

Da die abgeleitete Systemarchitektur mit der Übermittlung von Index- und weiterverarbeiteten Messwerten auskommt und zugleich die über die Prozessdaten-Nachrichten möglichen Gerätebeschreibungen (*device description data*) vorteilhaft sind, wird vorgeschlagen, den Datenaustausch innerhalb des MICS über die Prozessdaten-Nachrichten durchzuführen. Da der „*In-field Controller*" in dieser Arbeit aus den angeführten Gründen nicht mit dem *Task Controller* gleichgesetzt bzw. in ihn integriert wird, müsste für den *„In-field Controller"* ein Replikat der Protokolldefinitionen für das Zusammenspiel *Task Controller* und Prozessdatennachrichten in den Standard aufgenommen werden. Mittels der Gerätebeschreibungsdaten kann die vorliegende Online-Sensorkonfiguration, z.B. die Anzahl der Messköpfe, die Konfiguration oder die geometrischen Daten des Sensorsystems, beschrieben werden. Im Mittelpunkt des Datenaustausches stehen die Prozessvariablen (*process variable value),* die Soll-, Ist- und Statuswerte. Während sich die Istwerte als reine Sensormesswerte und abgeleitete Indexwerte (z.B. der Vegetationsindex REIP) direkt erschließen, bedarf es bei den Sollwerten im Zusammenhang mit einem Sensor, Sensor-System bzw. einem Knotenprozessor eines zweiten Blickes. Im Rahmen des *JDL Level 4 Processing* sind auch Sollwerte für die Sensorsystemkonfiguration (z.B. *sensor cueing*) nötig. Auch gehen derzeit am Markt erhältliche Online-Sensorsysteme über die vorgeschlagene Merkmals-/Zustandsfusion hinaus und agieren bereits als herstellerspezifischer „*In-field*

[32] A. a. O., S. 36-47

[33] A. a. O., S. 104

Controller" auf Entscheidungsebene, d.h. sie nehmen die Berechnung eines Applikationssollwertes vor, den sie direkt an den Düngerstreuer als Sollwert kommandieren.
Ein weiterer Vorteil der Nutzung der Prozessdaten-Nachrichten liegt darin, dass sie beim Datenaustausch zwischen MICS und FMIS bereits sehr gut unterstützt werden. Die zugehörigen Festlegungen sind in Teil 11 der Norm „*Part 11: Mobile data element dictionary*" [ISO11783-11] definiert und als Datenbank im Internet auf http://www.isobus.net öffentlich bereitgestellt. Die Datenbank wird stetig weiterentwickelt. Hierzu werden neue Vorschläge für Datenelemente bei einem zugeordneten internationalen ISO-Expertengremium zur Evaluierung eingereicht und bei positivem Bescheid in die Datenbank übernommen. Für diese Arbeit fand ein kompletter Auszug dieser Datenbank zum Stand vom 31.08.2011 Anwendung.
Über die Definition der Datenbankelemente (*data dictionary entity*) sind die in den Prozessdaten-Nachrichten übertragenen Informationen im MICS und FMIS eindeutig interpretierbar. Über eine 16-Bit-Nummer, *data dictionary identifier (*DDI*)*, wird das Datenbankelement sowohl in der Datenbank als auch in der Prozessdaten-Nachricht eindeutig identifiziert. Im Folgenden werden die beiden Online-Sensor-Typen „Pflanze" und „Boden" sowie ein Typ „Wetter" unter Nutzung bestehender DDI's und neu geforderter DDI's spezifiziert. Dabei orientiert sich die Vorgehensweise an dem nachfolgend beschriebenen Schema. Zuerst wird die Geräteklasse (*Device* (*DVC)*)[34] ausgewählt. Dies ist vorrangig der Typ „*Sensor systems*", könnte aber auch in fester Zuordnung einer für Applikationstätigkeit typischen Geräteklasse (*Fertilizers* (Düngerstreuer)*, Sprayers* (Pflanzenschutz)*, Irrigation* (Beregnung) definiert werden. Dann wird das *Device Element* (*DET*)[35] spezifiziert, d.h. z.B. Online-Sensor „Pflanze". Dieser lässt sich wiederum in interner (*internal*) oder externer (*external*) Knotenprozessor unterscheiden, d.h. z.B. Online-Sensor „Pflanze On-board" (Typ: *internal* Knotenprozessor) oder Online-Sensor „Pflanze WSN" (Typ: *external* Knotenprozessor). Diesem *Device Element* werden entsprechende Funktionen (*Device Element Function*) zugeordnet, die eine Menge von Merkmalen und Zuständen wie beispielsweise Vegetationsindizes repräsentieren. Diesen Funktionen werden wiederum einzelne Datenelemente, referenziert über eine eindeutige DDI, zugeordnet. Über dieses Schema ist aus dem Blickwinkel eines integrativen Sensorsystems eine durchgängige Spezifizierung der Prozessvariablen (*process variable value),* d.h. der Soll-, Ist- und Statuswerte, durchgängig möglich. Jedoch kann auch einer räumlich differenzierten

[34] Abk. nach Notation für XML-Elemente gemäß [ISO11783-10] S. 17.

Sichtweise des Sensorsystems entsprechend der Sensorkonfiguration und -geometrie Rechnung getragen werden. Dies lässt sich über eine zusätzliche Definition der dafür vorgesehenen *Device element type* - Untergruppen, *Bin*, *Section*, *Unit* und *Connector*, bewerkstelligen ([ISO11783-10][35]). Dieses aufgezeigte Schema zur Definition eines Online-Sensors ist mit den sogenannten Gerätebeschreibungen (*device description data*) ([ISO11783-10][36]) maschinenverständlich beschreibbar und zwischen MICS und FMIS austauschbar.

Gemäß dem aufgezeigten Schema wird in Tabelle 4.1 ein fahrzeuggetragener Online-Sensor „Pflanze“, wie er den getroffenen Festlegungen zum *JDL Level 1* und *Level 2 Processing* entspricht, beschrieben. Zusätzlich zu den Fähigkeiten zur Abbildung von Pflanzeneigenschaften über Vegetationsindizes und physikalische Messgrößen, wie Temperatur und Feuchte, sind Attribute zur Sensorkonfiguration und der Zustandsbeschreibung (*JDL Level 4 Processing*) mit aufgenommen. Erweitert ist diese Online-Sensor-Definition um die Möglichkeit, dass dieser Sensor bereits den reinen Sensor-Ansatz zur Applikation umsetzt und den Applikationswert bestimmen und kommandieren kann. Hierzu sind unterschiedliche Sollwerttypen (Tatsächlicher Wert, Minimum, Maximum, Defaultwert) vorgesehen. Auffällig ist, dass mit bereits in der Norm definierten Datenbankelementen[37] der Online-Sensor beschreibbar ist. Besonders für diese Arbeit fehlen jedoch weitere Vegetationsindizes, vorrangig der REIP-Wert. Daher wird angeregt, für die in den Pflanzenwissenschaften bereits etablierten Vegetationsindizes eigene DDI zu beantragen und aufzunehmen. Das vorgestellte Beispiel erhebt keinen Anspruch auf Vollständigkeit. Es zeigt vielmehr, wie ein derartiger Sensor beschreibbar und damit in den ISOBUS integrierbar ist. Falls weitere Attribute benötigt werden oder der technische Fortschritt ihre messtechnische Erfassung möglich macht, kann in gleicher Weise die Definition erweitert werden.

Tabelle 4.2 spezifiziert einen fahrzeuggetragenen Online-Sensor „Boden“, der sich rein auf physikalische Messgrößen und wenige Angaben zur Sensorkonfiguration beschränkt. Die Verfügbarkeit entsprechender Sensortechnik zur Erfassung der Bodenparameter bei Überfahrt (, d.h. online,) wird bei dieser Definition außer Acht gelassen. Bei der Angabe der wenigen Messgrößen wurden Attribute ausgewählt, die derzeit normalerweise (über ein absatzweises Verfahren) zur Kartierung verwendet werden und somit als Überlagerungskarten (*spezifisch langfristiges deklaratives Wissen*) zur Anwendung kommen. Die bodenphysikalischen

[35] A. a. O., S. 32: A.3.3-A.3.7

[36] A. a. O., S. 25-28

[37] Bereits definierte Data Dictionary-Elemente sind mit der zugehörigen DDI in der Tabelle 4.1 angegeben.

Attribute sind die (scheinbare) elektrische Leitfähigkeit (*Electrical ground conductivity (EM38)*), die Zugkraft (*Draft force*) und die Bodenfeuchte (*Soil moisture*). Bodenchemische Attribute sind nur als *DET Function* spezifiziert, die zugehörigen Datenbankelemente werden noch durch eine leere Menge repräsentiert.

Tabelle 4.1: ISOBUS-Gerätebeschreibung für einen Online-Sensor „*Plant On-board*“

DVC	**DET**	**DET Function**	**Zugeordnete DDE (DDI)**
„Sensor system“			
	-- Online-Sensor “*Plant On-board*“		
		-- Vegetation index	
			-- REIP
			-- NDVI (DDI=153)
		-- Physical Attributes	
			-- Crop Moisture (DDI=99)
			-- Crop temperature (DDI=234)
		-- Configuration	
			-- Yaw Angle (DDI=144)
			-- Roll Angle (DDI=145)
			-- Pitch Angle (DDI=146)
			-- Setpoint Working Height (DDI=61)
			-- Actual Working Height (DDI=62)
			-- Work State (DDI=141)
		-- Application	
			-- Mass Per Area Application Rate
			-- “ Setpoint (DDI=6),
			-- “ Actual (DDI=7),
			-- “ Default (DDI=8),
			-- “ Minimum (DDI=9),
			-- “ Maximum (DDI=10)

Die Eingangsinformation zur Einbeziehung des aktuellen Wetters kann durch eine fahrzeuggetragene Wetterstation oder im Feld installiertes WSN bereitgestellt werden. Eine Definition mit bereits definierten Datenelementen wird in Tabelle 4.3 für einen fahrzeuggetragenen Online-Sensor „Wetterstation“ aufgelistet. Konfigurationsdaten bleiben unberücksichtigt. Die charakterisierenden Attribute und damit die Datenbankelemente sind vom Typ Temperatur und Feuchtigkeit (*Temperature & Moisture*), Wind (*Wind*) und

Bewölkung und Sicht (*Sky cover & Visibility*).

Tabelle 4.2: ISOBUS-Gerätebeschreibung für einen Online-Sensor „*Soil On-board*“

DVC	DET	DET Function	Zugeordnete DDE (DDI)
„Sensor system“			
	-- Online-Sensor “*Soil On-board*“		
		-- Soil physical attributes	
			Electrical ground conductivity (EM38)
			Draft force
			Soil moisture
		-- Soil chemical attributes	
			…
		-- Configuration	
			Setpoint Working Height (DDI=61)
			Actual Working Height (DDI=62)
			Work State (DDI=141)

Tabelle 4.3: ISOBUS-Gerätebeschreibung für einen Online-Sensor „*Weather station On-board*“

DVC	DET	DET Function	Zugeordnete DDE (DDI)
„Sensor system“			
	Online-Sensor “*Weather station On-board*“		
		-- Temperature & Moisture	
			-- Ambient temperature (DDI=192)
			-- Delta T (DDI=224)
			-- Air Humidity (DDI=209)
		-- Wind	
			-- Wind speed (DDI=207)
			-- Wind direction (DDI=208)
		-- Sky cover & Visibility	
			-- Sky conditions (DDI=210)

4.4.4 Konkurrierender Zugriff auf Geräteressourcen

Die grundsätzliche Zielsetzung, eine offene und skalierbare ISOBUS-Lösung zur Umsetzung der drei Systemansätze für die Prozessführung bei Applikationsaufgaben zu entwerfen, hat zur Folge, dass der Vorgang der Kommandierung und Umsetzung des Applikationssollwertes neu betrachtet werden muss. Die Annahme, dass nur ein Systemteilnehmer die Bestimmung eines Sollwertes vornimmt und in einer Art Punkt zu Punkt -Verbindung mit dem Geräterechner diesen austauscht, ist so nicht mehr gültig. Neben der Bedienperson kann der Applikationswert vom *Task Controller* (Kartierungsansatz), von einem N-Online-Sensor (Sensor-Ansatz) oder von einem „*In-field Controller"* (Sensor-Ansatz mit Kartenüberlagerung) stammen. Alle vier Teilnehmer können in einem ISOBUS-System vorhanden sein und sich zur Laufzeit auch dynamisch an- und abmelden. Diese Teilnehmer können ein Konfliktszenario hervorrufen, indem sie auf die Geräterechner-Funktion (Ressource) „Ausbringung einer kommandierten Düngerstreuermenge", gemeinsam zugreifen. Ausgeführt kann diese Funktion nur in eindeutiger Weise werden, da es sich um eine exklusive Ressource handelt, die nicht unterschiedliche Werte zur gleichen Zeit annehmen kann. Die naheliegende Auflösung dieses Zugriffskonfliktes besteht darin, dass die Bedienperson vor Beginn der Aufgabenbearbeitung den Teilnehmer festlegt, der die Steuerung und Regelung durchführt. Dazu aktiviert sie über die *Virtual Terminal*-Menüs der einzelnen Teilnehmer den bevorzugten Teilnehmer und desaktiviert die anderen bzw. lässt sie desaktiviert. Alternativ ließe sich auch in der Auftragsdatei eine Prioritätsreihenfolge festlegen. Dies erfordert jedoch eine Modifikation der Task Controller Definition, der folglich mit den anderen Teilnehmern bei Auftragsbeginn die Prioritätsreihenfolge vereinbaren müsste. Durch diese statische Systemkonfiguration treten zur Laufzeit keine konkurrierenden Ressourcenzugriffe auf den Düngerstreuer auf. Soll jedoch eine weitgehend automatisierte Prozessführung mit der Bedienperson als letzte Überwachungsinstanz ermöglicht werden, so gilt es ein flexibleres Verfahren zu wählen. In seinen Arbeiten zum Thema „Gerät steuert Traktor" im ISOBUS sah sich FREIMANN (2003) [Fre03] mit den gleichen Fragestellungen konfrontiert. Er schlägt zur Auflösung derartiger Zugriffskonflikte vor, dass sie nur durch eine eindeutige, gegebenenfalls ressourcen- und konfigurationsspezifische Prioritätsstrategie und Reglerzuordnung gelöst werden kann. Vorzugsweise sollte dies im zentralen Traktorrechner (TECU) erfolgen. OSTERMEIER ET AL. (2003) [OAD03] haben einen vergleichbaren Vorschlag formuliert, nur wird hierin ein „intelligenter Geräterechner" gefordert. Die Entscheidung, welchem sollwertkommandierenden Teilnehmer die Ressource Ausbring-Düngerstreuermenge zugeordnet wird, trifft der Geräterechner. Für diesen Zweck hält er eine

Prioritätsliste vor. Die höchste Priorität wird für das Bedienmenü des Geräterechners selbst reserviert, um der Bedienperson stets einen direkten Ressourcenzugriff zu gewähren. Schließlich trägt sie auch aus Gründen der Haftung die Gesamtverantwortung über die Ausführung der Bewirtschaftungsmaßnahme. Die nächsthöhere Priorität wird dem „*In-field Controller"* zugeordnet. Dieser verfügt über die am meisten umfassende maschinelle Situationseinschätzung und fällt auf dieser Basis seine Entscheidung über den Sollwert. An dritter Stelle kommt der Online-Sensor, weil er den aktuellen Pflanzenzustand am besten beurteilen kann. An letzter Stelle wird der *Task Controller* platziert, der eine Sollwertempfehlung besitzt, die auf die aktuelle Situation am wenigsten eingeht, jedoch gut als eine Art Default-Sollwert gewertet werden kann. Zu berücksichtigen bei diesem Konzept ist, dass die Bedienperson im Arbeitsmenü des Geräterechners die Ressource nach einem Nutzereingriff auch wieder für einen automatischen Betrieb freigibt, sobald sie die automatische Prozessführung wieder für gerechtfertigt empfindet. Der *Task Controller* ist neben dem Traktorrechner und dem *Virtual Terminal* als feste Größe eines ISOBUS-Systems mittlerweile etabliert, daher obliegt ihm stets die Dokumentation der Applikationstätigkeit. Über die Funktion der Istwertabfrage kann er den aktuellen Ausbringwert beim Geräterechner erfragen. Abschließend sei noch darauf hingewiesen, dass der Geräterechner den Teilnehmern, deren Sollwertkommando nicht ausgeführt wird, eine Ablehnung der Anfrage sendet. Sie sollten daraufhin ihren Dienst nicht komplett einstellen, sondern in regelmäßigen Abständen testen, ob ihr Ressourcenzugriff wieder zugelassen wird.

5 Simulationsergebnisse der MSDF ISOBUS-Lösung für die N-Düngung

Das aufgezeigte Modell wurde als Software-Simulation mit intuitiver Interaktionsmöglichkeit aufgebaut und getestet. Der Leitgedanke für diese Simulation war geprägt von einer „*proof of concept*“-Philosophie für eine adäquate Implementierung der abgeleiteten Problemlösungsform und Systemarchitektur in Form eines Labor-Prototyps. Das Hauptaugenmerk sollte dabei auf den MSDF realisierenden Systemelementen und auf einer angepassten Entwicklungsmethodik liegen. Für nötiges interdisziplinäres Fachwissen und Feldtestdaten konnte auf die Arbeiten der IKB-Dürnast DFG-Forschergruppe zurückgegriffen werden.

Die Ergebnisse werden in diesem Kapitel dokumentiert. Zuerst wird der Stand der Technik (Kap. 5.1) für die abgeleitete Problemlösungsform vom Typ Produktionssystem beschrieben und danach die realisierte Simulation mit ihren Ergebnissen vorgestellt (vgl. Kap. 5.2 und Kap. A.4).

5.1 Expertensysteme als Systemgrundlage

In diesem Kapitel wird der theoretische Unterbau für die Simulation gelegt, bei dem Expertensysteme die tragende Rolle einnehmen. Dazu werden die theoretischen Grundlagen wissensbasierter Systeme mit dem Schwerpunkt des Teilgebietes Expertensysteme und der Spezialisierung auf Produktionssysteme dargestellt. Ausgehend von der Einordnung in die Fachdisziplin der „Künstlichen Intelligenz“ werden der Aufbau eines Expertensystems gezeigt, Methoden zur Wissenserhebung aufgeführt und eine Methode zur Entwicklung eines Expertensystems vorgestellt, die bei der Realisierung der Simulation Anwendung fand.

5.1.1 Architektur von Expertensystemen

Der Begriff „Künstliche Intelligenz (KI)“ ist eine Übersetzung des englischen Begriffs „*Artificial Intelligence* (AI)“. Nach KURBEL (1989) [Kur89] ist der Gegenstand der „Künstlichen Intelligenz“ die Erforschung intelligenten Problemlösungsverhaltens und darauf aufbauend die Entwicklung intelligenter Computersysteme. LUGER (2001) [Lug01] definiert Künstliche Intelligenz, als Zweig der Informatik, der sich mit der Automatisierung intelligenten Verhaltens befasst. Daher ist er der Überzeugung, dass die korrekten theoretischen und angewandten Prinzipien der Informatik zu Grunde gelegt werden müssen. Dazu gehören die zur Wissensrepräsentation verwendeten Datenstrukturen, die zur

Anwendung des Wissens erforderlichen Algorithmen sowie die Sprachen und Programmiertechniken, die zu deren Implementierung eingesetzt werden. Sowohl KURBEL (1989) [Kurb89] als auch LUGER (2001) [Lug01] weisen darauf hin, dass die Definition erschwert bzw. eingeschränkt wird, dadurch dass der Begriff der Intelligenz an sich nicht eindeutig definiert ist und unterschiedlich verstanden wird. Angesichts dieser Schwierigkeiten bei der Definition erachtet es KURBEL (1989) [Kur89] als sinnvoller, die Künstliche Intelligenz anhand der verwendeten Methoden und der Anwendungsgebiete abzugrenzen. Typische Methoden sind Lernen, Konnektionismus, Wissenserwerb, Schließen und Folgern, Wissensrepräsentation, Problemlösen und Planen, Kognitionsmodelle, heuristische Suche und Sprachverstehen. Diese Methoden kommen vor allem beim Verstehen natürlicher Sprache und der semantischen Modellierung, beim Spiele spielen, beim maschinellen Schließen und Beweisen von Theoremen, dem Bilderkennen und Bildverstehen, der Robotik, der Planung, den KI-Sprachen und KI-Werkzeugen, beim maschinellen Lernen und bei den Expertensystemen zum Einsatz. Für eine tiefer gehende Darstellung der einzelnen Methoden und Anwendungsgebiete werden dem interessierten Leser KI-Lehr- und Handbücher wie z.B. [BF81] oder [Lug01] empfohlen.

Mit zunehmender Forschung und praktischer Anwendung von Künstlicher Intelligenz wurde die Bedeutung des Wissens für die Leistungsfähigkeit von KI-Programmen/Systemen erkannt. So wird auch über den Bereich des Wissens eine Unterscheidung zwischen konventionellen (Software-)Systemen und wissensbasierten Systemen (*Knowledge Based Systems*) hergeleitet. Bei einem konventionellen System sind das Wissen über die Anwendung und das allgemeine Problemlösungswissen, im Programm als Datenstrukturen und Algorithmen vermengt und in starrer Struktur miteinander verbunden. Bei einem wissensbasierten System wird hingegen das anwendungsbezogene Wissen explizit vom anwendungsunabhängigen Wissen getrennt. KURBEL (1989) [Kur89] beschreibt den Grundgedanken folgendermaßen: „…. Den Kern eines wissensbasierten Systems bildet die *Wissensbasis*, in der das Fachwissen in einer geeigneten Repräsentationsform abgelegt wird. Die Auswertung der Wissensbasis und damit die Verarbeitung des Fachwissens nimmt eine gesonderte *Problemlösungs-* oder *Abarbeitungskomponente* vor; da diese vor allem Schlussfolgerungen durchführt, spricht man manchmal auch von *Inferenzmaschine*. In der Problemlösungskomponente ist das allgemeine, anwendungsunabhängige Wissen enthalten, beispielsweise die Mechanismen, wie aus dem Vorliegen bestimmter Voraussetzungen Schlüsse abgeleitet werden können. Zur Darstellung

des Wissens und für die Schlussfolgerungen werden Methoden der KI verwendet. …“[38] Expertensysteme werden in der Regel mit diesem wissensbasierten Systemansatz (*Knowledge-Based System Approach*) realisiert. In seinem Lehrbuch über „Intelligente Systeme für das Ingenieurwesen“ bezeichnet SRIRAM (1997) [Sri97] diese Expertensysteme auch als *Knowledge-Based Expert Systems.* Allgemein sind dies Software-Systeme, die menschliches Expertenwissen, das durch Ausbildung und Erfahrung erworben wurde, verwenden, um in erklärungsfähiger Form Probleme zu lösen, die normalerweise menschliche Intelligenz erfordern [Bal00]. Expertensysteme wurden vor allem in den letzten drei Jahrzehnten für verschiedenste Anwendungsfälle entwickelt und werden als eine geeignete Technik zu Lösung von Aufgaben angesehen, die nicht gut strukturiert (*ill-structured*) und normalerweise einer rein algorithmischen Lösung nicht zugänglich sind. Die dabei zur Anwendung kommenden Problemlösungsmethoden und Formen der Wissensrepräsentation werden im nächsten Kapitel 5.1.3, das von der Methodik zur Entwicklung eines Expertensystems handelt, an den jeweils zutreffenden Stellen vorgestellt. Zuvor wird jedoch noch der grundsätzliche Aufbau eines Expertensystems gezeigt.

Die Module, aus denen ein Expertensystem üblicherweise besteht, sind in Abbildung 5.1 dargestellt. Der Benutzer kommuniziert über eine Benutzerschnittstelle mit dem Expertensystem. Die verschiedenen Varianten bei der Benutzerschnittstelle erstrecken sich von einfachen textbasierten Ein- und Ausgaben, grafischen Schnittstellen bis hin zu natürlichsprachigen Schnittstellen. Die Wissensbasis bildet den Kern des Expertensystems. Im allgemeinen Teil der Wissensbasis ist das Wissen eines bestimmten Anwendungsbereiches enthalten. Im anderen Teil werden fallspezifische Daten bzw. der Kontext vorgehalten. Zu den fallspezifischen Daten gehören hauptsächlich Fakten (Tatsachen), Schlussfolgerungen und weitere relevante Informationen, wie Konfidenzmaße von Schlussfolgerungen. Die Inferenzmaschine wendet das Wissen an, um eine Lösung für die gegebene Aufgabenstellung abzuleiten. Über eine Erklärungskomponente kann dem Nutzer der gefundene Lösungsweg aufgezeigt werden. Einerseits sollen Fragen des Nutzers beantwortet werden können, wie das System bzw. aufgrund welcher Schlussfolgerungen das System auf die abgeleitete Lösung kommt. Anderseits soll dem Nutzer beantwortet werden, warum eine bestimmte Information gerade von ihm angefragt wird. Weiterhin ist bei vielen Expertensystemen ein Editor für die Wissensbasis bzw. eine Wissenserwerb-Komponente vorhanden. Diese Komponente stellt Verfahren zur Eingabe neuen Wissens bereit, ermöglicht Syntax- und

[38] A. a. O., S. 18

Konsistenzüberprüfungen nach Änderung der Wissensbasis und gestattet dem Entwickler, (jedoch gewöhnlich nicht dem Nutzer) die Fehlersuche (*Debugging*) und Leistungsbewertung im laufenden Betrieb.

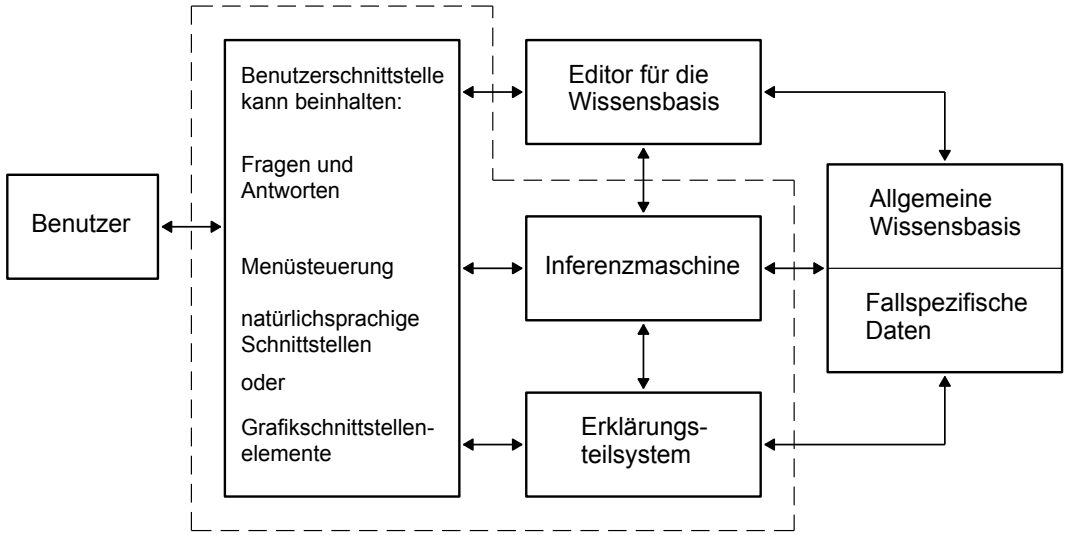


Abbildung 5.1: Architektur eines typischen Expertensystems (nach [Lug01])

5.1.2 Wissensakquisition

Da die Wissensbasis von herausragender Bedeutung für die Leistungsfähigkeit eines Expertensystems ist, nimmt der Prozess der Wissensakquisition (*knowledge acquisition*) eine sehr wichtige Rolle beim Entwicklungsprozess ein. Zunächst sollen jedoch kurz die einzelnen Personen, die mit der Entwicklung eines Expertensystems und Nutzung hauptsächlich befasst sind, vorgestellt werden. Der Wissensingenieur (*knowledge engineer*) ist der KI-Experte für die Wissensrepräsentation und die Implementierung von KI-Systemen. Er ist dafür zuständig, das Wissen des Aufgabengebietes zu ermitteln, aufzubereiten und zu strukturieren, um es dann mit Hilfe von KI-Werkzeugen oder Programmiersprachen als Expertensystem zu implementieren. Dabei hat der Wissensingenieur zu Entwicklungsbeginn oftmals keinerlei Kenntnisse über das Aufgabengebiet. Der Fachexperte (*domain expert*), hingegen ist die Person, deren Problemlösungswissen über das Anwendungsgebiet in dem Expertensystem nachgebildet werden soll. Seine Aufgabe ist es, dem Wissensingenieur dieses Wissen zugänglich zu machen. Der Endnutzer wiederum ist die Person, die das Expertensystem in der praktischen Anwendung einsetzen soll bzw. wird. Daher dürfen seine Anforderungen und Ansprüche während des Entwicklungsprozesses nicht unberücksichtigt bleiben.

Laut KURBEL (1989) [Kur89] bezeichnet Wissensakquisition den Vorgang des Wissenserwerbs durch ein wissensbasiertes System. Das Anwendungswissen muss den

Wissensquellen entnommen, transformiert und in die Wissensbasis des Systems übertragen werden. Bei dieser Begriffsdefinition wird der Vorgang hauptsächlich aus der Sicht des Expertensystems betrachtet. Wird dabei jedoch die Tätigkeit des Wissensingenieurs in den Mittelpunkt gestellt, so hat sich der Begriff *Knowledge Engineering* etabliert. Dabei gilt es aber zu beachten, dass die Implementierungsaufgaben des Wissensingenieurs nicht zur Wissensakquisition zählen.

Hinsichtlich der Methoden der Wissensakquisition reicht die Spanne gemäß MCGRAW und HARBISON-BRIGGS (1989) [MH89] von manuellen bis hin zu vollständig automatisierten Techniken. Sie unterscheiden dazu neun verschiedene Arten, denen prinzipiell folgende drei Grundformen zugrunde liegen:

- Indirekte Wissensakquisition:
  Das Wissen wird durch einen Wissensingenieur aus einem Fachexperten oder Fachtexten (z.B. Lehrbücher, wissenschaftliche Artikel, Vorschriften, Verfahrensanweisungen, Betriebs- oder Reparaturanleitungen) „extrahiert“ und in die Wissensbasis eingebaut. Da dies mit erheblichem Aufwand sowohl für den Experten als auch den Wissensingenieur verbunden sein kann, findet sich in der Fachliteratur auch die Umschreibung *knowledge engineering bottleneck* für diesen Engpass.
- Direkte Wissensakquisition:
  Erfolgt der Wissenserwerb durch das Expertensystem ohne das Einschalten eines Wissensingenieurs, wird von direkter Wissensakquisition gesprochen. Dies erfordert jedoch, dass das Expertensystem über eine intelligente, leistungsfähige Akquisitionskomponente verfügt, die direkt mit dem Experten, der in der Regel kein KI-Fachmann ist, kommunizieren kann. Nach KURBEL (1989) [Kur89] eignet sich daher die direkte Form normalerweise nicht für die erstmalige Erstellung einer Wissensbasis, sondern eher für die Pflege und Erweiterung der Wissensbasis beim späteren Einsatz.
- Automatische Wissensakquisition:
  Bei dieser Grundform erwirbt das Expertensystem sein Wissen ohne die Unterstützung eines Wissensingenieurs oder eines Fachexperten. Ansätze für diese sicherlich anspruchsvollste Form stellt SRIRAM (1997) [Sri97] in dem Kapitel „*Automated Tools for Knowledge Acquisition*“ zusammen. Die methodischen Ansätze entstammen dabei weitgehend dem KI-(Forschungs-)Bereich des maschinellen Lernens (*machine learning*).

5.1.3 Evolutionärer Entwicklungsprozess

Bei der Entwicklung eines Expertensystems ist es erforderlich, recht früh Prototypen zu erstellen, denn Code schrittweise auszuarbeiten und zu optimieren [Lug01]. Diese vom herkömmlichen Entwicklungszyklus abweichende explorative Vorgehensweise ist für viele KI-Anwendungen typisch. KURBEL (1989) [Kur89] wählt im Gegensatz zu (LUGER, 2001) [Lug01] nicht die Bezeichnung explorativ sondern evolutionär und schlägt für diesen

explorativen bzw. evolutionären Entwicklungsprozess ein Schema[39] vor, das in Abbildung 5.2 wiedergegeben wird. Entsprechend dem evolutionären Charakter weist Kurbel darauf hin, dass das Schema nicht dahingehend interpretiert werden soll, dass die Aufgabenkomplexe Phasen darstellen, die nacheinander ablaufen müssen. Hingegen kann grundsätzlich von jedem Aufgabenbereich zu jedem anderen verzweigt werden. Es ist davon auszugehen, dass zumindest mit der Identifikation begonnen und mit dem Testen abgeschlossen wird.

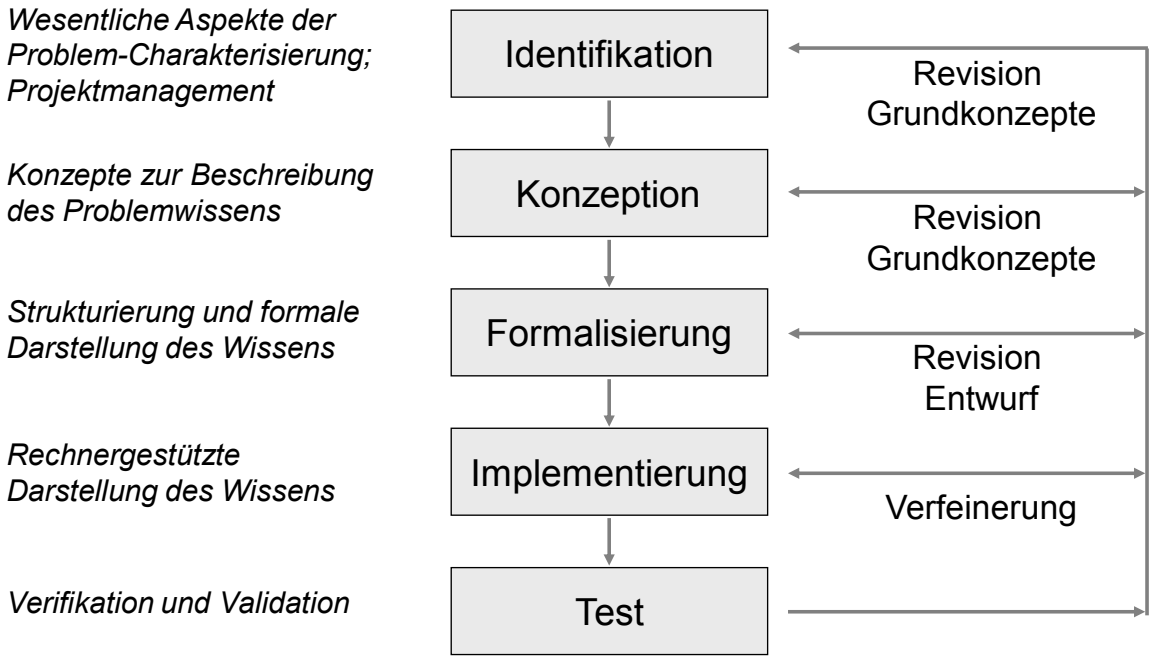


Abbildung 5.2: Evolutionärer Expertensystem-Entwicklungsprozess (abgeändert nach [Kur89])

In der Phase bzw. dem Aufgabenkomplex Identifikation gilt es die prinzipiellen Aspekte der Aufgabenstellung und des Projektmanagements zu klären. So sind die Charakteristika des Problems oder der Aufgabenstellung zu identifizieren. Fragen nach Art des Problem- und Aufgabenbereiches, Umfang des benötigten Wissens, nach Durchführungshäufigkeit, dem damit verbundenen Aufwand, der nötigen Echtzeitanforderung oder der nötigen Genauigkeit sind dabei hilfreich. Ein weiterer wichtiger Unterpunkt ist die Zielidentifizierung. Dabei ist zu definieren, was mit dem Expertensystem erzielt werden soll. Soll beispielsweise knappes Expertenwissen von fachlich weniger qualifizierten Mitarbeitern angewendet und damit breiter eingesetzt werden können, soll es einen Experten unterstützten oder aber Teil einer Automatisierungslösung sein. Hinsichtlich des Projektmanagements sind die am Projekt Beteiligten zu identifizieren und ihre Rollen festzulegen. Es ist darüber zu entscheiden, ob es

[39] Dieses Schema orientiert sich vorrangig an der indirekten Wissensakquisition.

sich um eine Eigenentwicklung oder Auftragsentwicklung durch Dritte handeln soll. Schließlich sind auch die benötigten bzw. zur Verfügung stehenden Ressourcen zu beachten. Hierbei handelt es sich in erster Linie um die verfügbaren Wissensquellen, die verfügbare Zeit und KI-Werkzeuge (Hard- und Software).

Der nächste Aufgabenkomplex, die Konzeption, ist gekennzeichnet durch die Fragestellung, welche grundlegenden Konzepte die Problemlösung ermöglichen und welche Informationen und Techniken der Fachexperte hierzu verwendet. Teilaufgaben, Strategien und Einschränkungen beim Problemlösungsprozess sind zu analysieren. Der Wissensingenieur und der Fachexperte untersuchen, welche Daten verfügbar sind, was abgeleitet werden muss, mit welchem Detaillierungsgrad das Wissens erfasst und repräsentiert werden soll.

In der Phase der Formalisierung sollen die in der Konzeptionsphase erarbeiteten Grundkonzepte in eine formale Darstellung überführt werden. Dies erfordert die Festlegung auf geeignete Wissensrepräsentationsformen und Abarbeitungsstrategien (Inferenz-Mechanismen). Als Ergebnis der Formalisierung liegt die Struktur des Expertensystem-Designs vor und die Repräsentationsform des Wissens, d.h. das grundlegende Design der Wissensbasis, wurde definiert. Weiterhin wird in der Formalisierung die Auswahl eines passenden Entwicklungswerkzeuges getroffen. Dabei spielen Aspekte wie Möglichkeiten zur Wissensrepräsentation, Inferenz-Mechanismen, Bedienbarkeit, Hardwareanforderungen, Leistungsfähigkeit, Erweiterbarkeit, Wartungsfreundlichkeit oder Kosten eine große Rolle.

Während der Implementierung werden der formale Systementwurf der Wissensbasis und der Inferenz-Mechanismen sowie der weiteren Komponenten eines Expertensystem in ein technisches System, d.h. vor allem in Software und entsprechende Hardware, umgesetzt. Getreu dem explorativen Entwicklungskonzept wird dabei schnell die Implementierung eines Prototypen angestrebt, der entweder sukzessive überarbeitet und bis zum Endprodukt verbessert oder aber verworfen wird, falls sich die Ergebnisse der Konzeption und oder der Formalisierung nicht bewähren.

Diese Eignung und Leistungsfähigkeit lässt sich in der Phase bzw. dem Aufgabenkomplex des Testens überprüfen. Ziel des Testens ist wie auch bei konventioneller Softwareentwicklung die Fehler und Schwächen des Systems aufzudecken und die Leistungsfähigkeit zu bestimmen bzw. nachzuweisen. Fehler und mangelhafte Leistungsfähigkeit können unterschiedliche Ursachen besitzen. Abgesehen von Implementierungsfehlern können sich die Wissensrepräsentationsformen und die Inferenz-Mechanismen als doch nicht geeignet erweisen. Der Detaillierungsgrad des Wissens könnte nicht ausreichend sein oder das akquirierte Wissen ist von sich aus falsch oder inkonsistent. Grundsätzlich ist das Auftreten

von Mängeln nicht ausschließlich negativ zu sehen. So weist LUGER (2001) [Lug01] darauf hin, dass Expertensysteme durch fortschreitende Annäherung an das fertige System erstellt werden, wobei Fehler des Systems zu Korrekturen oder Erweiterungen der Wissensbasis führen. In diesem Sinne wächst die Wissensbasis, sie wird nicht als Gesamtheit entworfen. Testen lässt sich in der Regel in Verifikation und Validation unterscheiden. Unter Verifikation wird die Überprüfung der Übereinstimmung zwischen einem Software-Produkt und seiner Spezifikation verstanden, während unter Validation die Eignung bzw. der Wert eines Produktes bezogen auf seinen Einsatzzweck verstanden wird [Bal98]). Laut SRIRAM (1997) [Sri97] ist bei der Expertensystementwicklung die Verifikation problembehaftet, da die Softwarespezifikation eher vage oder nicht existent ist oder sich mit den evolutionären Entwicklungsschritten beständig ändert. So liegt das Hauptaugenmerk auf der Validation. Hierfür führt er drei Testrichtungen auf. Eine davon ist die Beurteilung der informationstechnischen Eignung und Leistung (Zeit, Speicherbedarf, Robustheit, Konsistenz, Vollständigkeit, Portierbarkeit und Erweiterbarkeit). Eine weitere Dimension ist die psychologische Eignung, ausgedrückt in einfacher Bedienbarkeit, guter Erlernbarkeit und Verständlichkeit (der Erklärungskomponente). Die dritte Testrichtung betrifft die Leistungsfähigkeit bezüglich der Problemlösung (Genauigkeit, Präzision, Zuverlässigkeit, Wiederholbarkeit, Breite und Tiefe des Lösungsspektrums).

Es wurde bereits am Kapitelanfang darauf hingewiesen, dass die aneinandergereihte Darstellung der einzelnen Aufgabenbereiche keinen strengen linearen Phasenablauf widerspiegelt. So unterscheidet KURBEL (1989) [Kur89] auch drei Formen der Rückkopplung wie in Abbildung 5.2 dargestellt. Bei der Verfeinerung zwischen Implementierung und Testen werden Elemente der Wissensbasis und Kontrollstrategien der Inferenz-Mechanismen abgeändert und angepasst. Bei der Revision des Entwurfs geht es um Nachbesserung, Erweiterung oder Ersatz der gewählten Wissensrepräsentationsformen und Inferenz-Mechanismen. Bei Revision der Grundkonzepte sind Änderungen auf der Ebene der Konzeption oder der Identifikation durchzuführen. Dies hat z.B. zur Folge, dass andere Wissensquellen mit aufgenommen oder die Teilaufgaben anderes strukturiert werden.

5.1.3.1 Konzeption und Formalisierung – Ausgewählte Methoden

Der vorgestellte evolutionäre Prozess ist etablierter Stand der Technik bei der Entwicklung von Expertensystemen. Verglichen mit konventionellen Entwicklungsprozessen des Ingenieurwesens (vgl. Kap. 2.5) erscheint er möglicherweise etwas unstrukturiert, indirekt und wenig effizient. Diese Einschätzung lässt sich für einen Teil der Identifikation, der

Konzeption und der Formalisierung abmildern, indem folgender methodischer Vorschlag von SRIRAM (1997) [Sri97] herangezogen wird. Er schlägt vor die Entwicklung von *Knowledge-Based Expert Systems* als eine Transformation von einem *„Specification Level“,* über einen *„Task Level“*, einen *„Problem Solving Level“* und einen *„Knowledge-base Level“* hin zu einem *„Tool Level“* zu betrachten. Diese Transformation ist in Abbildung 5.3 bildlich dargestellt.

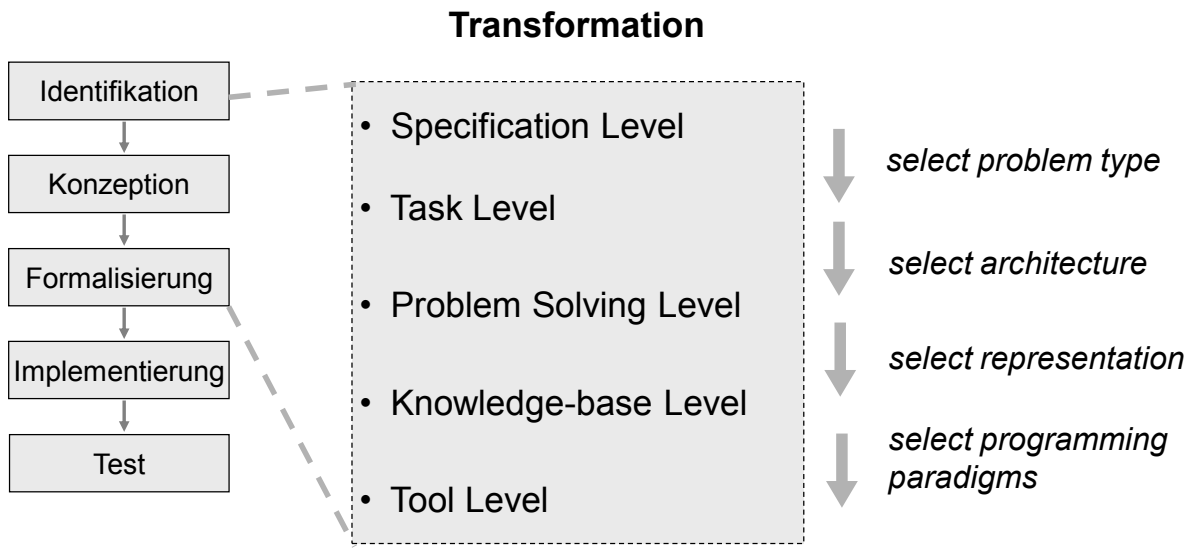


Abbildung 5.3: Ebenen zur Beschreibung eines *Knowledge-Based Expert System* (abgeändert nach [Sri97])

Eingeordnet in den übergeordneten explorativen Entwicklungsprozess kann diese Transformation hauptsächlich der gesamten Konzeption und Formalisierung zugerechnet werden und sie überdeckt auch noch den Teilbereich der Identifikation hinsichtlich der Problem- bzw. Aufgabendomänenidentifizierung. In der Abbildung 5.3 wird die Transformationsrichtung verknüpft mit den konkreten Aufgabenschritten wiedergegeben, die im folgenden Beschreibungs-*Level* zu erarbeiten sind.
Die einzelnen Beschreibungs-*Levels* werden in den folgenden Abschnitten näher erläutert. Dabei werden auswählte Methoden bzw. Techniken der einzelnen Ebenen näher dargestellt, falls sie für die Realisierung der Simulation von besonderer Bedeutung sind.

Specification Level:
Die Definition dieser Ebene deckt sich mit dem Teilbereich der Identifikation, der sich mit den Charakteristika des Problems oder der Aufgabenstellung beschäftigt. Die Aufgabenbereiche der Identifikation, die sich mit den Belangen des Projektmanagements befassen, sind nicht Teil dieser Ebene.

Task Level:

Diese Ebene beschreibt Problem- und Aufgabenstellungen des Ingenieurwesens, die durch ein Expertensystem bewältigt werden sollen bzw. können. Die Spanne dieses Anwendungsbereichs für Expertensysteme reicht laut SRIRAM (1997) [Sri97] von eher derivativen (*derivation*) hin zu formativen (*formation*) Problemstellungen. Dieses Spektrum derivativer-formativer-Ausrichtung für grundsätzliche Problemstellungen des Ingenieurwesens, das im Ursprung auf [WHL83] zurückgeht, zeigt Abbildung 5.4.

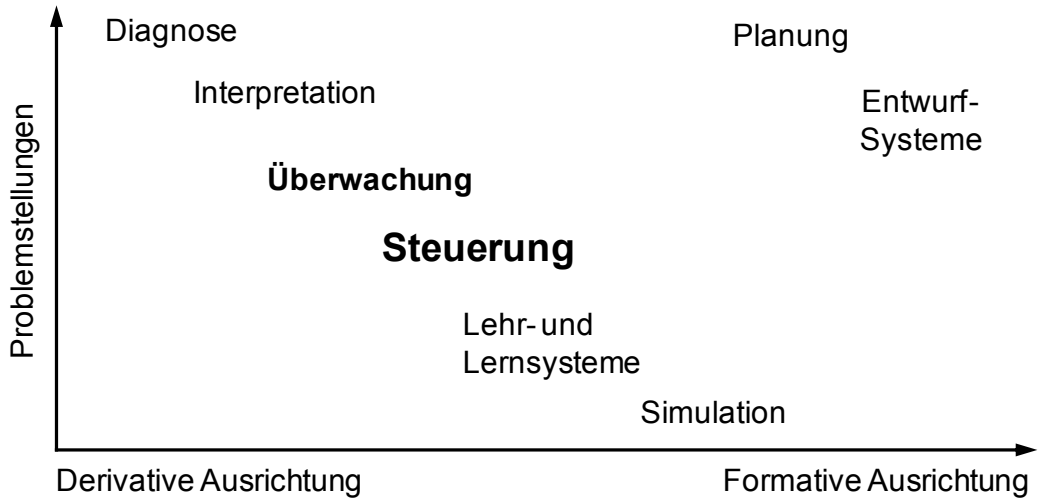


Abbildung 5.4: Derivatives-formatives Spektrum von Problemstellungen [WHL83]

In derivativen Aufgabenstellungen sind die *problem conditions* bereits als Teile einer Lösungsbeschreibung spezifiziert, d.h. mögliche Ergebnisse sind in der Wissensbasis existent. So erstreckt sich die Problemlösung weitgehend auf die Identifizierung des geeigneten Lösungsweges. Bei formativen Aufgabenstellungen hingegen existiert die (exakte) Lösung normalerweise nicht in der Wissensbasis, jedoch kann der Inferenzmechanismus Lösungen mithilfe des Wissens der Wissensbasis erzeugen. Praktische Probleme weisen meist Teile von beiden Ausrichtungen auf.

Diagnose, Interpretation, Überwachung und Steuerung sind eher derivativ geprägt. Lehr- bzw. Lern- oder Unterrichtssysteme und Simulation weisen deutliche Charakteristika beider Ausprägungen auf. Dem formativen Zweig sind vor allem die Planung und die Entwurfssysteme zuzuordnen. Seit den Anfängen der Expertensystem-Entwicklung in den sechziger Jahren des vergangenen Jahrhunderts wurden für all diese Anwendungsfälle bereits Expertensysteme entwickelt. Auflistungen bekannter Expertensysteme sowie die historische Entwicklung der Expertensystemtechnik können u. a. bei den bereits vielfach zitierten

Autoren [Lug01], [Kur89], [Sri97, [BF81], [WHL83] gefunden werden. Mit Fokus auf realisierte Expertensysteme im Bereich der landtechnischen und teilflächenspezifischen Anwendung hat ROGGE (2004) [Rog04] eine Literaturrecherche in europäischen und amerikanischen Fachzeitschriften durchgeführt. Er hat dabei festgestellt, dass die meisten Arbeiten hierzu in den späten Achtzigerjahren und der ersten Hälfte der Neunzigerjahre des vergangenen Jahrhunderts durchgeführt wurden. Die Spanne der Anwendung erstreckt sich von Beratungssystemen für das Bestandes- oder Herdenmanagement, Anbauplanungssystemen, Pflanzenschutzempfehlung, Diagnosesysteme zur Krankheitsbestimmung oder zur Beikrautbestimmung, Gewächshaus- und Beregnungsanlagen-Prozesssteuerungen, Düngeberatungs-Systemen bis hin zu Marketing-Beratungsinstrumentarien. Beispiele zur Anwendung von Expertensystemen im Bewässerungsmanagement sowohl als Planungs-, Analyse- und Vorhersagewerkzeug als auch zum Einsatz im Feld zu Steuerungszwecken haben ZAZUETA ET AL. (2006) [ZXPM06] zusammengestellt. In jüngerer Zeit wurde Expertensystemtechnik für den Anwendungsbereich Teleservice bei Landmaschinen im Rahmen einer Forschungsarbeit erprobt [Föl06] sowie zur Simulation einer Real-time Prozessführung in sensorgestützten Düngesystemen genutzt [ORA06].

Problem Solving Level:

Auf dieser Ebene wird die Auswahl der einzusetzenden Problemlösungsverfahren getroffen. SRIRAM (1997) [Sri97] unterscheidet zwei grundsätzliche Klassen von Verfahren. Die eine Klasse wird als suchbasiert (*search-centered*), die andere Klasse als wissensbasiert (*knowledge-centered*) umschrieben. Suchbasierte Techniken werden im Allgemeinen als sehr generelle Lösungsverfahren angesehen, jedoch tendieren sie zu einer exponentiellen Ausweitung des Suchraumes bei Zunahme der Aufgabenkomplexität. Daher werden sie in der Literatur oftmals auch als „schwache“ Problemlösungsmethoden bezeichnet. Im Gegensatz hierzu werden wissensbasierte Verfahren als „starke“ Problemlösungsmethoden eingestuft. Sie verwenden gezielt domänenspezifisches Wissen, um zu einer Aufgabenlösung zu gelangen. Es soll jedoch nicht unerwähnt bleiben, dass suchbasierte Verfahren implizit von wissensbasierten Techniken genutzt werden. Typische Vertreter suchbasierter Problemlösungsmethoden sind:

- Tiefen- und Breitensuche (*Depth-first*, *Breadth-first*),
- „*Hill-Climbing*“-Strategien,
- Bestensuche *(Best-first*),

- A*-Algorithmus,
- Minimax-Verfahren (aus Spieltheorie),
- Alpha-Beta-Verfahren (aus Spieltheorie).

Charakteristisch für den Bereich der wissensbasierten Methoden und Ansätze sind:

- Vorwärts- und Rückwärtsverkettung (*Forward/Backward Chaining*),
- Theorembeweise,
- „*General Problem Solver*“ (mit Mittel-Ziel-Analyse (*means-ends analysis*)),
- Produktionssystem,
- Modellbasiertes Schließen,
- Fallbasiertes Schließen,
- Schließen in unsicherer Situation (approximatives oder nicht-exaktes Schließen) über Konfidenzfaktor-Algebra, unscharfe Mengen (*„Fuzzy Theorie“*), Bayessches Schließen oder Dempster-Shafer-Theorie,
- Hierarchische Problemzerlegung (*Hierarchical Refinement*).

Speziell die wissensbasierten Verfahren stehen in engen Zusammenhang mit der jeweils nötigen bzw. zugrundeliegenden Wissensrepräsentationsform, die auf Ebene des *Knowledge-base Level* thematisiert wird. Die aufgeführten Methoden geben vor allem klassische Methoden wider. Alternative und teilweise neuere Ansätze, sieht LUGER (2001) [Lug01] in agentenbasiertem und verteiltem Problemlösen und Formen von parallelen und emergenten Rechnen. Da in der vorliegenden Arbeit Produktionssysteme eine tragende Rolle einnehmen, wird darauf vertieft eingegangen.

Ein Produktionssystem ermöglicht die musterorientierte Steuerung eines Problemlösungsprozesses und besteht aus einer Menge von Produktionsregeln, einem Arbeitsspeicher und einem „Erkennen-Handeln-Steuerungszyklus“ (*recognize-act-control-cycle*) Ein Produktionssystem ist die Grundlage eines regelbasierten Expertensystems [Bal00] und verweist somit auch auf die zugrundeliegende Wissensrepräsentation in Form von Produktionsregeln. Die nachfolgende ausführlichere Definition eines Produktionssystems folgt den Darstellungen von [Lug01][40], [Bal00][41].

Ein Produktionssystem wird definiert durch:

- Menge der *Produktionsregeln*,
- *Arbeitsspeicher,*
- „*Erkennen-Handeln-Zyklus“*.

[40] A. a. O., S. 197-200, 206-213

[41] A. a. O., S. 295-296, 309-310

Menge der *Produktionsregeln*:

Eine Produktionsregel, oftmals einfach Produktion oder Regeln genannt, besteht aus einer Vorbedingung und einer Aktion. Die Vorbedingung beschreibt eine Situation, in der der Aktionsteil ausgeführt werden soll. Die Aktion ist entweder eine Implikation oder Deduktion zur Herleitung des Wahrheitsgehaltes einer Feststellung, oder aber eine Handlung, mit der ein Zustand verändert wird.

Arbeitsspeicher:

Er enthält die gültigen Fakten bzw. eine Beschreibung des aktuellen Zustands der Umgebung in einem Schlussfolgerungsprozess. Diese Fakten werden mit dem Bedingungsteil der Produktionsregeln verglichen. Stimmt der Bedingungsteil einer Regel mit dem Inhalt des Arbeitsspeichers überein, dann kann die mit dieser Bedingung verknüpfte Aktion ausgeführt werden. Aktionen sind darauf ausgelegt, den Inhalt des Arbeitsspeichers zu ändern.

Der „*Erkennen-Handeln-Zyklus"*:

Der Arbeitsspeicher wird mit dem Beginn der Problembeschreibung initialisiert. Der aktuelle Status des Problemlösungsprozesses wird durch die Fakten im Arbeitsspeicher dargestellt. Ein Vergleich der Fakten mit dem Bedingungsteil der Regel ergibt eine Teilmenge der Produktionsregeln, die sogenannte Konfliktmenge, deren Bedingungen den Fakten entsprechen. Die Regeln der Konfliktmenge werden als anwendbar bezeichnet. Im nächsten Schritt wird eine Regel dieser Konfliktmenge ausgewählt und angewendet, d.h. die Regel „feuert". Dabei wird ihr Aktionsteil ausgeführt und somit der Inhalt des Arbeitsspeichers geändert. Nach dem „Feuern" der ausgewählten Regel wird der Steuerungszyklus mit dem veränderten Arbeitsspeicher wiederholt. Das Ende des Prozesses wird erreicht, wenn der Inhalt des Arbeitsspeichers mit keiner Regelbedingung mehr übereinstimmt.

Der beschriebene „*Erkennen-Handeln-Zyklus"* folgt der Strategie der Vorwärtsverkettung (*forward chaining*). Hinsichtlich der Auswahl einer Regel aus der Konfliktmenge sind verschiedene wichtige Konfliktlösungsstrategien bekannt:

- Auswahl nach der Reihenfolge oder Aktualität (d.h. die erste anwendbare Regel feuert bzw. die Regel, deren Vorbedingung sich auf möglichst neue Einträge in der Datenbasis bezieht),
- Auswahl nach syntaktischer Struktur der Regeln (z.B. die spezifischste Regel feuert, d.h. die Regel, deren Vorbedingung die einer anderen Regel und zusätzlich noch weitere Aussagen enthält),
- Auswahl mittels Zusatzwissens (z.B. die Regel mit der höchsten Priorität feuert, vorausgesetzt Regeln werden Prioritäten zugeordnet, oder zusätzliche Regeln, sogenannte Meta-Regeln, steuern den Auswahlprozess).

Neben der Strategie der Vorwärtsverkettung kann die Steuerung der Suche in einem Produktionssystem auch dem Prinzip der Rückwärtsverkettung (*backward chaining*) folgen. Dabei wird nach einer Übereinstimmung zwischen dem Ziel und dem Aktionsteil der Produktionsregeln gesucht und der Bedingungsteil dann als Teilziel eingesetzt, dessen „Richtigkeit“ es zu beweisen gilt.
Eine weitere Möglichkeit zur Steuerung der Problemlösung besteht in der Strukturierung von Regeln. Eine größere Regelmenge lässt sich in Module mit zusammengehörenden Regeln aufteilen. Zu den einzelnen Prozessschritten ist dann jeweils nur ein Modul aktiv und ein Wechsel zwischen den einzelnen Modulen erfolgt in der Regel über einen Fokuswechsel-Mechanismus.
Ein wichtiger Gesichtspunkt des Produktionssystemmodells ergibt sich aus dem Fehlen syntaktischer Interaktionen zwischen den Produktionsregeln. Regeln können die Anwendung anderer Regeln nur beeinflussen, indem sie Muster/Fakten im Arbeitsspeicher ändern und nicht indem sie andere Regeln direkt wie eine Unterroutine aufrufen können. Diese syntaktische Unabhängigkeit begünstigt die evolutionäre und inkrementelle Entwicklung eines Expertensystems. Jedoch weist KURBEL (1989) [Kur89] darauf hin, dass der Wissensingenieur manchmal dazu gezwungen wird, Kontroll- und Abarbeitungswissen in Regeln zu codieren, wenn die Implementierungswerkzeuge keine anderen Möglichkeiten vorsehen, die Abarbeitungsstrategie zu beeinflussen. Zu diesem Zweck werden Regeln und Fakten zur Ablaufsteuerung angewandt, was vom theoretischen Ansatz des Produktionssystems einen missbräuchlichen Einsatz darstellt und dem Paradigma des wissensbasierten Ansatzes widerspricht.

Knowledge-base Level:
Auf dieser Ebene werden in enger Abstimmung mit den Techniken des *Problem Solving Levels* geeignete Formen zur Repräsentation des Wissens in der Wissensbasis ausgewählt. KURBEL (1989) [Kur89] unterscheidet zwei unterschiedliche Ausrichtungen von Wissensrepräsentationen, die deklarative und die prozedurale Repräsentationsform. Deklarativ wird eine Wissensrepräsentation bezeichnet, wenn sie sich auf die reine Beschreibung von Sachverhalten beschränkt und keine Angaben über die Anwendung des Wissens zur Lösung eines konkreten Problems enthält. Bei der prozeduralen Wissensrepräsentation liegt das Hauptaugenmerk auf dem aktiven Gebrauch des Wissens. Für die Darstellung werden bereits Angaben benutzt, wie das Wissen angewendet bzw. aufgebaut werden soll. Beide Repräsentationsformen schließen sich nicht gegenseitig aus, sondern

kommen bei den bekannten Wissensrepräsentationsformen meist gemeinsam, jedoch in unterschiedlich starker Gewichtung vor. Laut LUGER (2001) [Lug01] basieren die traditionellen KI-Repräsentationsmethoden auf der *physischen Symbolsystemtheorie*, die von NEWELL UND SIMON (1976) [NS76] formuliert wurde. Ausgehend von dieser Theorie wurden unter anderem folgenden Repräsentationsformen für KI-Einsatzzwecke entwickelt oder herangezogen:

- Logik,
- Semantische Netze,
- Skripte,
- Frames,
- Objektorientierte Wissensrepräsentation,
- Regelbasierte Wissensrepräsentation.

Auf der Basis von mathematischer Logik kann Wissensrepräsentation realisiert werden. In der Regel kommt dabei die Prädikatenlogik 1. Ordnung zum Einsatz, die Variable, Konstante, Funktionen und Prädikate kennt, die mit Verknüpfungsoperatoren zu Ausdrücken verbunden werden. Semantische Netze stellen Wissen in Form eines Graphen dar, der aus einer Menge von Knoten und Kanten besteht, wobei die Knoten Objekte, Ereignisse, Konzepte, abstrakte Begriffe oder andere Sachverhalte darstellen. Die Kanten beschreiben beliebige Beziehungen zwischen den Knoten. Vor allem hierarchische Beziehungen finden Anwendung, die mit einem Vererbungsmechanismus verbunden sind. Ein Skript ist eine strukturierte Repräsentation zur Beschreibung stereotypischer Abläufe in einem bestimmten Kontext [Lug01]. Skripte werden in Systemen zur Verarbeitung von natürlicher Sprache für den Aufbau der Wissensbasis anhand von Situation angewandt, die das System verstehen können soll. Ein den Skripten ähnliches Repräsentationsschema sind *Frames* (MINSKY, 1975). *Frames* stellen die impliziten Verknüpfungen zwischen Information in einer Domäne durch Datenstrukturen dar. Laut Minsky kann ein Frame als statische Datenstruktur angesehen werden, womit stereotype Situationen dargestellt werden können. Die objektorientierte Repräsentationsform wurde nicht im Rahmen der KI-Forschung entwickelt, hat jedoch auch im Bereich der Expertensysteme Anwendung gefunden. KURBEL (1989) [Kur89] zufolge kann die objektorientierte Wissensrepräsentation als eine Verallgemeinerung des Konzepts der abstrakten Datentypen angesehen werden. Objekte sind streng gekapselt, verfügen über Eigenschaften, tauschen mit anderen Objekte Nachrichten aus, die wiederum beim Empfänger Methoden aktivieren. Grundlegend für die objektorientierte Repräsentationsform sind die Konzepte der Klassen, Instanzen sowie der Klassen- und -Vererbungshierarchien. Eine in

Expertensystemen weitverbreitete Wissensrepräsentationsform ist die Darstellung von Wissen über Regeln. Bereits im vorangegangenen Abschnitt des *Problem Solving Levels* wurde die Repräsentationsform in Form einer Regel bzw. Produktionsregel vorgestellt. Von den vorgestellten Repräsentationsformen weisen die Logik und die semantischen Netze eher deklarativen Charakter auf, die regelbasierte Form ist eher dem prozeduralen Zweig zuzuordnen und Skripte, Frames und die objektorientierte Form vereinen beide Aspekte. Nach SRIRAM (1997) [Sri97] eignen sich Regeln und Logik, um Heuristiken für Problemstellungen des Ingenieurwesens zu kodieren, während Frames, semantische Netzwerke oder die objektorientierte Repräsentationsform Formalismen bereitstellen, um modellbasierte und strukturierte Repräsentationsformen für die unterschiedlichen technischen Entitäten (Objekte) zur Verfügung zu stellen. Im Prinzip weisen all die aufgeführten Repräsentationsformen die Möglichkeit auf, den Aspekt der Unsicherheit (*uncertainty*) bzw. des nicht eindeutigen Wissens zu adressieren. Nach HALL UND MCMULLEN (2004) [HM04] kann dies auf zwei Wegen geschehen, entweder durch Zuweisung von Unsicherheiten zu Objekteigenschaften/Attributen oder durch die Definition der Unsicherheit von logischen Beziehungen. Für diese Zwecke werden hauptsächlich Konfidenzfaktor-Algebra, Theorie der unscharfen Mengen („Fuzzy-Mengen“), Bayessches Schließen und Bayessche Belief-Netze oder die Dempster-Shafer-Theorie der Evidenzen herangezogen. Für eine ausführliche Auseinandersetzung mit der Fragestellung der Repräsentation von unsicherem Wissen im wissensbasierten Zusammenhang kann beispielsweise auf [KC93] verwiesen werden.
Abschließend soll nicht unerwähnt bleiben, dass auch alternative Repräsentationsformen in der KI-Forschung und -anwendung verfolgt werden, die nicht auf die physische Symbolsystemhypothese zurückgehen. Ein größerer Teil dieser Formen orientieren sich an der Architektur des (tierischen) Gehirns und an Prozessen der biologischen Evolution. Ein bekannter Vertreter dieser alternativen Wissensrepräsentationsformen ist der Ansatz der Künstlichen Neuronalen Netze.

Tool Level:
Auf dieser Ebene gilt es auf der Grundlage der Ergebnisse und Anforderungen der anderen Ebenen ein geeignetes (Software)-Werkzeug (*Tool*) zur Implementierung eines Expertensystems auszuwählen. Dabei reicht die Bandbreite der Möglichkeiten und Werkzeuge von konventionellen prozeduralen Programmiersprachen bis hin zu Expertensystem-Shells [Sri97]. Traditionelle Programmiersprachen, vor allem der dritten Generation wie z.B. Ada, C, Cobol, Fortran oder Pascal, geben dem Wissensingenieur sehr

allgemeine Ausdrucksmittel und Konstrukte an die Hand, die eine große Flexibilität bieten, die benötigten Wissensrepräsentationen und Inferenz-Mechanismen zu realisieren. Anderseits erzwingt dies eine recht aufwändige und aus der *Engineering*-Perspektive recht ineffiziente Vorgehensweise, bei der ein großer Teil der Arbeit auf die Erzeugung der „Infrastruktur" verwandt und nicht dem eigentlichen Wissen und den Problemlösungsstrategien gewidmet wird. Dennoch kann es aus Gründen der Systemperformance, begrenzter (Hardware-)Ressourcen oder aufgrund von Echtzeitanforderungen erforderlich sein, ein Expertensystem auf Basis dieser prozeduralen Programmiersprachen zu implementieren.

Während diese prozeduralen Programmiersprachen stark von numerischen und zeichenbasierten Datentypen geprägt sind, spielt bei wissensbasierten Systemen die Darstellung und Verarbeitung von Symbolen eine herausragende Rolle ([Kur89], [Lug01], [Sri97]), was von den am weitesten verbreiteten KI-Sprachen LISP und PROLOG direkt unterstützt wird [Lug01]. PROLOG ist eine logische Programmiersprache und setzt auf der Prädikatenlogik erster Stufe auf. LISP wurde bereits in den späten 50er Jahren des 20. Jahrhunderts von John McCarthy mit dem Ziel entworfen, eine Sprache für die symbolische statt der numerischen Berechnung zu entwickeln [Lug01]. LISP steht für die Bezeichnung „*LISt Processing*" und verweist bereits darauf, dass diese KI-Sprache eine leistungsfähige Menge von Funktionen für die Listenverarbeitung bereitstellt. Von LISP haben sich über die Zeit verschiedenste Sprachversionen entwickelt, von denen Common LISP (CLIPS) besonders heraussticht, da es nach LUGER (2001) [Lug01] und KURBEL (1989) [Kur89] durch die Definition einer Kernmenge von Funktionen eine Art Quasi-Standard geschaffen hat.

Ein weiteres Spektrum an Tools wird unter dem Begriff Wissensverarbeitungssprachen zusammengefasst, Gemeinsam ist diesen Werkzeugen, dass sprachliche Konstrukte zur Darstellung von Wissensverarbeitungskonzepten auf einer höheren Ebene als bei den KI-Sprachen zur Verfügung gestellt werden [Kur89]. SRIRAM (1997) [Sri97] trennt hierbei noch zwischen der Gruppe der „*single paradigm languages*", die im wesentlichen nur eine Art von Wissensrepräsentationsform und Problemlösungstechnik anbieten, und der Gruppe der „*integrated paradigms languages (hybrids)*", die mehrere Repräsentationsformen und Problemlösungstechniken bereit stellen. Ein Beispiel für eine „*single paradigm language*" ist die Sprache OPS5, die die Entwicklung von regelbasierten Expertensystem mit Vorwärtsverkettung als Abarbeitungsstrategie unterstützt. Typische Vertreter für die „hybriden" Wissensverarbeitungssprachen sind COSMOS, LOOPS™, KEE™, Knowledge-CRAFT™, ART™ oder NEXPERT™. Diese Wissensverarbeitungssprachen werden nicht nur zur Realisierung von Expertensystemen angewendet, sondern auch in anderen Feldern der

KI. So sind typische Expertensystemelemente wie Erklärungskomponente oder Schnittstellen zum Wissenserwerb nicht direkt als Sprachelemente enthalten, sondern müssen vom Wissensingenieur selbst mit den Ausdrucksmitteln der Sprache implementiert werden.
Daher hat sich für den Bereich der Expertensysteme noch eine weitere Klasse von Werkzeugen herausgebildet, die Expertensystem-Shells. Eine Expertensystem-Shell stellt bereits ein Grundgerüst mit allen für ein Expertensystem typischen Komponenten dar. Die wesentliche Basis bei der Realisierung des Systems liegt damit auf dem Einbringen des Wissens in die Wissensbasis und dem Konfigurieren der integrierten Inferenz-Mechanismen. Oftmals ist eine Expertensystem-Shell aus einem konkreten Expertensystem-Entwicklungsprogramm hervorgegangen und wird anschließend im Grunde mit „leerer Wissensbasis“ als Werkzeug zur Realisierung weiterer Expertensysteme mit ähnlicher Aufgabenstellung angeboten. Dieses Vorgehen zeigt das Effizienzpotential für den Realisierungsprozess basierend auf Shells auf. Gleichzeitig offenbart es aber auch den Nachteil dieser Lösung, den Verlust an Flexibilität aufgrund der fest vorgegebenen Strukturen. Jedoch sind auch Expertensystem-Shells auf dem Markt, die unterschiedlichste Wissensrepräsentationsformen und Problemlösungstechniken- und -strategien unterstützten, was wiederum eine klare Unterscheidung zu hybriden Wissensverarbeitungssprachen erschwert [Kur89]. Bekannte Vertreter von Expertensystem-Shells sind Emycin, VP-EXPERT™, KES™, NEXPERT™, LEVEL-5™, G2™, NEXPERT™, ESE, CONGEN oder EDESYN. Eine Kurzbeschreibung dieser und weiterer Shells sowie auch von Vertretern der hybriden Wissensverarbeitungssprachen hat SRIRAM (1997) [Sri97] in einem Appendix zusammengestellt. Diese Zusammenstellung erhebt keinen Anspruch auf Vollständigkeit, aber gibt einen repräsentativen Überblick über ein zufolge HALL UND MCMULLEN (2004) [HM04] breites Angebot an kommerziellen Lösungen und frei verfügbaren Werkzeugen aus dem Bereich der Wissenschaft. Hinsichtlich einer aktuellen Übersicht verweisen sie auf die meist jährlich erscheinenden „*Buyer‘s Guides*“ in den entsprechenden Fachzeiten für KI und Expertensysteme.

5.2 Realisierung der Simulation

Die Realisierung der Simulation erfolgt in Form eines Expertensystems nach der im Kapitel 5.1.3 beschriebenen Methodik. Wie bereits hervorgehoben wurde, steht die Implementierung der MSDF für das *Level 2 Processing* im Mittelpunkt.

5.2.1 Identifikation

Die Identifikation umfasst hauptsächlich das übergelagerte Projektmanagement und die Fragen der prinzipiellen Aufgaben- und Problemstellung. Der zweite Aspekt lässt sich nach der Methode nach SRIRAM (1997) [Sri97] auch dem *Specification Level* zurechnen und wird daher im folgenden Kapitel wieder aufgegriffen. Die Ausführungen dieses Kapitels beschränken sich auf das organisatorische Umfeld, in der die Simulation umgesetzt und präsentiert wurde.

Die Realisierung der Simulation war Teil des Teilprojektes 8 „Entwicklung und Test einer Real-time-Prozessführung für sensorgestützte Düngesysteme“ der DFG Forschergruppe FOR 473 „Informationssystem Kleinräumige Bestandesführung Dürnast (IKB-Dürnast)“ [AOMSSW06] unter Leitung von Herrn Prof. Dr. Auernhammer. Das umfangreiche Verbundprojekt umfasste 12 Teilprojekte und hatte eine Gesamtlaufzeit vom 1.9.1998 bis zum 31.3.2005. Die erste 3-jährige IKB-Projektphase umfasste die Teilprojekte 1-7, an die sich die zweite ebenfalls 3-jährige Projektphase mit den Teilprojekten 8-13 anschlossen. Von besonderer Bedeutung für die Realisierung der Simulation war die Zusammenarbeit mit dem Teilprojekt 12 „Überprüfung von Produktionsfunktionen und Ableitung von Entscheidungsregeln für die teilflächenspezifische Bestandesführung“. Die angewandte Methodik, die Ergebnisse und ihre Diskussion hat WEIGERT (2006) [Wei06] in seiner Dissertation ausführlich dargelegt. Er hat für seine Arbeit wiederum eng mit dem Teilprojekt 13 „Ökologisch-ökonomische Bewertung der teilfächenspezifischen Bewirtschaftung“ zusammengearbeitet [Gan05]. Beide Projekte haben sowohl auf die in Feldversuchen gewonnenen kleinräumigen Daten der ersten IKB-Dürnast-Phase und das erstellte Informationssystem zurückgegriffen, als auch eigene auf ihre Aufgabenstellungen abgestimmte Feldversuche durchgeführt. Für die vorliegende Arbeit wurden keine eigenen praktischen Versuche zur (Test-)Datengewinnung durchgeführt, sondern auf die von WEIGERT (2006) [Wei06] zur Verfügung gestellten Daten zurückgegriffen. Im Speziellen wurde hierfür die Datenlage des Versuchsjahres 2003 auf dem Versuchsfeld D4 der TU München zur Applikation der zweiten N-Gabe (bei EC32) bei Winterweizen verwendet. Die exakte Versuchsanlage und Versuchsdurchführung ist bei [Wei06] im Kapitel 4.2.1 „Versuchsanlagen und Versuchsdurchführung TU München-Weihenstephan“ umfassend beschrieben. Im Anhang gibt die Abbildung A.1 die grundsätzliche Gestaltung der Versuchsanlage wider und zeigt die Aufteilung in einzelne Parzellen mit einer Größe von 7,5 m Breite auf 20 m Länge. Diese Breite konnte mittels der Teilbreitensteuerung vom eingesetzten pneumatischen Auslegerstreuer der Fa. Rauch, Sinzheim mit einer individuellen

Stickstoffmenge versorgt werden. Die 20 m Länge wurde gewählt, um möglichst kleinräumige Heterogenität erfassen zu können, aber auch um Fehler bei der Ausbringmenge und der Ertragserfassung zu vermeiden. Dabei wurde von stabilen Verhältnissen nur in einem Kernbereich von ca. 12 m ausgegangen, der mit einem Parzellenmähdrescher abgeerntet wurde. Für jede einzelne Parzelle lagen folgende Eingangsdaten für das Testen der Simulation vor:

- REIP-Messwert zur 2. N-Gabe (nm),
- Ertrag des Jahres 1998 (Korn-Ertrag Winterweizen; Mg/ha),
- EM38-Messwert (scheinbare elektrische Leitfähigkeit (mS/m) normiert auf 25°C (EC_{25})),
- Zugkraft-Messwert (gemessen bei Stoppelbearbeitung, kN),
- Düngermenge der 1. N-Gabe zu Vegetationsbeginn (kg N/ha).

Obige Schilderung hat bereits maßgebliche Quellen für die zur Realisierung und den Test der Simulation herangezogene Wissensquellen und Testdaten offengelegt. Dies führt auch zur Frage nach der Rollenverteilung der Beteiligten. Der Autor, der zu dieser Zeit als wissenschaftlicher Mitarbeiter am „Fachgebiet Technik im Pflanzenbau“ der TU München angestellt war, übernahm die Rolle des Wissensingenieurs bei der Implementierung des Expertensystems. Für das pflanzenbauliche und betriebswirtschaftliche Wissen konnte er auf die Projektbearbeiter der Teilprojekte 12 und 13 als Experten zurückgreifen und ihre Arbeitsergebnisse in die Simulation einfließen lassen. Die Ergebnisse der einzelnen Teilprojekte wurden in regelmäßigen Projekttreffen auch mit den Experten der anderen Teilprojekte vom „Lehrstuhl für Pflanzenernährung“ (TP 10) und dem „Lehrstuhl für Ökologischen Landbau und Pflanzenbausysteme“ (TP 9) der TU München diskutiert und abgeglichen. Als weitere Wissensquellen wurden die Veröffentlichungen der internationalen wissenschaftlichen Gemeinschaft zum Thema Präziser Ackerbau herangezogen sowie Lehrbücher aus den Bereichen Landtechnik, Informations- und Automatisierungstechnik. Der Leiter des „Fachgebietes Technik im Pflanzenbau“ der TU München, Herr Prof. Dr. Auernhammer, und die Kollegen an diesem Fachgebiet standen bei Bedarf als Experten für agrarsystemtechnische Fragestellungen zur Seite. Der zeitliche Rahmen war vorgegeben durch die Projektlaufzeit des DFG-Projektes mit internen Präsentationen der Zwischenstufen gemäß dem evolutionärem Entwicklungsprozess und einer abschließenden Demonstration der Simulation für das interessierte Fachpublikum auf dem Abschluss-Symposium der IKB-Dürnast DFG Forschergruppe am 11. und 12. Oktober 2005 am Campus Weihenstephan der TU München.

5.2.2 Konzeption und Formalisierung

5.2.2.1 Specification Level

Der Aspekt der Aufgaben- und Problemstellung und die Identifizierung der zugehörigen Charakteristika wurde in dieser Arbeit bereits an mehreren Stellen durchgeführt und soll daher in diesem Unterkapitel nicht mehr wiederholt sondern nur auf die entsprechenden Kapitel referenziert werden.

Aus Sicht der Zielidentifizierung bleibt nochmals festzuhalten, dass es gilt, Expertenwissen auf dem Feld (*in-field*) in Form einer Automatisierungslösung verfügbar zu machen, die eine minimale Nutzerinteraktion erfordert und hinsichtlich der Durchführungshäufigkeit für einen 7 Tage und 24 Stunden Dauereinsatz grundsätzlich tauglich sein sollte.

5.2.2.2 Task Level

Die *Task Level* Einordnung innerhalb der Spanne des Anwendungsbereiches für Expertensysteme gibt keine Rätsel auf. Die Aufgabenstellung, der *Task*, ist der Klasse der Steuerungen zuzurechnen. Der Sensor-Ansatz mit Kartenüberlagerung erfordert die Situationsbeurteilung der vorliegenden Teilfläche und der darauf basierenden Applikationssollwert-Ableitung innerhalb einer fest vorgegebenen Zeitspanne. Dies fordert bzw. unterstützt zumindest die Auslegung der Steuerung basierend auf einer derivativen Orientierung. Dies bedeutet, dass es vorteilhaft für die Echtzeitfähigkeit ist, wenn Ergebnisse bzw. Teile davon bereits in der Wissensbasis existieren und möglichst effizient „gesucht“ werden können.

5.2.2.3 Problem Solving Level

In dieser Arbeit wird der Ansatz verfolgt, ein geeignetes Problemlösungsverfahren hauptsächlich aus dem Blick der MSDF-Technik abzuleiten. Für den für die Simulation ausschlaggebenden Fall des Sensor-Ansatzes mit Kartenüberlagerung für intensive Stickstoffdüngung wurde im Kapitel 4.3.2 diese Analyse dargelegt. Diese Analyse kam zu dem Ergebnis, dass der *generation-based* Problemlösungsansatz „*canonical form IX*“, ein Produktionssystem mit vorwärtsverkettetem Inferenzmechanismus, einen geeigneten Lösungsansatz darstellt. Mit der Entscheidung die Simulation in Form eines Expertensystems umzusetzen, wurde diesem Paradigma auf prinzipieller Ebene schon Rechnung getragen. Für die einzelnen Details gibt dies die Leitlinien für die Festlegungen auf der Ebene des *Problem Solving Level* und dem *Knowledge-base Level* vor.

Ein kurzer Blick zurück auf das *Level 2 Processing* des funktionalen Modells (vgl. Kap. 4.2.4) gibt einen Hinweis, wie ein Experte die Aufgabenstellung beispielsweise lösen würde. In einem ersten Schritt würde er basierend auf den Pflanzen- und Bodeneigenschaften unter Einbeziehung des aktuellen Wetters und des aktuellen Zeitpunktes den aktuellen Pflanzen- und Bodenzustand auf der vorliegenden Teilfläche diagnostizieren.

- In einem weiteren Schritt würde er das grundsätzliche Ertragspotenzial basierend auf korrigierten Ertragskarten und die im gleichen Jahr bereits ausgebrachten Düngergaben mitberücksichtigen. Anschließend würde er überprüfen, ob Auflagen und Grenzwerte im Sinne des Umweltschutzes vorliegen und einzuhalten sind.
- In einem darauffolgenden Schritt sollten all diese Diagnosen, Beziehungen und Beschränkungen mit einer modellbasierten Vorstellung eines ökologischen und ökonomischen Optimums verglichen werden.
- Vor der endgültigen Empfehlung eines geeigneten Applikations-Sollwertes für die vorliegende Teilfläche wird er noch die technische Machbarkeit, den Funktionszustand seiner Sensor- und Applikationstechnik und die Vermeidung von wiederholter Applikation mit in Betracht ziehen.

Was für den Experten keine Rolle spielt, dafür aber für eine Automatisierungslösung gefragt ist, ist die Möglichkeit, dass ein Endbenutzer die Sollwertempfehlung übersteuern kann. Diese Strukturierung in einzelne Problemlösungsschritte lässt sich durch Modularisierung der Wissensbasis bzw. des angewandten Regelwerkes auf die konkrete Aufgabenstellung übertragen. Angewandt auf die vorliegende Aufgabenstellung und die Möglichkeiten zur Wissensakquisition innerhalb der IKB-Dürnast DFG-Forschergruppe wurde folgende Aufteilung in Hauptmodule gewählt:

- Pflanzenbauliche Situationsbewertung (Modul: *„CROP_PRODUCTION“*),
- Einbeziehung von Beschränkungen (Modul: *„CONSTRAINTS“*),
- Technische Situationsbewertung (Modul: *„AG_ENGINEERING“*),
- Zusammenfassung und Applikationssollwertempfehlung (Modul: *„SUM_UP“*).

Das weitaus wichtigste und umfangreichste Modul repräsentiert dabei das „*CROP_PRODUCTION“*-Modul. Es fasst das Expertenwissen, das im IKB-Teilprojekt 12 mit Berücksichtigung von Teilprojekt 13 erarbeitet wurde, in Form von Entscheidungsregeln zusammen (vgl. Kap. 5.2.2.4). Als Zwischenergebnis empfiehlt dieses Modul einen Applikationssollwert (*„CROP_PRODUCTION-SUM_UP::N_recommendation“*), der den Wert in der Maßeinheit kg N/ha beinhaltet und für die Erklärungs- und Dokumentationsfunktionalität eine knappe Texterklärung des angewandten Lösungsweges beinhaltet. Auf die Regeln, die diesen Abschnitt des prozeduralen Teils der Wissensbasis ausmachen, wird im zugehörigen Abschnitt des Kapitels 5.2.2.4 näher eingegangen.

Das nächste Modul „*CONSTRAINTS*“ umfasst drei Funktionskategorien, die bei der Problemlösung zur Anwendung kommen. Die erste Kategorie integriert Begrenzungen, z.B. im Sinne von Umweltschutzauflagen, in Form von einer oder mehreren vordefinierten „Hintergrundkarten“, alternativ auch als Ergebnis von „Online-Abdrift-Berechnungen“ in die Lösungsfindung. Die zweite Kategorie ist der Vermeidung von Über-Applikation gewidmet. Diese könnte durch mehrfache Applikation auf der vorliegenden Teilfläche als Folge einer Überlappung oder durch versehentliches wiederholtes Befahren einer Fahrgasse entstehen. Die dritte Kategorie trägt den Übersteuerungswünschen des Endbenutzers Rechnung, indem sie einen relativen Einstellungswert „*Adjustment_factor*“ mit in die Menge der Fakten während des Inferenz-Prozesses einspeist. In der grundsätzlichen Systemauslegung wurde für zukünftige Erweiterungen ebenfalls vorgesehen, dass weitere Systemteilnehmer, z.B. übergelagerte Steuerungen oder gleichberechtigte Teilnehmer im Sinne einer „kooperativen Sollwert-Kompromissfindung“ ihre Übersteuerungswünsche mit einbringen könnten. Analog zum *„CROP_PRODUCTION“*-Modul empfiehlt auch das „*CONSTRAINTS*“-Modul als Zwischenergebnis einen Applikationssollwert (*„CONSTRAINTS-SUM_UP::N_recommendation“*), der den Wert in der Maßeinheit kg N/ha beinhaltet und für die Erklärungs- und Dokumentationsfunktionalität ebenfalls eine knappe Texterklärung des angewandten Lösungsweges beinhaltet. Weiterhin wird nachfolgenden Problemlösungsschritten ein relativer Übersteuerungswert (*„CONSTRAINTS-SUM_UP::Adjustment_factor“*) und der Wert der bereits ausgebrachten Stickstoffmenge auf dieser Teilfläche zur aktuellen N-Gabe (*„CONSTRAINTS-SUM_UP::N_applied“*) bereitgestellt. Auch diese Werte sind mit entsprechenden Kurzerklärungen versehen.

Das dritte Modul zur Strukturierung der Wissensbasis und des Problemlösungsprozesses ist das „*AG_ENGINEERING*“-Modul. Im Wesentlichen integriert es die Auswahl des Arbeitsmodus der Prozesssteuerung, d.h. manuell gegenüber automatisch, eine mögliche Sollwertvorgabe im manuellem Modus, sowie die Zustandsbewertung der Gesamtsteuerung und im Speziellen der Online-Sensorik und der Applikationstechnik hinsichtlich der Einhaltung von Genauigkeit und Echtzeitfähigkeit. Für die vorliegende Arbeit beschränkt sich diese Zustandsbewertung auf die Angabe, ob ein Online-Sensor zur REIP Bestimmung vorhanden bzw. funktionstüchtig ist. Die Beurteilung der Applikationstechnik, der Echtzeitfähigkeit und der Genauigkeit der Prozessführung wurde nicht umgesetzt, jedoch wurde das Fehlen für Erklärungs- bzw. Dokumentationszwecke als Fakten für die Wissensbasis hinterlegt. Als Zwischenergebnis steuert dieses Modul somit gegebenenfalls einen Applikationssollwert (*„AG_ENGINEERING-SUM_UP::N_recommendation“*) mit

Kurzerklärung zur Gesamtlösung bei. Ebenfalls wird die Faktensammlung um den gewählten Modus der Prozesssteuerung erweitert (*„AG_ENGINEERING-SUM_UP::Control_Mode“*). Bei Vorliegen von Zustandsberichten von mehreren Online-Sensoren, der Düngerstreuertechnik und der Steuerung als solches, werden alle in die Menge der Fakten, vorrangig als Erklärungstexte, übernommen. Für die vorliegende Arbeit gilt dies nur für eine Aussage, ob ein „REIP-Online-Sensor“ vorhanden ist und genutzt werden soll (*„AG_ENGINEERING-SUM_UP::Quality_statement“*).

In einem vierten und abschließenden Schritt, wiederum zusammengefasst in einem Modul mit dem Namen „SUM_UP“, werden alle Zwischenergebnisse gesammelt, bewertet und letztendlich daraus die aktuelle N-Applikationssollwertempfehlung abgeleitet. Dieses Modul stellt als Ergebnis den Sollwert „*N2-Setpoint*“ in der Maßeinheit kg N/ha für die Funktionalität der Sollwertkommandierung an die Applikationstechnik und zur Information des Endnutzers auf der Mensch-Maschine-Schnittstelle bereit. Die zur Lösungsfindung herangezogenen Schritte wurden mittels der verknüpften Kurzerklärungen mitgeführt, aneinandergereiht und stehen der Erklärungskomponente und der Dokumentationsfunktion zur Verfügung.

Parallel zu dem vorgestellten Lösungsprozess ist ein weiteres Modul bzw. Funktion stets präsent, welches die Prozessführung jederzeit stoppen bzw. pausieren lassen kann. Dieses Modul „EMERGENCY_STOP“ bedient damit die Notaus-Funktionalität auf Ebene der „*In-field Controller”*-Prozessführung. Dieses Modul verfügt über die Fähigkeit den Fokus des Inferenz-Prozesses unmittelbar auf sich zu ziehen, die Schlussfolgerung abzubrechen und durch das eigene Regelwerk zu prozessieren. Die Entscheidungsfindung ist äußerst einfach gehalten und bestimmt einen Applikationssollwert von 0 kg N/ha für den „*N2-Setpoint*“.

Mit der Wahl eines Produktionssystems mit vorwärtsverkettetem (*forward chaining*) Inferenzmechanismus kommen zwei „starke Problemlösungsmethoden“ bzw. wissensbasierte Ansätze zum Einsatz. Vorwärtsverkettung bedeutet eine datengetriebene Vorgehensweise, die ausgehend von bekannten Daten aus der Datenbasis schlussfolgert, bis ein Ziel erreicht ist. Angewandt auf das „*CROP_PRODUCTION“*-Modul sind die Symptome oder bekannten (Eingangs-)Daten, der REIP-Messwert zur 2. N-Gabe, der Ertragswert des Jahres 1998, der EM38-Messwert, der Zugkraft-Messwert und die Düngermenge der 1. N-Gabe. Ausgehend von diesen Fakten versucht das System, auf eine Lösung aus der Menge der möglichen Applikationswerte zu schließen. Die Vorwärtsverkettung angewandt auf die Module „*CONSTRAINTS*“, „*AG_ENGINEERING*“ und „*SUM_UP*“ bedeutet, dass der Schlussfolgerungsprozess ausgehend von verschiedenen Applikationssollwertempfehlungen,

Arbeits-/Systemmodi, Übersteuerungsfaktoren hin zu einem einzigen N-Sollwert für die aktuelle Teilfläche führt. Im theoretischen Teil dieser Arbeit zum Produktionssystem wurde bereits angesprochen, dass der „*Erkennen-Handeln-Zyklus"* in den meisten Fällen zu einer Konfliktmenge von mehreren aktivierten Regeln mit erfüllten Prämissen führt. Ein erster Schritt einer Konfliktlösung wurde mit der Strukturierung des prozeduralen Teils der Wissensbasis in Module bereits durchgeführt. Denn die einzelnen Suchräume der einzelnen Module sind begrenzter, als wenn alle Fakten und Daten in einen gemeinsamen Suchraum gleichzeitig vorliegen. Ein weitere Möglichkeit zur Konfliktlösung wären Prioritätszuweisungen für einzelne Regeln. Davon wurde in dem vorliegenden Ansatz jedoch Abstand genommen, da ein übermäßiger Einsatz dieser Methode nicht der guten fachlichen Praxis der regelbasierten Expertensystementwicklung entspricht, sondern eher darauf hinweist, dass ein wissensbasierter Ansatz möglicherweise der Aufgabenstellung nicht gerecht wird. Zu dieser Problematik schreibt FRIEDMAN-HILL (2003) [Fri03]: *„…. Note that extensive use of salience is generally discouraged, for two reasons. First, use of salience has a negative impact on performance, at least with the built-in conflict resolution strategies. Second, it is considered bad style in rule-based programming to try to force rules in a particular order. If you find yourself using salience on most of your rules or if you are using more than two or three different salience values, you probably need to reconsider whether you should be using rule-based approach to your problem. If you want strict control over execution order, then you're trying to implement a procedural program. …"*[42]

Im ersten Teil der Funktionsbeschreibung für die ersten drei Module wurde jeweils das Namenskürzel „*SUM_UP*" angeführt. Dies liegt darin begründet, dass dem *„CROP_PRODUCTION"-,* dem „*CONSTRAINTS*"- und dem „*AG_ENGINEERING*"-Modul stets ein zugehöriges „*SUM_UP*"-Modul nachgeschaltet ist. Da in den Hauptmodulen nicht immer davon ausgegangen werden kann, dass der Schlussfolgerungsprozess nur ein einziges Zwischenergebnis für die schlussendliche Weitergabe an das abschließende „*SUM_UP*"-Modul ergibt, wird auf dieser Stufe eine Aggregation der Zwischenergebnisse durchgeführt. Weiterhin ist diese Ebene dafür verantwortlich einen Default-Wert in den aktuellen Kontext einzuspeisen, falls das jeweilige Hauptmodul zu keinem (Zwischen-)Ergebnis gekommen ist. Selbst diese Konfliktlösung basiert im Grunde auf Heuristiken, also auf der Anwendung von Zusatzwissen. Auf wesentlich elementarer Ebene ist für ein Produktionssystem eine Entscheidung für eine Suchstrategie aus dem Bereich der suchbasierten

[42] A. a. O., S. 122-123

Problemlösungsmethoden zu treffen, z.B. zwischen Tiefen- und Breitensuche (*depth-first*, *breadth-first*). Auf dieser Ebene kommen Verfahren zur Zustandsraumsuche zum Einsatz. Eine wichtige Klasse von Zustandsräumen sind Baumstrukturen, die oftmals bei Produktionssystemen zum Einsatz kommen (vgl. Kap. 5.2.2.4). Baumstrukturen gehen von einem Wurzelknoten aus und verzweigen in einzelne Äste, wobei die Verzweigungsknoten bzw. die Enden der Äste einen Zustand repräsentieren. Laut SRIRAM (1997) ist die Wahl einer Tiefen- oder Breitensuche für eine gegebene Problemstellung abhängig von der Position des Zielzustandes innerhalb des baumförmigen Suchraumes [Sri97]. Er verweist darauf, dass die Breitensuche tendenziell für Fälle effektiver ist, bei denen der Zielzustand näher am Wurzelknoten liegt und der Verzweigungsfaktor klein ist, während die Tiefensuche Stärken bei einem Zielzustand zeigt, der hinsichtlich der Tiefe weiter entfernt vom Wurzelknoten liegt und einen höheren Verzweigungsfaktor aufweist. Eine genaue Analyse wäre möglich, indem die Länge des *worst case*-Suchweges und des mittleren Suchweges für den jeweiligen Fall aufsummiert und verglichen werden. Für eine Tiefen- oder Breitensuche ohne weitere Heuristiken, d.h. eine vollständige Suche, lässt sich dies mathematisch formulieren [Sri97], [Lug01].

Für die vorliegende Aufgabenstellung wurde die Tiefensuche als Suchstrategie gewählt, da vor allem für das *CROP_PRODUCTION*-Modul ein Suchraum mit größerer Tiefe und Verzweigungsfaktor entstehen kann. Die Tiefe des Suchraumes wird dabei durch die Anzahl der Eingangsdatentypen bestimmt, d.h. REIP-Messwert zur 2. N-Gabe, Ertragswert des Jahres 1998, EM38-Messwert, Zugkraft-Messwert und Düngermenge der 1. N-Gabe. Diese fünfstufige Tiefe für den umgesetzten Anwendungsfall könnte sich jedoch für zukünftige Anwendungsfälle mit weiteren Eingangsvariablen auch wesentlich vergrößern. Eine weitere detaillierte Gegenüberstellung wurde nicht durchgeführt, zumal für den im Produktionssystem realisierten *„Erkennen-Handeln-Zyklus“* dieses grundsätzliche Suchverfahren in modifizierter Form zum Einsatz kommt (vgl. Ausführungen zum RETE-Algorithmus in Kap. 5.2.2.5).

Hinsichtlich des Umganges mit dem Aspekt der Unsicherheit (*uncertainty*) bzw. nicht eindeutigem Wissens wurde kein explizites Verfahren wie z.B. Konfidenzfaktor-Algebra oder Dempster-Shafer-Theorie angewandt. Die grundlegende Idee war dabei, den MSDF entsprechend auszunutzen, dass auf *JDL Level 2 Processing*-Ebene mit sicherem Wissen gearbeitet wird und die Unsicherheit in den Eingangsdaten bereits auf der *JDL Level 1 Processing*-Ebene adressiert wird. Die teilflächenspezifischen Eingangsdaten, die auf *Precision Farming*-Karten beruhen, wurden bereits in einem FMIS oder GIS (vgl. Kap. 2.3.2.5) mit Methoden zur Datenbereinigung, -filterung und -korrektur aufbereitet, wie sie in

Kapitel 2.3.2.3 beschrieben wurden. Für die vorliegende Simulation ist der im Rahmen des IKB-Dürnast-Forschungsprojektes neu entwickelte Kartierungsalgorithmus „Paraboloides Butterfly Fitting“ [BA05] hervorzuheben, da er zur Vorverarbeitung der Ertragsdaten zum Einsatz kam. Diese Vorverarbeitung und die Datenaufbereitung für das weitere Hintergrunddatenmaterial (*overlay maps*) sind bei [Wei06] im Kapitel „4.3.1 Datenaufbereitung IKB-Versuche“ dokumentiert. Für die Messwerte der Online-Sensorik wird ebenfalls davon ausgegangen, dass sie auf der *JDL Level 1 Processing*-Ebene entsprechend vorverarbeitet werden. Hierzu bieten sich einerseits einfache Methoden wie Mittelwertverfahren (z.B. *simple moving average*), simple Filterverfahren (z.B. Tiefpass) oder das Einfügen von Default-Werten an. Anderseits empfiehlt sich die Anwendung von statistischen Schätzverfahren wie z.B. die Kalman Filterung [HM04]. Für den vorliegenden Fall kommt als Online-Sensorwert nur der REIP-Wert zur zweiten N-Gabe zum Einsatz, der aufgrund der Versuchsanstellung bereits bei einer Vorabbefahrung bzw. -begehung ermittelt und dann ebenfalls wie ein Kartenwert vorverarbeitet wurde [Wei06].
An einer Stelle der Simulation wurde aber doch noch der Weg beschritten, Unsicherheit auf der *JDL Level 2 Processing*-Ebene zu behandeln. Für den Fall, dass der REIP-Sensorwert fehlt oder im Rahmen der Sensorstatusbewertung als zu unzuverlässig beurteilt wird, kommt ein alternatives Regelwerk innerhalb des „*CROP_PRODUCTION“*-Moduls zum Einsatz, das ohne die Berücksichtigung des REIP-Wertes eine Sollwertempfehlung ableitet.

5.2.2.4 Knowledge-base Level

Im theoretischen Teil zum *Knowledge-base Level* wurde bereits auf die beiden unterschiedlichen Kategorien der Wissensrepräsentation, die deklarative und die prozedurale Repräsentationsform, hingewiesen. Ein kurzer Rückblick auf das Prozessmodell (vgl. Kap. 4.3.1) offenbart das nötige Wissen für den Sensor-Ansatz mit Kartenüberlagerung und zeigt die Zuordnung der einzelnen Wissensarten zur deklarativen oder prozeduralen Kategorie. Der deklarativen Kategorie können zugerechnet werden:

- die Sensordaten (REIP-Wert, Position) als *kurzfristiges deklaratives Wissen*,
- die Kartendaten (Ertrag des Jahres 1998, EM38-Messwert, Zugkraft-Messwert, Düngermenge der 1. N-Gabe, Grenzwert Umweltschutz) als *spezifisches langfristiges deklaratives Wissen,*
- Zwischenergebnisse oder Ergebnisse der Situationsbewertung (z.B. *CONSTRAINTS-SUM_UP::N_applied oder N2-setpoint*) als *mittelfristiges deklaratives Wissen*.

Die Eingaben über die Mensch-Maschine-Schnittstelle sind am besten den Sensordaten als *kurzfristiges deklaratives Wissen* gleichzusetzen, in den Fällen der Modus-Auswahl ist auch

die Einstufung als *mittelfristiges deklaratives Wissen* angebracht. Die bereits im *Problem Solving Level* aufgeführten Problemlösungsschritte setzen den eigentlichen *Data Fusion*-Prozess um und entsprechen dem *langfristigen expliziten prozeduralen Wissen*. Weiterhin erfordert die Implementierung auch einige Wissenselemente, die helfen, die Aufgabenlösung durch „Infrastruktur"-Maßnahmen zu unterstützen. Je nach Ausprägung sind dies eher deklarative Daten oder aber prozedurales Wissen, das beispielsweise auf die Ablaufsteuerung einwirkt.

Bei der Realisierung der Simulation sollte für die Umsetzung des deklarativen Wissens eine möglichst intuitive und skalierbare Repräsentationsform angewandt werden. Eine derartige Abbildung der einzelnen Elemente der Aufgabenstellung lässt sich durch die Anwendung einer objektorientierten Repräsentation erreichen. So können Sensoren, Kartenmaterial, Applikationstechnik, Mensch-Maschine-Schnittstelle als generische Objekte mit ihren zugehörigen Eigenschaften und integrierten Methoden umgesetzt werden. Der Aufbau von Hierarchien ist möglich und bereits durch die Instanziierung mehrerer Objekte einer Klasse lässt sich Skalierbarkeit erreichen, ohne die Möglichkeiten der Vererbung explizit genützt zu haben.

Ausgehend von der Analyse des Prozessmodells und in Abstimmung mit den Überlegungen zum *Problem Solving Level* fiel die Wahl der Repräsentationsform für das prozedurale Wissen auf eine regelbasierte Form.

Dieses wurde innerhalb der Module *„CROP_PRODUCTION", „CROP_PRODUCTION-SUM_UP",* „*CONSTRAINTS*", „*CONSTRAINTS-SUM_UP*", „*AG_ENGINEERING*", „*AG_ENGINEERING-SUM_UP*" und „*SUM_UP*" formuliert.

5.2.2.4.1 Modul „*CROP_PRODUCTION*"

Von zentraler Bedeutung für die realisierte Simulation ist das Wissen des *„CROP_PRODUCTION*"-Moduls, das im IKB-Dürnast-Teilprojekt 12 unter Anwendung von *Data Mining* und Wissensentdeckung (WED) im Präzisions-Ackerbau erarbeitet bzw. akquiriert wurde. Aufgrund der Wichtigkeit dieses Wissensmoduls für die vorliegende Arbeit werden die grundsätzlichen Ansätze knapp zusammengefasst. Gemäß WEIGERT (2006) setzt sich die *Data Mining*-Phase, das zentrale Element innerhalb dieses modifizierten WED-Prozesses, aus der Exploration der Daten, der Modellierung einer teilflächenspezifischen Ertragsprognose und Ansätzen zur Validation und der Extraktion von ökonomisch optimierten Entscheidungsregeln zusammen [Wei06]. Die letztendliche Ableitung des Wissens, das in Form von Entscheidungsregeln vorliegen soll, teilt sich bei der angewandten *Data Mining*-

Technik in drei Teilaufgaben auf, die aufeinander aufbauen. In einem ersten Schritt wird ein Künstliches Neuronales Netz für eine teilflächenspezifische Ertragsprognose für variable Stickstoff-Mengen erarbeitet. In einem zweiten Schritt wird dieses Prognosemodell verwendet, um für eine bestimmte teilflächenspezifische Situation die ökonomisch optimale Stickstoff-Applikation herauszufinden. Dies wird durch die Simulation mit einer begrenzten Auswahl von technisch möglichen Düngergaben erreicht, deren Bewertung und darauf basierende Auswahl einer Sollwertempfehlung nach ökonomischen Gesichtspunkten erfolgt. Angewandt auf eine Auswahl von Teilflächen, die mit dem vorliegenden Datenmaterial eine möglichst gute Abdeckung von Merkmalsausprägungen erlauben, ergeben diese Schritte einen Datensatz, welcher in einem dritten Schritt verwendet wird, um mit einem Entscheidungsbaumverfahren entsprechende Regeln abzuleiten. WEIGERT (2006) betont neben der eigentlichen Aufgabenerfüllung den mit einer Aufteilung in Teilschritte einhergehenden Vorteil, dass sich dadurch überprüfbare Zwischenschritte finden lassen, z.B. die Ergebnisse der teilflächenspezifischen Ertragsprognose, die mit den Ergebnissen des aktuellen Standes der Technik verglichen werden können [Wei06].

Für die Simulation konnte mit den ersten beiden Teilschritten für alle Instanzen (Teilflächen) des Versuchsfeldes D4 der TU München des Versuchsjahres 2003 die optimale Applikationsrate basierend auf den entsprechenden kleinräumigen Attributsausprägungen bestimmt werden. Damit ließ sich ein Datensatz über alle Teilflächen erzeugen, der zu den entsprechenden Attributen jeweils auch die optimale N-Gabe aufwies und damit als Trainingsmenge für das Entscheidungsbaumverfahren auf Basis des C&RT (*Classification and Regression Trees*)-Algorithmus [BFOS84] diente. Entscheidungsbäume lassen sich relativ einfach in eine regelbasierte Form überführen. Um die Interpretationsfähigkeit der abgeleiteten Entscheidungsbäume bzw. Entscheidungsregeln für Experten und Nutzer zu gewährleisten, hat WEIGERT (2006) [Wei06] sich in seiner Arbeit darauf beschränkt, die Tiefe der Bäume auf drei Ebenen zu begrenzen. Abbildung 5.5 zeigt den entwickelten Entscheidungsbaum.

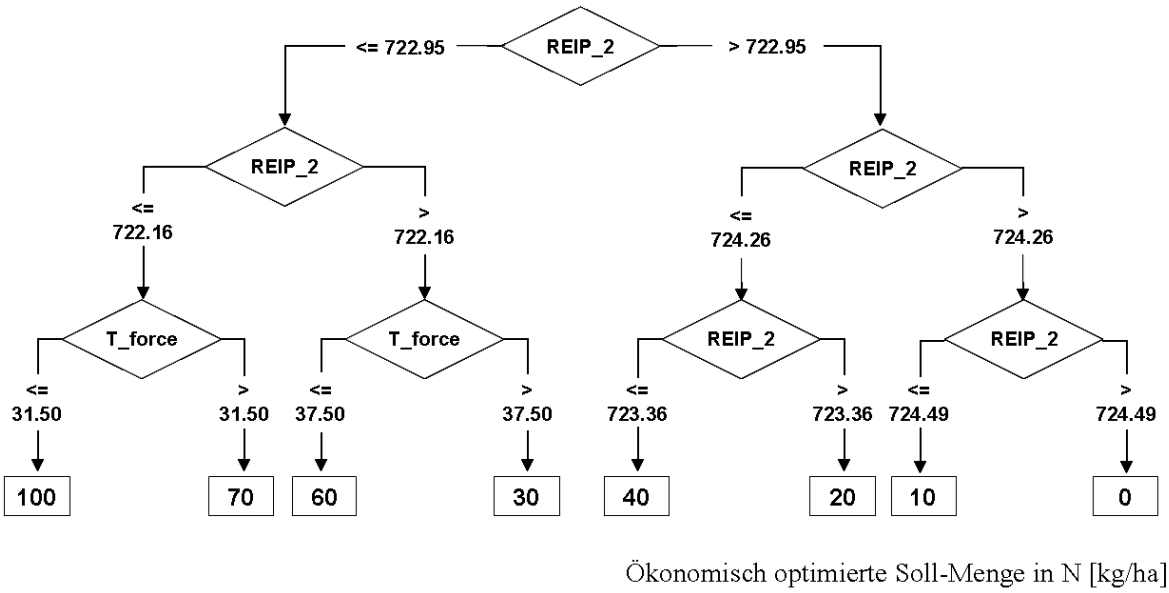


Abbildung 5.5: Entscheidungsbaum zur zweiten N-Applikation (D4-03) (nach [Wei06])

Folgende Abkürzungen werden in dieser Arbeit für die Attribute in den Entscheidungsbäumen verwendet (basierend auf den Schnittstellen-Absprachen zwischen IKB TP8 und IKB TP12):

- REIP-Wert zur 2. N-Gabe: *REIP_2,*
- Ertrag des Jahres 1998: *Yield98,*
- EM38-Messwert: *EM38,*
- Zugkraft-Messwert: *Tforce*,
- Düngermenge der 1. N-Gabe: *N_1.*

Dieser Entscheidungsbaum zeigt den dominierenden Einfluss des Vegetationsindexes *REIP_2* auf die Bestimmung der optimalen Applikationsrate. Erst auf der dritten Ebene hat der Zugkraftwert einen Einfluss auf den Auswahlprozess, dort führt er jedoch zu starken Unterschieden bei den empfohlenen Stickstoff-Applikationswerten.

WEIGERT (2006) [Wei06] wies darauf hin, dass die Ebenen-Einschränkung beim Entscheidungsbaum aus Gründen der Minimierung der Komplexität fast zwangsläufig zu Abstrichen bei der Qualität des Ergebnisses führt. Für den Einsatz in einem automatisierten Verfahren, wie der Implementierung einer Prozessführung über einen *„In-field Controller"* ist diese Ebenen-Beschränkung nicht nötig und kann mit der optimalen Komplexität durchgeführt werden. Für die vorliegende Arbeit wurden daher ein Entscheidungsbaum und dessen alternative regelbasierte Repräsentationsform mit einer Tiefe von bis zu maximal neun Ebenen erzeugt. Dieser Entscheidungsbaum ist im Anhang (vgl. Kap. A.2) als Abbildung A.2 dokumentiert. Vor einem etwas detaillierteren Blick auf diesen Entscheidungsbaum werden

noch einige Anmerkungen zur Beschriftung des Entscheidungsbaumes und dem Zusammenhang mit der Umsetzung im „*CROP_PRODUCTION“*-Wissensmoduls gegeben.
Im *CROP_PRODUCTION-Modul* wird dieser Entscheidungsbaum in einzelne Entscheidungsregeln umgesetzt und jedes Blatt, d.h. jede Applikationsratenempfehlung, ergibt ein entsprechendes Zwischenergebnis *CROP_PRODUCTION::N_recommendation (setpoint*, *explanation*). Für *setpoint* wird die jeweilige Sollwertempfehlung in der Maßeinheit kg N/ha eingesetzt und für die Erklärungs- und Dokumentationsfunktionalität wird die angewandte Entscheidungsregel mit dem zugehörigen Kürzel (*rule-ID*) in der Variable *explanation* beschrieben. Für die Erklärungs- und Dokumentationsfunktionalität wurde daher jedem Blatt eine eindeutige Identifikationsbeschreibung zugeordnet, die mit dem Buchstaben N als Referenz auf die Stickstoff-Applikationswertempfehlung beginnt, dann um den numerischen Wert des Sollwertes ergänzt wird und nach einem Trennzeichen „-“ werden gleiche Applikationsraten mittels einer aufsteigenden Nummerierung unterschieden. Diese *rule-ID’*s sind im Entscheidungsbaum ebenfalls aufgeführt. Exemplarisch wird dieses Prinzip für drei „Äste“, die zu einer Sollwertempfehlung von 70 kg N/ha führen, in Abbildung 5.6 gezeigt.

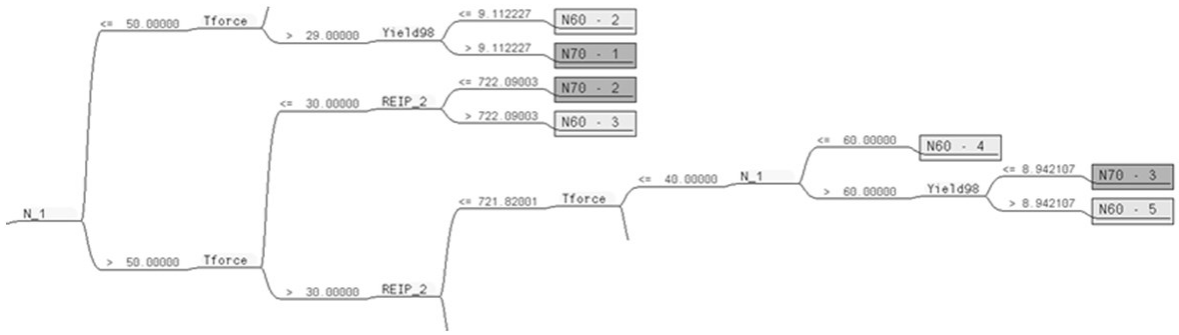


Abbildung 5.6: Benennung der Blätter im Entscheidungsbaum

Für diese Äste, die bei unterschiedlichen teilflächenspezifischen Attributwerten jeweils zur selben Stickstoff-Sollwertempfehlung kommen, unterscheiden sich die Zwischenergebnisse über die Variable der Erklärungs- und Dokumentationsfunktionalität:

- *CROP_PRODUCTION::N_recommendation (70, N70 - 1),*
- *CROP_PRODUCTION::N_recommendation (70, N70 - 2),*
- *CROP_PRODUCTION::N_recommendation (70, N70 - 3).*

Mit der erläuterten Beschreibungsform und am Beispiel der Regel „N70-1“ wird nachfolgend die Transformation eines Entscheidungsbaum-Astes in eine Entscheidungsregel in einem

„*Pseudo-Code*“-Format gezeigt. Die tatsächliche Umsetzung dieses „*Pseudo-Code*“ in eine lauffähige Implementierung ist im Anhang Kap. A.4.3 dokumentiert.

REGEL CROP_PRODUCTION „Regel N70-1 / Applikationssollwertempfehlung 70 kg N/ha"
WENN
Ein REIP_2-Wert in der Faktenbasis vorliegt, der kleiner gleich 722,45001 ist
Ein REIP_2-Wert in der Faktenbasis vorliegt, der größer 721,62000 ist
Ein N_1-Wert in der Faktenbasis vorliegt, der kleiner gleich 50,00000 ist
Ein Tforce-Wert in der Faktenbasis vorliegt, der größer 29,00000 ist
Ein Yield98-Wert in der Faktenbasis vorliegt, der größer 9,11227 ist
DANN
Füge CROP_PRODUCTION::N_recommendation (70, N70 - 1)
neu zur Faktenbasis hinzu

Die Regel ist nahezu selbsterklärend. Im Bedingungsteil der Regel wird auf die Existenz der einzelnen Attribute geprüft und zugleich untersucht, ob ein zugeordneter Größenvergleich ein positives Ergebnis liefert. Wird in der Faktenbasis ein derartiges Muster mit den vier Attributen gefunden, so ist die Regel aktiv. Im Aktionsteil ist definiert, dass ein neuer Fakt vom Typ *CROP_PRODUCTION::N_recommendation (setpoint*, *explanation*) mit den Werten 70 für *setpoint* und *N70 - 1* für *explanation* in die Faktenbasis eingestellt wird, falls die Regel zum „Feuern“ kommt.

Nachdem die Nomenklatur und das Verfahren zur Regelableitung aufgezeigt wurden, soll ein Blick auf einige Charakteristika des verwendeten Entscheidungsbaumes geworfen werden. Die einfache Betrachtung der Häufigkeit eines Attributs zeigt, dass *REIP_2* mit zehnmaligen Auftreten an der Spitze liegt, die Attribute *N_1, Tforce und Yield98* mit jeweils siebenmaligen Auftreten gleich verteilt sind und das Attribut *EM38* nur einmal zur Entscheidung herangezogen wird. Die reine Häufigkeit eines Attributes ist bereits ein Anhaltspunkt für die Dominanz bei der Entscheidungsfindung, jedoch ist auch deren Verteilung über die Äste und die Position in der Ebenenhierarchie mitentscheidend. Wie schon beim Entscheidungsbaum mit nur drei Ebenen ist der Vegetationsindex *REIP_2* ebenfalls auf den ersten beiden Ebenen vorherrschend und wirkt auch auf den folgenden vier Ebenen bei verschiedenen Ästen an der Entscheidung mit. Auf der neunten Ebene differenziert er in einem Fall die Applikationsrate nochmals zwischen 30 kg N/ha und 40 kg N/ha. Auf dritter Ebene kommt der Einfluss des Attributs *N_1* ins Spiel und wirkt auch auf Ebene vier, sechs und sieben in unterschiedlichen Ästen mit. Die Zugkraft *Tforce* trägt ebenfalls bereits ab Ebene drei zur Entscheidungsfindung bei und hat Auswirkungen auf den folgenden vier Ebenen. Auf vierter Ebene entscheidet auch erstmals der Ertragswert *Yield98* über den Entscheidungsweg mit, trägt dann aber in

unterschiedlichen Ästen auf den Ebenen Vier bis Acht zur Entscheidungsfindung bei. Entsprechend dem vorliegenden Entscheidungsbaum ist das (Bodeneigenschafts-)Attribut *EM38* nahezu vernachlässigbar, da es nur einmal auf der sechsten Ebene zur Ableitung des Applikationswertes herangezogen wird und hierbei über eine Differenz von 20 kg N/ha (80 bzw. 100 kg N/ha) bestimmt.

Tabelle 5.1 listet die Häufigkeitsverteilung der einzelnen diskreten Applikationsraten-Sollwertempfehlungen für diesen Entscheidungsbaum in der zugehörigen Spalte auf. Dabei wird deutlich, dass bei insgesamt 34 Empfehlungen jeder Sollwertschritt von 0 kg N/ha bis 100 kg N/ha vorkommt. Der Bereich von 20 kg N/ha bis 70 kg N/ha ist dabei am häufigsten vertreten mit dem „Spitzenwert“ 60 kg N/ha, der allein sechsmal empfohlen wird.

Tabelle 5.1: Häufigkeitsverteilung von Sollwert-Empfehlungen für die beiden Entscheidungsbäume mit und ohne das Attribut REIP_2

Sollwertempfehlung für Applikationsrate [N kg/ha]	Häufigkeit im Entscheidungsbaum mit dem Attribut REIP_2	Häufigkeit im Entscheidungsbaum ohne das Attribut REIP_2
0	2	7
10	1	2
20	4	3
30	5	4
40	5	1
50	3	1
60	6	1
70	3	0
80	1	2
90	1	1
100	3	1
Summierte Anzahl:	34	23

Verglichen mit dem dreistufigen Entscheidungsbaum der Abbildung 5.5 fällt auf, dass trotz etwas verschiedener Entscheidungsschwellwerte die Entscheidung auf den ersten beiden Ebenen anhand *REIP_2* recht ähnlich verläuft und der Ast mit den kleineren *REIP_2*-Werten auf eine Entscheidung zuläuft, die auf der nächsten Stufe von dem Attribut *N_1* im Gegensatz zu dem Attribut *Tforce* abhängt. In der groben Tendenz führt jedoch der Ast mit dem niedrigsten *REIP_2*-Wert zu den höchsten Applikationswerten von 80 kg N/ha - 100 kg N/ha. Auch der Ast mit dem nächsthöheren *REIP_2*-Wertebereich führt tendenziell eher zu den

höheren Applikationsraten im Bereich von 30 kg N/ha - 100 kg N/ha, während der Ast mit dem höchsten *REIP_2*-Wertebereich zu den eher niedrigeren Applikationsraten im Bereich von 0 kg N/ha - 60 kg N/ha mit dreimaligen Auftreten einer Empfehlung zur Null-Applikation führt. Von der groben Tendenz her deckt sich dies mit dem noch interpretationsfähigen dreistufigen Entscheidungsbaum, eine tiefer gehende „Musteruntersuchung“ bzw. pflanzenbauliche Interpretation wird dem entsprechenden Experten überlassen. Die vorliegende Arbeit ist mehr auf die technische Umsetzung fokussiert und verlässt sich auf das von den Experten in den anderen IKB-Dürnast-Teilprojekten bereitgestellte Wissen.

Im Kapitel zum *Problem Solving Level* wurde bereits darauf hingewiesen, dass auch ein alternatives Regelwerk innerhalb des „*CROP_PRODUCTION“*-Moduls zum Einsatz kam, das ohne die Berücksichtigung des REIP-Wertes eine Sollwertempfehlung ableitet. Hierfür wurde im Rahmen des IKB-Dürnast Teilprojektes 12 der beschriebene WED-Prozess für die zweite Stickstoffgabe für das Versuchsfeld D4 der TU München des Versuchsjahres 2003 diesmal ohne das Attribut *REIP_2* durchgeführt. Der daraus resultierende Entscheidungsbaum liegt dieser Arbeit wiederum in Form einer Abbildung (Abb. A.3) im Anhang bei. Für die Erklärungs- und Dokumentationsfunktionalität wurde den Blättern wieder eine Identifikationsbeschreibung zugeordnet, die um das Kürzel „- noREIP2“ ergänzt wurde.

Auch dieser Entscheidungsbaum verfügt über eine Tiefe von maximal neun Ebenen. Ein Blick auf die Häufigkeit der Attribute zeigt, dass *N_1* mit achtmaligem Auftreten am meisten vertreten ist. Mit siebenmaligem Auftreten folgt das Attribut *Tforce,* das Attribut *Yield98* ist fünfmal vertreten und das Attribut *EM38* wird noch zweimal zur Entscheidung herangezogen. Auf den ersten drei Ebenen wird die Entscheidungsfindung von dem Attribut *N_1* dominiert. Auch auf der vierten Ebene hat *N_1* entscheidenden Einfluss auf die Sollwert-Empfehlung für zwei Äste. Ebenfalls auf Ebene vier trägt erstmals das Attribut *Tforce* zur Entscheidungsfindung bei. Der Einfluss dieses Attributs ist auch noch auf der fünften, sechsten und achten Ebene wirksam. *EM38* tritt das erste Mal auf der vierten Ebene auf und unterscheidet auf der siebten Ebene zwischen den beiden Applikationsraten von 20 kg N/ha und 30 kg N/ha. Erst ab der fünften Ebene wird das Attribut *Yield98* in Entscheidungsprozesse eingebunden und differenziert auf dieser Ebene einmal zwischen 80 kg N/ha und 90 kg N/ha und das andere Mal zwischen 0 kg N/ha und 10 kg N/ha. Auf der sechsten, der siebten und der neunten Ebene ist *Yield98* hauptsächlich in zwei Ästen wirksam. Tabelle 5.1 listet auch für diesen Entscheidungsbaum die Häufigkeitsverteilung der einzelnen diskreten Applikationsraten-Sollwertempfehlungen in der zugehörigen Spalte auf. Im Gegensatz zum Entscheidungsbaum mit dem Attribut *REIP_2* wird diesmal bei insgesamt 23

Empfehlungen nicht jeder Sollwertschritt von 0 kg N/ha bis 100 kg N/ha abgedeckt. Der Wert 70 kg N/ha kommt nicht vor. Es ist auch auffällig, dass der Bereich von 0 kg N/ha bis 30 kg N/ha am häufigsten vertreten ist. Allein die „Nullwert-Applikation“ 0 kg N/ha ist mit siebenmaligem Vorkommen sooft vorhanden wie alle Empfehlungen für den Sollwertbereich von 40 kg N/ha bis 100 kg N/ha zusammen.

Auch bei diesem Entscheidungsbaum wird eine weiterführende „Musteruntersuchung“ bzw. pflanzenbauliche Interpretation und gegebenenfalls eine entsprechende Anpassung den pflanzenbaulichen Experten überlassen. Gewisse Tendenzen sollen dennoch kurz skizziert werden. In der groben Tendenz führt der Hauptast mit den niedrigsten *N_1* Werten zu den höchsten Applikationswerten von 80 kg N/ha - 100 kg N/ha. Die bei der gegebenen Häufigkeitsverteilung ebenfalls höheren Applikationsraten von 50 kg N/ha – 60 kg N/ha sind in letzter Konsequenz ein Resultat der etwas höheren *N_1* Werte. Gemäß dem Entscheidungsbaum ist der Bereich mit *N_1* -Werten größer 50 kg N/ha verantwortlich für die Abdeckung der Sollwert-Empfehlungen des niedrigeren Wertebereiches von 0 kg N/ha - 40 kg N/ha. Die Differenzierung erfolgt dabei über die Attribute *Tforce*, *Yield98* und *EM38* auf der vierten bis neunten Entscheidungsebene.

5.2.2.4.2 Modul *„CROP_PRODUCTION-SUM_UP”*

Das Entscheidungsbaumverfahren führt zu einer eindeutigen Klassifikation. Da jedoch die mit diesem Verfahren abgeleiteten Entscheidungsregeln nach ihrer maschinellen Erzeugung von Experten begutachtet, überarbeitet und erweitert werden können, kann dies dazu führen, dass die Eindeutigkeit nicht mehr stets gegeben ist und konkurrierende Sollwertempfehlungen im Lauf des Inferenz-Prozess „aktiv“ werden. Somit wurde wie im *Problem Solving Level* bereits angeführt, in dem nachgeschalteten Wissensmodul „*CROP_PRODUCTION-SUM_UP“* prozedurales Wissen für den Umgang mit dem Auftreten mehrerer Sollwertempfehlungen oder dem Fehlen einer Sollwertempfehlung zusammengestellt.

Aus der Menge der theoretisch null bis n möglichen Zwischenergebnisse vom Typ *CROP_PRODUCTION::N_recommendation (setpoint*, *rule-ID*) wird in der Simulation nach dem schlichten Prinzip vorgegangen, dass die höchste Sollwertempfehlung in diesem Konfliktfall gewinnt. Für den Fall, dass der gleiche Applikations-Sollwert mehrfach auftritt, werden die verkürzten Erklärungstexte alle aneinandergereiht (*explanation concatenation*). Bereits im „*CROP_PRODUCTION*“-Hauptmodul wurde der Fall abgefangen, dass keine Sollwertempfehlung auf Basis der Entscheidungsregeln gefunden wird. Hier tritt ein knappes Regelwerk in Aktion, welches den Default-Wert *CROP_PRODUCTION::N_recommendation*

(default-value, „N0 - default“) als Sollwertempfehlung für die weitere Problemlösung einbringt. Als Default-Wert wurde die Null-Applikation 0 N kg/ha festgelegt.
Diese grundlegenden Prinzipien des angewandten Lösungsweges und die zugehörigen Wissenselemente der beiden Module „*CROP_PRODUCTION“* und „*CROP_PRODUCTION-SUM_UP“* wurden in Form von beschrifteten Baumdiagrammen, sogenannten *Mind-Maps* [BB02], skizziert und für die Erklärungs- und Dokumentationsfunktionalität genutzt.

5.2.2.4.3 Modul *„CONSTRAINTS“*

Das weitere Modul „*CONSTRAINTS*“ wurde bereits im Kapitel des *Problem Solving Levels* mit seinen drei Funktionskategorien eingeführt. Die Berücksichtigung von Limitierungen, die Vermeidung von Überapplikation aufgrund mehrfacher Applikation auf der vorliegenden Teilfläche sowie die Einbringung der Übersteuerungswünsche des Nutzers bzw. weiterer Systemteilnehmer wurden ebenfalls in regelbasierter Form formuliert. Die indirekte Wissensakquisition wurde wesentlich einfacher als im „*CROP_PRODUCTION“*-Modul durchgeführt. Der Autor agierte dabei in Personalunion als Wissensingenieur und zugleich als Fachexperte. Das Wissen wurde hierfür vor allem aus den Vorgaben bzw. Ideen des Funktionalen Modells abgeleitet und in eine regelbasierte Form übertragen. Analog zu dem pflanzenbaulichen Wissen wurde auch für diese Wissenselemente mit den „*CONSTRAINTS“*- und „*CONSTRAINTS–SUM_UP“*-Modulen zweistufig vorgegangen. Während im Hauptmodul für die einzelnen Kategorien die möglichen Zwischenergebnisse abgeleitet, aufbereitet und gesammelt werden, werden sie im „-*SUM_UP*“-Modul wiederum aggregiert und führen jeweils nur zu einem einzigen Zwischenergebnis für die einzelne Kategorie. Dementsprechend werden in der Kategorie der Limitierungen alle Sollwertbeschränkungen aufgesammelt, die die Beschränkung des Applikations-Sollwertes empfehlen. Da in den Arbeiten zu den IKB-Dürnast-Teilprojekten 12 und 13 keine spezifischen Begrenzungsfunktionen definiert wurden, wurde entsprechend der funktionalen Modellierung als einfacher Ansatz die „Hintergrundkarten“-Information „*environmental limit*“, also eine teilflächenspezifisch festgelegte Beschränkungsfunktion zur Berücksichtigung des Umweltschutzes, definiert. In der Praxis treten diese Fälle zum Beispiel als Randflächen oder Abstandflächen zu Gewässern oder Vorgaben einer Höchstausbringmenge auf. Rein informationstechnisch gesehen, ist die Herkunft der Limitierung auf die eigentliche Anwendung jedoch unerheblich, entscheidend ist, dass das Konzept die Integration von keiner bis zu mehreren unterschiedlichen „Überlagerungskarten“ mit limitierenden Werten vorsieht. In der Kategorie der Überlagerungskarten mit Grenzwerten werden als Zwischenergebnis

folglich keine oder mehrere neue Fakten vom Typ *„CONSTRAINTS::N_recommendation (setpoint = limit-value, explanation = „<type of limit> restricts nitrogen application to:“ limit-value „ kg N/ha“)* in den gesamten Schlussfolgerungsprozess eingebracht. Der limitierende *setpoint* Wert wird als numerischer Wert *limit-value* angegeben, der in der Maßeinheit kg N/ha zu interpretieren ist. Der zugehörige Erklärungstext setzt sich aus der Beschreibung der beschränkenden Kategorie, d.h. „*environmental limit*“ im vorliegenden Fall, und dem Textbaustein „*restricts nitrogen application to:“* gefolgt von dem *limit-value*-Wert und dem abschließenden Textbaustein zur Angabe der Maßeinheit zusammen.

Für die zweite Kategorie lag der Schwerpunkt auf der Berücksichtigung von bereits erfolgter Stickstoffapplikation zur gleichen Stickstoffgabe, d.h. zur zweiten Gabe. Im Rahmen der Implementierung und Dokumentationsfunktionalität wurde dies als „*repeated application/overlapping*“ bezeichnet. Das zugrundeliegende Wissen lässt sich umgangssprachlich sehr einfach formulieren:

> Prüfe, ob auf der aktuellen Teilfläche bereits Stickstoffdünger zur zweiten Gabe appliziert wurde. Wenn dies Fall ist, so erzeuge einen Fakt, der die bereits applizierte Menge beinhaltet und mit einer entsprechenden Kurzbeschreibung versehen ist! Generiere auch für den Fall, dass noch keine Applikation durchgeführt wurde, einen Fakt, der die Menge 0 kg N/ha und einen entsprechenden Erklärungstext beinhaltet!

Mit der dritten Kategorie zur Integration von Übersteuerungswünschen des Nutzers bzw. weiterer Systemteilnehmer wird ein bisher noch nicht genutztes „Stilmittel“ eingeführt, das die bisherige Form der Zwischenergebnisse bzw. Fakten, bestehend aus einem Applikationssollwert, -grenzwert und einer kurzen Texterklärung, um eine Prioritätsangabe erweitert. Diese Prioritätsangabe ist jedoch nicht zu verwechseln mit der Expertensystemfunktionalität auf Ebene des *Problem Solving Level*, Konfliktlösungen über Prioritätszuweisungen (*salience*) für einzelne Regeln zu bewirken. Diese zusätzliche Prioritätszuweisung soll dem Gedanke Rechnung tragen, dass es Prozessführungs-Systemarchitekturen mit einer Hierarchie von über- oder untergelagerten Systemteilnehmern oder Regelkreisen gibt.

Bei der vorliegenden Arbeit beschränkt sich die Funktionalität der dritten Kategorie auf die Übersteuerungsmöglichkeit des Nutzers über die „*In-field Controller”*-GUI. Für diese Zwecke besteht das Wissen wiederum in einer simplen Beschaffungs-, Aufbereitungs- und Einspeisefunktion für weitere Schlussfolgerungsprozesse im Rahmen der „*In-field Controller”*-Prozessführung. Der generierte Fakt folgt der Form *„CONSTRAINTS::Adjustment_factor (factor = user-relative-value, explanation = „Adjustment factor due <type of adjustment>:“ user-relative-value „ [priority: neutral]“,*

priority = neutral). *Factor* nimmt dabei den numerischen dimensionslosen Wert für einen relativen (bzw. prozentualen) Übersteuerungswunsch an, der über eine andere „*In-field Controller"*-Funktionalität dem Wissensmodul vergleichbar einem Online-Sensorik-Wert in stets aktueller Form zur Verfügung gestellt werden muss. Der zugehörige Erklärungstext beginnt mit der Nennung der Kategorie und dem Typus für den Änderungswunsch. Dem folgen die Angabe des ermittelten Änderungsfaktors und die Angabe der Priorität. Die Angabe dieser Priorität ist der abschließende Teil des generierten Fakts. In der vorliegenden Arbeit wurde dem Nutzerwunsch die Priorität „neutral" zugeordnet.

5.2.2.4.4 Modul *„CONSTRAINTS-SUM_UP"*

In diesem Wissensmodul wurde das prozedurale Wissen zu einer Auswahlfunktion bei mehreren Vorschlägen für eine Applikationssollwertbegrenzung, der Berücksichtigung von priorisierten relativen Änderungswünschen bezüglich der Sollwertempfehlung und der schlichten Fakten-Bereitstellung über bereits durchgeführte Applikation(en) in regelbasierter Form zusammengestellt.

Der Auswahlprozess zur Bestimmung der Sollwertlimitierung *„CONSTRAINTS-SUM_UP::N_recommendation"* ist die Suche nach dem minimalen Wert aus der Menge aller Zwischenergebnisse vom Typ *„CONSTRAINTS::N_recommendation"*. Sollte der minimale Begrenzungswert mehrfach auftreten, so wird über die Aneinanderreihung der verkürzten Erklärungstexte („*explanation concatenation*") gewährleistet, dass diese Information über die Herkunft der Begrenzungen der Dokumentation- bzw. Erklärungsfunktionalität zur Verfügung gestellt werden kann. Dieser beschriebene Auswahlprozesses lässt sich mit nur drei Regeln realisieren, wie nachfolgend in „*Pseudo-Code*"-Format gezeigt:

REGEL CONSTRAINTS-SUM_UP „Sammlung aller Sollwertlimitierungs-Zwischenergebnisse und Formatwandlung"

WENN

Eine Limitierung CONSTRAINTS::N_recommendation (setpoint, explanation) in der Faktenbasis vorliegt

DANN

Entferne diese Limitierung CONSTRAINTS::N_recommendation (setpoint, explanation) aus der Faktenbasis

Füge CONSTRAINTS-SUM_UP::N_recommendation (setpoint, explanation) neu zur Faktenbasis hinzu

REGEL CONSTRAINTS-SUM_UP „Auswahl der Empfehlung mit der niedrigsten Sollwertlimitierung"

WENN

Eine Limitierung CONSTRAINTS-SUM_UP::N_recommendation (setpoint1, explanation1) in der Faktenbasis vorliegt

Eine andere Limitierung CONSTRAINTS-SUM_UP::N_recommendation (setpoint2, explanation2) in der Faktenbasis vorliegt

Der Wert von setpoint1 kleiner als der Wert von setpoint2 ist

DANN

Entferne die Limitierung CONSTRAINTS-SUM_UP::N_recommendation (setpoint2, explanation2) aus der Faktenbasis

REGEL CONSTRAINTS-SUM_UP „Aneinanderreihung von Erklärungstexten bei gleichem Begrenzungswert"

WENN

Eine Limitierung CONSTRAINTS-SUM_UP::N_recommendation (setpoint1, explanation1) in der Faktenbasis vorliegt

Eine andere Limitierung CONSTRAINTS-SUM_UP::N_recommendation (setpoint2, explanation2) in der Faktenbasis vorliegt

Der Wert von setpoint1 gleich dem Wert von setpoint2 ist

Der Erklärungstext von explanation1 stimmt nicht mit dem von explanation2 überein

DANN

Entferne die Limitierung CONSTRAINTS-SUM_UP::N_recommendation (setpoint1, explanation1) aus der Faktenbasis

Entferne die Limitierung CONSTRAINTS-SUM_UP::N_recommendation (setpoint2, explanation2) aus der Faktenbasis

Füge CONSTRAINTS-SUM_UP::N_recommendation (setpoint1, (explanation1 + explanation2) neu zur Faktenbasis hinzu

Für den Fall, dass kein einziger Sollwert-Limitierungs-Fakt aus dem „*CONSTRAINTS“*-Modul vorliegt, wird auch kein Default-Wert erzeugt und weitergereicht. Diese Situation wird im Wissensmodul „*SUM_UP“* adressiert.

Für die Berücksichtigung der zweiten Kategorie einer möglicherweise bereits vorliegenden Stickstoffapplikation zur gleichen Stickstoffgabe wird prinzipiell nur eine Regel zur Formatumwandelung angewandt. Dazu wird das Zwischenergebnis in die Form *CONSTRAINTS-SUM_UP::N_applied (amount = applied-value, explanation = applied-value „N kg/ha were already applied for N2“)* gebracht. Dabei ist *applied-value* ein numerischer Wert entsprechend der bereits ausgebrachten Düngermenge bezogen auf die Maßeinheit kg N/ha. Der zugehörige Erklärungstext ist in der deutschen Übersetzung selbsterklärend. In der vorliegenden Arbeit wird davon ausgegangen, dass stets ein *CONSTRAINTS-SUM_UP::N_applied*-Fakt vorliegt, dessen Voreinstellung (Default) mit der Null-Applikation 0 kg N/ha vorbelegt ist.

Die dritte Kategorie zur Integration von Übersteuerungswünschen des Nutzers bzw. weiterer Systemteilnehmer erfordert ein etwas erweitertes Regelwerk im Vergleich zu den ersten beiden Kategorien. Als Endergebnis gilt es hierzu einen Fakt der Form *CONSTRAINTS-SUM_UP::Adjustment_factor (factor = relative-value, explanation = „Adjustment factor due*

<type of adjustment>:“ relative-value „ [priority: p-text]“, priority) bereit zu stellen. Die Erklärung der Parameter wurde bereits im Hauptmodul gegeben. Nur bei der Prioritätsangabe wird auch von anderen Prioritätseinstufungen als „neutral“ ausgegangen. So wurden *very high, high, neutral, low* und *very low* definiert. Auch wenn für die vorliegende Simulation nur der Übersteuerungswunsch der Bedienperson genutzt wird, ist das Regelwerk auf mehrere Quellen von Übersteuerungsanfragen ausgelegt. Dazu wird nach den folgenden Grundsätzen verfahren. Zuerst werden alle „*CONSTRAINTS::Adjustment_factor“*-Fakten in „*CONSTRAINTS-SUM_UP::Adjustment_factor”*-Fakten gewandelt. Der Fakt „*CONSTRAINTS-SUM_UP::Adjustment_factor”* mit der höchsten Priorität setzt sich als bestimmender Faktor durch. Liegen mehrere Fakten mit der höchsten Priorität vor, so wird nach dem Fakt mit dem niedrigsten Übersteuerungswunsch als Kompromisslösung vom Typ „kleinster gemeinsamer Nenner“ gesucht. Der Fall, dass Fakten mit gleicher Priorität und gleichem Übersteuerungsfaktor vorliegen, resultiert wiederum in dem Ersatz dieser Fakten durch einen neuen Fakt mit gleichem Prioritätswert und Übersteuerungsfaktor, jedoch einer Aneinanderreihung der verkürzten Erklärungstexte („*explanation concatenation*“) der Ursprungsfakten. Die Umsetzung dieser Vorgehensweise lässt sich über vier Regeln formulieren. Die erste Regel folgt dem Bespiel, wie sie für die *REGEL CONSTRAINTS-SUM_UP „Sammlung aller Sollwertlimitierungs-Zwischenergebnisse und Formatwandlung*" bereits gezeigt wurde, nur angepasst auf die Fakten *CONSTRAINTS::Adjustment_factor* und *CONSTRAINTS-SUM_UP::Adjustment_factor.* Auch die zweite Regel folgt dem Prinzip wie es in der *REGEL CONSTRAINTS-SUM_UP „Auswahl der Empfehlung mit der niedrigsten Sollwertlimitierung*“ bereits gezeigt wurde, nur dass diesmal im Aktionsteil der Regel festgelegt ist, den Fakt mit dem niedrigerem Prioritätswert zu entfernen. Die Suche nach dem Fakt mit dem niedrigsten Übersteuerungsfaktor bei Vorliegen von Fakten mit gleich hoher Priorität wird über folgende Regel bewerkstelligt.

REGEL CONSTRAINTS-SUM_UP „Adjustment factor – Bei gleicher Priorität wird der niedrigste Übersteuerungsfaktor ausgewählt"

WENN

Ein Übersteuerungswunsch CONSTRAINTS-SUM_UP::Adjustment_factor (factor1, explanation1, priority1) in der Faktenbasis vorliegt

Ein anderer Übersteuerungswunsch CONSTRAINTS-SUM_UP::Adjustment_factor (factor2, explanation2, priority2) in der Faktenbasis vorliegt

Der Wert von priority1 gleich dem Wert von priority2 ist

Der Wert von factor1 kleiner als der Wert von factor2 ist

Die Erklärungen explanation1 und explanation2 sich unterscheiden

DANN

Entferne den Übersteuerungswunsch CONSTRAINTS-SUM_UP::Adjustment_factor (factor2,

explanation2, priority2) aus der Faktenbasis

Den möglichen Sonderfall, dass Fakten mit gleicher Priorität und gleichem Übersteuerungsfaktor vorliegen, löst folgende Regel auf:

REGEL CONSTRAINTS-SUM_UP „Adjustment factor – Bei gleicher Priorität und bei gleichem Übersteuerungsfaktor werden die Erklärungstexte aneinandergereiht"

WENN

Ein Übersteuerungswunsch CONSTRAINTS-SUM_UP::Adjustment_factor (factor1, explanation1, priority1) in der Faktenbasis vorliegt

Ein anderer Übersteuerungswunsch CONSTRAINTS-SUM_UP::Adjustment_factor (factor2, explanation2, priority2) in der Faktenbasis vorliegt

Der Wert von priority1 gleich dem Wert von priority2 ist

Der Wert von factor1 gleich dem Wert von factor2 ist

Die Erklärungen explanation1 und explanation2 sich unterscheiden

DANN

Entferne CONSTRAINTS-SUM_UP::Adjustment_factor (factor1, explanation1, priority1) aus der Faktenbasis

Entferne CONSTRAINTS-SUM_UP::Adjustment_factor (factor2, explanation2, priority2) aus der Faktenbasis

Füge CONSTRAINTS-SUM_UP::Adjustment_factor (factor1, explanation1 + explanation2, priority1) neu zur Faktenbasis hinzu

Abschließend sei noch bemerkt, dass davon ausgegangen wird, dass die Eingabe durch die Bedienperson stets erfasst wird und der Normalzustand dabei die Nullstellung bzw. die „100 %"-Einstellung ist. Eine Behandlung eines fehlenden Fakts ist daher nicht zu beachten.
Wie bereits bei den „*CROP_PRODUCTION"*-Wissensmodulen wurden die gerade beschriebenen Prinzipien und Wissenselemente der beiden „*CONSTRAINTS"*-Module wiederum in Form von *Mind-Maps* skizziert und in die Erklärungs- und Dokumentationsfunktionalität der Simulation eingebunden.

5.2.2.4.5 Module *„AG_ENGINEERING"* und *„AG_ENGINEERING-SUM_UP"*

Das Wissen des „*AG_ENGINEERING*"-Modules wurde im *Problem Solving Level* in prozeduraler Weise bereits überblicksartig dargestellt. Das entsprechende Wissen wurde mittels einer indirekten Wissensakquisition erarbeitet, wobei der Autor wiederum als Wissensingenieur und zugleich als Fachexperte in Personalunion fungierte. Auch in diesem Fall wurde das Wissen vor allem aus den Vorgaben bzw. Ideen des Funktionalen Modells abgeleitet, mit den Kollegen am Fachgebiet Technik im Pflanzenbau der TU München diskutiert und anschließend regelbasiert formuliert. Die zweistufige Vorgehensweise mit einem *„AG_ENGINEERING"*- und einem *„AG_ENGINEERING-SUM_UP"*-Modul kommt auch hierbei wieder zum Einsatz. Im *„AG_ENGINEERING"*-Modul wird wie bei einem

Online-Sensor-Wert regelmäßig der Zustand des Schalters zur Auswahl zwischen manuellem und automatischem Modus abgefragt und zu einem Fakt *AG_ENGINEERING::Control_Mode (mode, explanation)* aufbereitet. Je nach Schalterzustand[43] nimmt die Variable *mode* den Wert „*manually*“ oder „*automatically*“ und die Variable *explanation* „*selected control mode is manually”* bzw. „*selected control mode is automatically*“ an. Für die Zeitpunkte, in denen sich der Nutzer für den manuellen Arbeitsmodus entscheidet, wird ein vom Nutzer gewählter Sollwert für die Applikationsmenge als Fakt benötigt. Dazu wird ein *AG_ENGINEERING::N_recommendation (setpoint = user-absolute-value, explanation = user-absolute-value + „ kg N/ha commanded by user“)* eingeführt, der die Eingabe der Bedienperson über ein Eingabeelement der MMI aufbereitet.

Im *Problem Solving Level* wurde die prinzipielle Möglichkeit eines umfassenderen Ansatzes für die ebenfalls zu diesem Wissensmodul gehörende Zustandsbewertung der Gesamtsteuerung und im Speziellen der Online-Sensorik und der Applikationstechnik hinsichtlich der Einhaltung von Genauigkeit und Echtzeitfähigkeit ausgeführt. Jedoch beschränkt sich für den vorliegenden Ansatz diese Zustandsbewertung auf die Angabe, ob ein Online-Sensor zur REIP Bestimmung vorhanden ist und dass sein mögliches Fehlen zumindest als Fakt für die Erklärungs- bzw. Dokumentationsfunktionalität in der Wissensbasis hinterlegt wird. Dazu wird eine Kategorie von Fakten des Typs *AG_ENGINEERING::Quality_statement (quality, explanation)* erzeugt. Für die Unterscheidung, ob ein Online-Sensor zur Bestimmung des Vegetationsindex REIP vorhanden ist oder nicht, nimmt die Variable *quality* den Text „yesREIP“ als Wert an, im gegensätzlichen Fall „noREIP“. Der zugehörige kurze Textbaustein für die Erklärungsfunktionalität, zusammengefasst in der Variable *explanation*, lautet „*Sensor for REIP 2 value is working*“ bzw. „*.... not working*“. Für die Zustände der Steuerung und der Applikationstechnik wird die Variable *quality* mit „NULL“ belegt und entsprechend die Variable *explanation* mit dem Textbaustein „*Process control / Implement: not enough information for conclusion and assessment*“.

Im nachgeschalteten „*AG_ENGINEERING-SUM_UP“*-Wissensmodul wird das bereits bewährte Schema der Sammlung und Formatwandlung der Fakten aus dem „*AG_ENGINEERING“*-Modul durchgeführt. Die regelbasierte Formulierung wurde bereits im Abschnitt zu den „*CONSTRAINTS*“-Modulen skizziert. Weiterhin werden alle *AG_ENGINEERING::Quality_statement*-Fakten in einem einzigen Fakt *AG_ENGINEERING-*

[43] Aus Sicherheitsüberlegungen gilt der manuelle Modus als Default-Einstellung, solange vom Nutzer keine andere Einstellung getroffen wird.

SUM_UP::Quality_statement (quality, explanation) zusammenfasst. Dies lässt sich in einem Produktionssystem wiederum mit nur einer Regel bewerkstelligen, selbst für eine wesentlich größere Anzahl von Fakten der zugehörigen Klasse:

REGEL AG_ENGINEERING-SUM_UP „Zusammenfassen aller Qualitätsaussagen (Quality Statements)"

WENN

Eine Qualitätsaussage AG_ENGINEERING-SUM_UP::Quality_statement (quality1, explanation1) in der Faktenbasis vorliegt

Eine Qualitätsaussage AG_ENGINEERING-SUM_UP::Quality_statement (quality2, explanation2) in der Faktenbasis vorliegt

Die beiden obigen Fakten sich in mindestens einer der beiden Variablen unterscheiden

DANN

Entferne die Qualitätsaussage AG_ENGINEERING-SUM_UP::Quality_statement (quality1, explanation1) aus der Faktenbasis

Entferne die Qualitätsaussage AG_ENGINEERING-SUM_UP::Quality_statement (quality2, explanation2) aus der Faktenbasis

Füge AG_ENGINEERING-SUM_UP::Quality_statement ((quality1 + quality2), (explanation1 + explanation2)) neu zur Faktenbasis hinzu

Auch bei diesem Modul wurden die beschriebenen Prinzipien und Wissenselemente in Form von *Mind-Maps* skizziert, um sie auch zugleich in die Erklärungs- und Dokumentationsfunktionalität der Simulation einbinden zu können.

5.2.2.4.6 Modul „SUM_UP“

In dem vierten und abschließenden Schritt des Situationsbewertungszyklus werden in dem Wissensmodul „*SUM_UP*“ die Zwischenergebnisse der vorgelagerten Wissensmodule gesammelt, bewertet und letztendlich daraus die aktuelle N-Applikationssollwertempfehlung abgeleitet. Eine zweistufige Vorgehensweise ist bei diesem Wissensmodul nicht nötig. Abermals wurde bei der Wissensakquisition in der Form vorgegangen, dass der Autor als Wissensingenieur und Experte fungierte.

Als Eingangsinformation liegen die nachfolgend aufgelisteten Zwischenergebnisse der bereits beschriebenen Module als Fakten vor:

- *CROP_PRODUCTION-SUM_UP::N_recommendation (setpoint, explanation),*
- *CONSTRAINTS-SUM_UP::N_recommendation (setpoint, explanation),*
- *CONSTRAINTS-SUM_UP::N_applied (amount, explanation),*
- *CONSTRAINTS-SUM_UP::Adjustment_factor (factor, explanation, priority),*
- *AG_ENGINEERING-SUM_UP::N_recommendation (setpoint, explanation),*
- *AG_ENGINEERING-SUM_UP::Control_Mode (mode, explanation),*
- *AG_ENGINEERING-SUM_UP::Quality_statement (quality, explanation).*

Daraus wird die zusammenfassende Sollwertempfehlung für die Applikationsrate *N2-Setpoint* (in kg N/ha) unter Zuhilfenahme des Fakts *SUM_UP::N_recommendation (setpoint, explanation)* abgeleitet. Dabei wird die folgende Vorgehensweise in regelbasierter Form umgesetzt. Je nach eingestelltem Modus des *AG_ENGINEERING-SUM_UP::Control_Mode* unterscheidet sich das weitere Vorgehen.

Wurde der manuelle Modus gewählt, so wird (als einzige Aktion) die Applikationsrate *N2-Setpoint* direkt als Ergebnis der Multiplikation von *AG_ENGINEERING-SUM_UP::N_recommendation (setpoint)* und *CONSTRAINTS-SUM_UP::Adjustment_factor (factor)* bestimmt. Es wurde für diesen Fall vorausgesetzt, dass der Nutzer sich um die Einhaltung von Sollwerteinschränkungen selbstverantwortlich sorgt.

Wurde jedoch der automatische Modus ausgewählt, so stellt sich die Situation anders dar. Dabei ist zu prüfen, ob eine Sollwertbeschränkung *CONSTRAINTS-SUM_UP::N_recommendation (setpoint, explanation)* vorliegt oder nicht. Ist keine Beschränkung vorhanden, so wird ein Fakt *SUM_UP::N_recommendation (setpoint, explanation)* erzeugt, dessen Variable *setpoint* den Ergebniswert der Multiplikation von *CROP_PRODUCTION-SUM_UP::N_recommendation (setpoint)* und *CONSTRAINTS-SUM_UP::Adjustment_factor (factor)* annimmt. Der Variable *explanation* wird der Textbaustein „*no constraints“* und die Erklärungstexte der an der Entscheidung beteiligten Fakten *CROP_PRODUCTION-SUM_UP::N_recommendation*, *CONSTRAINTS-SUM_UP::Adjustment_factor* und *AG_ENGINEERING-SUM_UP::Control_Mode* zugewiesen.

Für den Fall, dass eine Beschränkung vorliegt, werden zwei unterschiedliche Entscheidungen getroffen. Sollte das Produkt aus *CROP_PRODUCTION-SUM_UP::N_recommendation (setpoint)* und *CONSTRAINTS-SUM_UP::Adjustment_factor (factor)* kleiner oder gleich dem Wert der Variable *setpoint* von *CONSTRAINTS-SUM_UP::N_recommendation* sein, so resultiert dies in einem Fakt *SUM_UP::N_recommendation (setpoint, explanation)*, dessen Variable *setpoint* den Ergebniswert der Multiplikation von *CROP_PROCUCTION-SUM_UP::N_recommendation (setpoint)* und *CONSTRAINTS-SUM_UP::Adjustment_factor (factor)* annimmt. Der Variable *explanation* werden alle Erklärungstexte der an der Entscheidung beteiligten Fakten zugewiesen. Ist jedoch das Produkt aus *CROP_PRODUCTION-SUM_UP::N_recommendation (setpoint)* und *CONSTRAINTS-SUM_UP::Adjustment_factor (factor)* größer als der Wert der Variable *setpoint* von *CONSTRAINTS-SUM_UP::N_recommendation*, so wird ein Fakt *SUM_UP::N_recommendation (setpoint, explanation)* generiert, dessen Variable *setpoint* den

Wert der Limitierung *CONSTRAINTS-SUM_UP::N_recommendation (setpoint)* annimmt. Auch hier werden in der Variable *explanation* alle Erklärungstexte der an der Entscheidung beteiligten Fakten gesammelt.

Somit fehlt nur noch die Berücksichtigung einer möglicherweise bereits ausgebrachten Düngermenge, zusammengefasst und als Fakt dem Inferenz-Prozess zur Verfügung gestellt in Form von *CONSTRAINTS-SUM_UP::N_applied*. Für den Fall, dass der manuelle Modus gewählt wurde, wird diesem Fakt keine Beachtung geschenkt aufgrund der Annahme, dass der Nutzer selbstverantwortlich handelt. Im automatischen Modus hingegen ist eine Fallunterscheidung zu treffen, ob eine bereits ausbrachte Düngermenge den vorläufig empfohlenen Applikationssollwert gemäß *SUM_UP::N_recommendation (setpoint)* übersteigt bzw. gleich ist. Ist dies gegeben, so darf keine weitere Applikation mehr erfolgen und der endgültig empfohlene Sollwert für die zweite Stickstoffgabe ist 0 kg N/ha für *N2-Setpoint*. Für den Erklärungstext wird der zugehörigen Variablen der Textbaustein „*Further N-application blocked due to already applied amount of nitrogen*“ zugewiesen sowie auch die Erklärungstexte der Fakten *SUM_UP::N_recommendation* und *CONSTRAINTS-SUM_UP::N_applied* mit angehängt.

Übersteigt jedoch die empfohlene Sollwertvorgabe von *SUM_UP::N_recommendation* die bereits ausgebrachte Menge, so errechnet sich die endgültige Applikationsrate *N2-Setpoint* als Differenz mit dem Wert der Variable *amount* des Fakts *CONSTRAINTS-SUM_UP::N_applied*. Der Erklärungstext setzt sich dann aus dem Textbaustein „SUM-UP: *Nitrogen application setpoint: .... kg N/ha*“ mit dem errechneten Differenzwert für den durch die Punkte angedeuteten Platzhalter und wiederum gefolgt von Erklärungstexten der beiden an der Inferenz beteiligten Fakten.

Auf die ausbaufähige Funktionalität der Zustandsbewertung der Gesamtsteuerung, der Online-Sensorik und der Applikationstechnik wurde im „*AG_ENGINEERING“*-Wissensmodul bereits hingewiesen. Daher beschränkt sich im „*SUM-UP“*-Modul die Funktion auf das Berichtswesen (*reporting*) aller vorliegenden Meldungen bzw. Aussagen zur Zustandsbewertung. Im vorliegenden Fall umfasst dies nur den Fakt *AG_ENGINEERING-SUM_UP::Quality_statement*, der der MMI zur Anzeige übergeben wird und seitens der Dokumentationsfunktion genutzt werden kann. Zusätzlich ist dieser Fakt aus der Wissensbasis zu entfernen.

Die Übersetzung der vorgestellten „*SUM-UP“*-Wissenselemente und -prozesse in entsprechende (sieben) Regeln ist ein geradliniger Prozess, da das Vorgehen im Grunde stets auf binäre Entscheidungen und in einem Fall auf das Prüfen, ob ein spezieller Fakt existiert,

hinausläuft. Der Umgang mit Erklärungstexten und der regelbasierten Technik zu deren Aneinanderreihung oder zur Aufbereitung für die Darstellung am MMI wurde bei anderen Wissensmodulen bereits schematisch erläutert.

Auch für das „*SUM-UP*“ -Modul wurde eine *Mind-Map* erzeugt, die in die Erklärungs- und Dokumentationskomponente der Simulation eingebunden wurde.

5.2.2.4.7 Modul *„EMERGENCY_STOP“*

Im *Problem Solving Level* wurde bereits beschrieben, dass die Notaus-Funktion (*„EMERGENCY_STOP*“) stets über den gesamten Inferenz-Prozess aller bisher vorgestellten Module präsent sein und die Prozessführung jederzeit stoppen bzw. pausieren lassen können soll. Als Resultat der Betätigung des Notaus wird dann als Applikationsrate *N2-Setpoint* 0 kg N/ha empfohlen und letztendlich an die Düngerstreuer-ECU kommandiert.

Regelbasiert lässt sich dies folgendermaßen formulieren:

REGEL EMERGENCY_STOP „Auslösen der Notausfunktion“
WENN
Für diese Regel die Funktionalität zur automatischen Fokusakquirierung aktiv ist.
Wenn von der Benutzerschnittstelle der Zustandswechsel „Notausfunktion wurde aktiviert“ gemeldet wird.
DANN
Empfehle für die Applikationsrate N2-Setpoint 0 kg N/ha.
Halte die Erklärung „Abbruch durch Notaus/Stopp“ für die Dokumentations- und Erklärungsfunktionalität fest.
Bereinige die Faktenbasis von allen vorliegenden Zwischenergebnissen.
Wechsle den Fokus zum Startpunkt eines neuen Situationsbewertungszyklus

Bei der Funktionalität zur automatischen Fokusakquirierung wird davon ausgegangen, dass die Inferenzmaschine über das Leistungsmerkmal verfügt, dass sobald eine Regel aktiv wird, die mit der Autofokus-Eigenschaft deklariert wurde, das Modul, dem sie zugehört, automatisch den Ausführungsfokus erhält. In der Regel wird dies dadurch erreicht, dass die Module direkt in einem Fokus-*Stack* angeordnet sind, der über den Ablauf der Verarbeitung der einzelnen Module entscheidet. Tritt der Autofokus-Fall in Aktion, so nimmt das zugehörige Modul die Position auf dem Fokus-*Stack* ein, die aktuell zur Ausführung gebracht wird [Fri03].

In gleicher Form ist eine weitere Regel anzufügen, die auf die Zustandsänderung reagiert, dass die Notausaktivierung wieder desaktiviert wurde. Diese Regel folgt obigem Beispiel, nur dass kein Sollwert mehr kommandiert wird.

5.2.2.4.8 Module *„MAIN"*, *„CLEAN_UP"*, *„CLEAN_UP_DEMS"*

In der beschriebenen Regel zum Notaus wurden bereits im Aktionsteil der Regel die Aktionen „Bereinige die Faktenbasis von allen vorliegenden Zwischenergebnissen" und „Wechsle den Fokus zum Startpunkt eines neuen Situationsbewertungszyklus" aufgeführt.
Unter Einbeziehung der Erfahrungen während der Implementierung lässt sich festhalten, dass die „Bereinigungsaktion" und der Startpunkt eines neuen Bewertungszyklus besser in jeweils ein eigenes Wissensmodul ausgelagert werden und im Aktionsteil der „Notaus"-Regeln eine Anweisung zum Fokuswechsel auf die entsprechenden Wissensmodule gegeben wird. Da der Inhalt dieser Module nicht im eigentlichen Sinne mit der teilflächenspezifischen Situationsbewertung zu tun hat, sondern eher mit der technischen Umsetzung der Prozessführung, d.h. der technischen „Infrastruktur" dient, werden die Wissensmodule „*MAIN"*, „*CLEAN_UP"* und „*CLEAN_UP_DEMS"* nicht weiter ausgeführt. Zusätzliche Informationen zu diesen Modulen finden sich im Anhang (A.4.1).

5.2.2.5 Tool Level

In den vorangegangenen vier Kapiteln zur Konzeption und Formalisierung wurden Problemlösungsstrategien, die benötigten Formen zur Wissensrepräsentation sowie die entsprechenden Wissenselemente dargelegt. Daraus ließen sich die Anforderungen an ein geeignetes (Software)-Werkzeug (*Tool*) zur Implementierung des Expertensystems, dem Kernstück der Simulation, ableiten. Weiterhin galt es die Rahmenbedingungen des IKB-Dürnast-Teilprojektes hinsichtlich des Zeitrahmens, der personellen Ressourcen und den finanziellen Möglichkeiten zu berücksichtigen. Gerade die zeitlichen und personellen Voraussetzungen ließen eine Implementierung, die sich nicht vorrangig mit dem eigentlichen Wissen und der Umsetzung der Problemlösungsstrategien beschäftigt, sondern auch einen großen Teil der Aufmerksamkeit der Implementierung von „Infrastrukturmechanismen" schenkt, als nicht sehr sinnvoll erachten. Die grundsätzliche Philosophie bei der Auswahl des geeigneten Tools war daher, dass die Anwendung von konventionellen Programmiersprachen, von KI-Sprachen aber auch von Wissensverarbeitungssprachen nur dann in Betracht gezogen werden sollte, falls sich keine Expertensystem-Shell finden ließe, die den durch die Konzeption und Formalisierung erarbeiteten Anforderungen gerecht wird. Eine derartige Expertensystem-Shell sollte folglich über die Möglichkeit zur regelbasierten Repräsentation des prozeduralen Wissensanteils, über die Abarbeitungs- und Konfliktlösungsstrategien Vorwärtsverkettung *(forward chaining*) und Tiefensuche (*depth-first searching*), die Option zur Modularisierung der Wissensbasis sowie über die Funktionalität einer objekt-orientierten

Repräsentation von deklarativen Wissenselementen verfügen. Üblicherweise finden sich diese Leistungsmerkmale am ehesten in der Kategorie der hybriden Expertensystem-Shells umgesetzt. Neben weiteren Kriterien wie dem Umfang der Programmierumgebung, den integrierten Software-Schnittstellen und der Wartbarkeit spielte für den Sensor-Ansatz mit Kartenüberlagerung die Fähigkeit zum Echtzeitbetrieb eine äußerst wichtige Rolle, da gemäß dem *Specification Level* eine Echtzeitsteuerung vorliegt.

Ausgehend von all diesen Anforderungen war der Wunschkandidat die äußert mächtige Expertensystem-Shell G2® der Gensym Corporation, Austin, TX, USA. Entscheidend für diese Präferenz ist das Leistungsmerkmal einer echtzeitfähigen Inferenzmaschine. Eine Kurzbeschreibung der Produktmerkmale, der Leistungsfähigkeit sowie von Implementierungsbeispielen von G2® ist in [Sri97] zusammengestellt. Eine Angebotsanfrage bei Gensym Corporation für eine G2®-Lizenz zeigte, dass dieser Wunschkandidat selbst mit einem Nachlass auf rein akademische Nutzung über das vorhandene Projektbudget nicht finanzierbar war. So fiel die Wahl auf die Expertensystem-Shell JESS (Java Expert System Shell), die von FRIEDMAN-HILL (2003) [Fri03] in Diensten der Sandia National Laboratories, Albuquerque, NM, USA entwickelt wurde. JESS ist für akademischen und nicht-kommerziellen Einsatz kostenfrei. Laut FRIEDMAN-HILL (2003) [Fri03] ist JESS eine regelbasierte hybride Expertensystem-Shell, die in der Programmiersprache Java™ (Sun Microsystems Inc., Santa Clara, CA, USA) implementiert wurde und inspiriert von der KI-Sprache CLIPS Java-Objekte generieren, manipulieren und damit schlussfolgern kann. Die Option zur objekt-orientierten Repräsentation von deklarativen Wissenselementen ist somit direkt gegeben. JESS bietet sowohl die Möglichkeiten zur Vorwärts- als auch Rückwärtsverkettung sowie zur Tiefen- und Breitensuche. Aufgrund der Implementierung in Java sind vielfältige Schnittstellenmöglichkeiten gegeben, über die Mensch-Maschine-Schnittstellen und Datenbanksysteme angebunden bzw. implementiert werden können. Für eine ausführliche Beschreibung von JESS mit seinem Eigenschaften, Sprachumfang und Schnittstellen wird auf das Fachbuch „JESS in Action“ [Fri03] und auf die Dokumentation, erhältlich auf der offiziellen Homepage „http://www.jessrules.com“[44], verwiesen.

Im Gegensatz zu G2® verfügt JESS über keine echtzeitfähige Inferenzmaschine. Jedoch wird mit der Implementierung des RETE-Algorithmus eine sehr effiziente Regelverarbeitung erreicht. Dieser Algorithmus (*rete*, lateinisch für Netz bzw. Netzwerk) wurde von FORGY (1982) [For82] entwickelt, um für Produktionssysteme eine äußerst schnelle Abarbeitung des

[44] Im Rahmen dieser Arbeit wurde letztmals am 24. August 2010 auf die Homepage zugegriffen.

match-recognize-act-cycle zu erreichen und auch umfangreiche Regelsätze noch leistungsfähig prozessieren zu können. Die effiziente Regelverarbeitung erreicht der RETE-Algorithmus dadurch, dass das Regelwerk in eine Netzwerkstruktur umgesetzt wird, die das *Pattern Matching* -Verfahren vereinfacht und gegenüber der einfachen Implementierung des *match-recognize-act-cycle* beschleunigt. Vereinfacht dargestellt, repräsentiert jeder Knoten des Netzes ein Muster (*pattern*), das im Bedingungsteil einer Regel auftritt. Wird ein Teilast von der Wurzel bis zu einem Blatt durchlaufen, so ergeben alle Knoten mit den zugehörigen Mustern den kompletten Bedingungsteil einer Regel. In dieser Baumstruktur teilen sich folglich Regeln gleiche Knoten, falls sie gleiche Muster in ihren Bedingungsteilen aufweisen. Weiterhin besitzt jeder Knoten eine Speicher/(Gedächtnis)-Funktion, die alle Fakten speichert, die mit dem (Knoten-)Muster übereinstimmen. Wenn neue Fakten in den Inferenz-Prozess eingespeist (`assert`)[45], modifiziert (`modify`) oder entfernt (`retract`) werden, durchlaufen sie alle Knoten und werden auf Übereinstimmung (*matching*) geprüft. Sind somit basierend auf den Fakten in einem Teilast für alle Knoten Übereinstimmungen gefunden worden, so ist die Regel (Blatt) bereit, gefeuert zu werden. Durch die Nutzung dieser Netzwerkstruktur hängt folglich die Leistungsfähigkeit der Inferenzmaschine nicht mehr rein von der Anzahl der Regeln und Fakten ab, sondern mehr von der Anzahl der Teilübereinstimmungen (*partial matches*), die im Netz repräsentiert werden.

JESS wurde in der Version 6.14 als Expertensystem-Shell genutzt. Die Mensch-Maschine Schnittstelle beschränkt sich auf die Kommandozeilenein- und -ausgabe des Betriebssystems der Entwicklungsplattform. Da JESS auf dem plattformunabhängigen Java™ basiert, kam als Laufzeitumgebung die Java™ 2 SE Virtual Machine für MS Windows Betriebssysteme zum Einsatz. Für die Implementierung des Expertensystems fand für JESS bis auf einen Texteditor kein weiteres Werkzeug Anwendung. Für die Entwicklung der Java™-basierten Erweiterungen bzw. Zusatzfunktionen der Simulation bieten sich mächtige Entwicklungssysteme wie Eclipse [GBM03] an. Jedoch wurde diese Leistungsfähigkeit nicht benötigt, sondern mit JOE (Java Oriented Editing, Version 2.3.25) auf einen Freeware-Editor zurückgegriffen. Dieser Editor verfügt über eine farbliche Syntax-Hervorhebung für Java™ -Sprachelemente und greift direkt auf Funktionen des JDK (*Java Development Kit*) (Sun Microsystems Inc.) zu, so dass Quellcode direkt aus dem Editor heraus übersetzt und gestartet werden kann. JOE wurde ebenfalls als Editor für die JESS-Implementierungsarbeiten genutzt. Weiterführende Information zur Einrichtung einer Java™-Entwicklungsumgebung findet sich

[45] JESS-Code bzw. Ausschnitte davon werden in dieser Arbeit durch einen Courier New-Zeichensatz gekennzeichnet.

z.B. bei [Ull03] und hinsichtlich JESS sei auf das Kapitel 3 „*Introducing Jess*“ bei [Fri03] verwiesen.

5.2.3 Implementierung

In diesem Kapitel wird das Ergebnis des Aufgabenkomplexes der Simulations-Implementierung mit den beiden Säulen, der *JDL Level 2 Processing*-MSDF und der intuitiven Interaktionsmöglichkeit beschrieben. Gemäß dem vorgeschlagenen Entwicklungsprozess baut diese Implementierung auf in den vorhergehenden Kapiteln erarbeiten Ergebnissen der Ebenen *Specification Level, Task Level, Problem Solving Level, Knowledge-base Level* und *Tool Level* auf. Dieses Kapitel gibt die Implementierung der Simulation des Sensor-Ansatzes mit Kartenüberlagerung aus dem Blickwinkel der Bedienperson wieder und listet aus Platzgründen (mehr als 100 Seiten) keinen Programm-Code auf. Eine Dokumentation der grundlegenden Systemarchitektur und Schemata zur Funktionsumsetzung der Kernelemente Ablaufsteuerung, Faktenbasis, Simulation der Prozessumgebung, Regelwerk, Datenbank-Anbindung und Mensch-Maschine-Schnittstelle ist im Anhang, Kap. A.4 ausgeführt.

Anhand der grafischen Bedienoberfläche werden im Folgenden die typischen Anwendungsfälle (*use cases*) der Simulation beschrieben. Abbildung 5.7 stellt die realisierte interaktive Bedienoberfläche der Simulation dar.

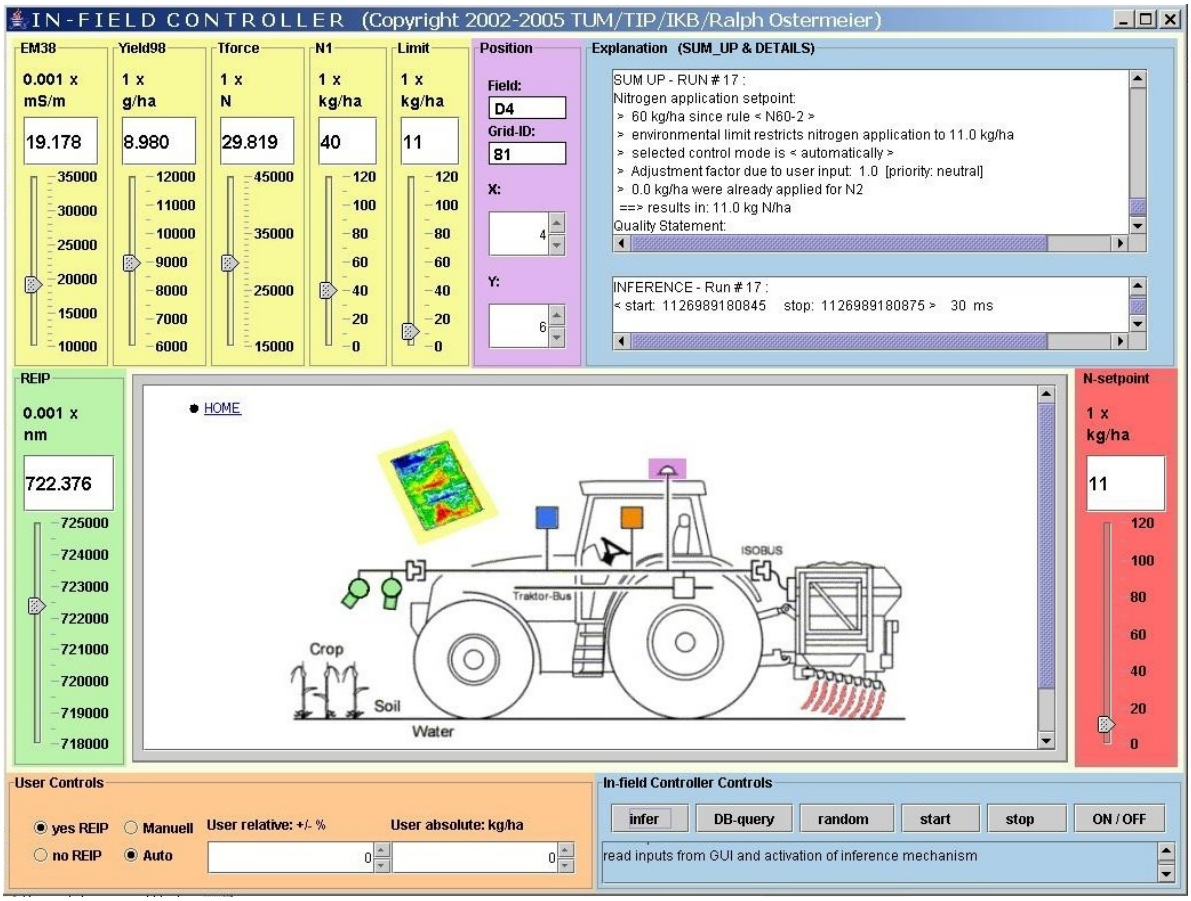


Abbildung 5.7: GUI der Simulation

Im zentralen Fenster *(„HTML Panel"*) wird die in Kapitel 4.4.1 eingeführte Darstellung einer Traktor-Düngerstreuer-Kombination mit „*In-field Controller*“, Online-Sensoren, angedeuteten Überlagerungskarten und einer Benutzerschnittstelle angezeigt. Diese Kernkomponenten sind jeweils farblich unterschieden und weisen auf eine Zuordnung mit den das zentrale Fenster umgebenden Ein- und Ausgabeelementen hin.

So werden in der linken oberen Ecke die einzelnen Überlagerungskartenwerte *EM38*, *Yield98*, *Tforce*, *N1*, *Limit* der aktuell vorliegenden Teilfläche mit gelb hinterlegten Ein- und Ausgabeelementen umgesetzt. Dabei zeigt ein linearer, vertikaler Schieberegler („*Sliders for Maps"*) kombiniert mit einem numerischen Feld den Kartenwert sowohl grafisch als auch als Text an. Gleichzeitig kann der Wert jedoch auch intuitiv über den Schieberegler oder durch Tastatureingabe über das Textfeld eingestellt bzw. eingegeben werden.

In gleicher Art und Weise wird auf der mittigen linken Seite mit der Farbe Grün codiert der Online-Sensorwert *REIP_2* umgesetzt („*Slider Online Sensor REIP*“).

Auf der mittigen rechten Position wird das Endergebnis, die Sollwertempfehlung für die zweite Stickstoffgabe mit einem roten „*Slider Application N-setpoint*“-Schieberegler-Element dargestellt. Jedoch wird hierbei die Funktionalität auf die reine Anzeige

beschränkt.

In der Mitte oben wird in dem violett unterlegtem Fensterelement („*Position Panel*") die Position mit der aktuell vorliegenden Teilfläche angegeben und auswählbar gemacht. Da ein rasterbasiertes Schema zugrunde gelegt wurde umfasst dies ein Textfeld mit dem ausgewählten Feld und einem Textfeld für den Bezeichner (*identifier*) des Rasterelements („*Grid-ID"*), welches sich aus den eingestellten x- und y-Werten ergibt. Die Anzeigeelemente („*Spinners"*) für x und y verfügen auch über Eingabefunktionalität an der rechten Seite, über die die Werte in Einser-Schritten in- oder dekrementiert werden können.

In der unteren linken Ecke der Bedienoberfläche werden die Eingabemöglichkeiten der Bedienperson („*User Controls Panel"*) in hellbrauner Farbgebung abgebildet. Dies umfasst die Auswahlmöglichkeiten zwischen der Verfügbarkeit des Online-Sensors für die *REIP_2*-Bestimmung („*yesREIP"*) oder dessen entsprechendem Fehlen („*noREIP"*) und der Wahl zwischen manuellen („*Manuell"*) oder automatischen Modus („*Auto"*). Weiterhin wird über zwei „*Spinner*"-Eingabeelemente die Möglichkeit geschaffen, einen relativen Übersteuerungswunsch („*User relative"*) zu tätigen oder einen Absolut-Wert („*User absolute*") für die Sollwertkommandierung anzugeben.

In der rechten unteren Ecke der Bedienoberfläche werden die Eingabeoptionen bzw. Steuerelemente der „*In-field Controller"*-Simulation („*In-field Controller Controls Panel"*) angeordnet. Über sechs verschiedene Tasten bzw. Schaltflächen kann die Simulation bedient und gesteuert werden. Die einzelnen Möglichkeiten werden nachfolgend anhand der typischen *use cases* beschrieben. Zusätzlich zu den Tasten wird in einem unterhalb angeordneten rollbaren (*scroll*-baren) Textausgabefenster in knapper Form die auswählte Aktion beschrieben.

Vervollständigt wird die Bedienoberfläche durch das blau gefärbte Fenster („*Explanation (SUM_UP & DETAILS) Panel"*) in der rechten oberen Ecke. In dem größeren der beiden *scroll*-baren Textausgabefelder wird das Ergebnis des jeweiligen Entscheidungsprozesses zur Bestimmung der Sollwertempfehlung kurz zusammengefasst und somit die Erklärungsfunktion des Expertensystems teilweise abbildet. In dem kleineren Textausgabefenster wird ein Ausschnitt aus der Diagnose bzw. Testfunktionalität mit der Bestimmung der Zeitdauer für den gerade durchgeführten Schlussfolgerungsprozess dargestellt.

Zusätzlich zu dem „*Explanation (SUM_UP & DETAILS) Panel"* wird auch in dem zentralem „*HTML Panel"* dem Benutzer Material für die Erklärungsfunktion bereit gestellt. Dieses Fenster verfügt über die Fähigkeit in HTML codierte Inhalte zu interpretieren und anzuzeigen. So sind neben der in Abbildung 5.7 gezeigten Abbildung des gesamten Prozessführungssystems, Abbildungen des Feldes mit den nummerierten Rasterelementen und dem x, y-Koordinatensystem, das Zustandsdiagramm der Ablaufsteuerung und die *Mind-Maps* der einzelnen Regelmodule hinterlegt.

Nach der Beschreibung der Elemente der GUI wird zur Erläuterung der möglichen

Anwendungsoptionen der Simulation gewechselt.

Die Standardfunktion, aktivierbar durch Betätigung des „infer"-Tasters des „In-field Controller Controls Panel", liest zuerst die Prozesseingangsinformationen, d.h. die über die „Slider" eingestellten Messwerte der Online-Sensorik und der Überlagerungskarten wie auch die Eingaben im „User Controls Panel" ein. Dann wird der Inferenz-Prozess, also das „JDL Level 2 Processing - Situation Assessment" angestoßen und durchgeführt. Abschließend wird der empfohlene Stickstoff-Applikationswert im zugehörigen GUI-Ausschnitt auf der Schieberegler-Skala und dem numerischen Ausgabefeld angezeigt. Weiterhin werden die zusammengefassten Ergebnisse im „Explanation (SUM_UP & DETAILS) Panel" angezeigt bzw. aktualisiert. Somit setzt diese Funktion die klassische Interview-Situation beim Einsatz eines Expertensystems um, nur dass der Nutzer seine Eingaben intuitiv über die grafischen Eingabeelemente tätigen kann und alle nötigen Eingangsinformationen in einem Schritt erfasst werden.

Die zweite Grundfunktion, auslösbar durch den „DB-Query"-Taster, ersetzt das Einlesen der Online-Sensor-Messwerte und der Überlagerungs-Kartendaten von der Bedienoberfläche durch Einlesen aus einer Datenbank mit im Feld erfassten Messwerten. Dabei werden diese Daten nur für ein Rasterelement eingelesen. Dieses Rasterelement wird vom Nutzer mithilfe des „Position Panel" über die x, y-Einstellfelder ausgewählt. Der zugehörige Rasterelement-identifier wird wie die zugehörigen Teilflächendaten aus der Datenbank gelesen. Mit den aus der Datenbank gelesenen Daten werden die entsprechenden Anzeigeelemente der Bedienoberfläche aktualisiert. Um den für den Schlussfolgerungsprozess nötigen Satz an Eingangsinformationen zu vervollständigen, werden die noch fehlenden Nutzereingaben aus dem „User Controls Panel" eingelesen. Der anschließende Inferenz-Prozess und die relevanten Ergebnisanzeigen erfolgen wie bei der bereits beschriebenen „infer"-Funktion. Zusätzlich werden jedoch die Ergebnisse und für Diagnosezwecke gemessene Zeitdauern für die Teilprozesse Inferenz und Datenbankzugriff in die Datenbank geschrieben.

Die dritte Grundfunktion, zu aktivieren durch den „random"-Taster, erzeugt eine Wertebelegung der Online-Sensor-Messwerte und der Überlagerungs-Kartendaten mittels eines Zufallsgenerators und aktualisiert die Anzeigeelemente entsprechend. Die restlichen Nutzereingaben des „User Controls Panel" bleiben der Bedienperson vorbehalten. Um den eigentlichen Data Fusion- bzw. Schlussfolgerungsprozess auszulösen, kann in einem zweiten Schritt der „infer"-Taster betätigt werden.

Mit den drei beschriebenen Anwendungsfällen kann der Nutzer die „*In-field Controller*"-Simulation in einer Art Einzelschrittverfahren bzw. statischen Modus bedienen und verschiedenste Eingangsgrößenszenarien durchspielen. Mit den drei restlichen Tasten hingegen lässt sich ein Anwendungsfall für ein dynamisches, zeitgesteuertes Verhalten simulieren.

Die Betätigung des „*start"*-Tasters startet diesen Anwendungsfall. Dabei wird zyklisch mit einer standardmäßigen Updaterate von 2 Hz folgende Prozedur durchgeführt. Zuerst werden zufällige Werte für Online-Sensor-Messwerte und die Überlagerungs-Kartendaten generiert, dann die Anzeigen dementsprechend aktualisiert und die Nutzereingaben des „User Controls Panel" abgefragt. Mit diesen Prozesseingangsinformationen wird dann der Schlussfolgerungsprozess durchgeführt und die resultierende Sollwertempfehlung angezeigt sowie die Erklärungstexte ausgegeben.

Mit der „stop"-Taste lässt sich dieser zyklische Vorgang wieder stoppen. Ein laufender Vorgang wird dabei jedoch nicht unterbrochen, sondern stets zu Ende geführt.

Um jedoch auch die Notausfunktionalität auf „In-field Controller“-Ebene zu simulieren, wurde auch ein „ON/OFF"-Taster mit Wechselschalterfunktion bereitgestellt. Die Aktivierung löst den Notaus aus und unterbricht sofort den aktuellen Inferenz-Vorgang. Dies führt, wie im Knowledge-base Level definiert, zur Empfehlung der Nullapplikation mit entsprechender Anzeige und Erklärungstext. Die zufällige Generierung der Eingangsinformationen und die zugehörige Anzeigenaktualisierung läuft aber weiter, genauso wie wenn auf einem (heterogenen) Feld weitergefahren wird. Der Schlussfolgerungsvorgang bleibt jedoch sozusagen „kurzgeschlossen“ bis der Notaus durch erneutes Betätigen des „ON/OFF“-Tasters wieder gelöst wird.

Neben dieser nach wenigen Durchläufen intuitiv bedienbaren GUI wurde auch eine zusätzliche Ein- und Ausgabefunktionalität auf Basis der JESS-Kommandozeilenumgebung (Abb. 5.8) geschaffen.

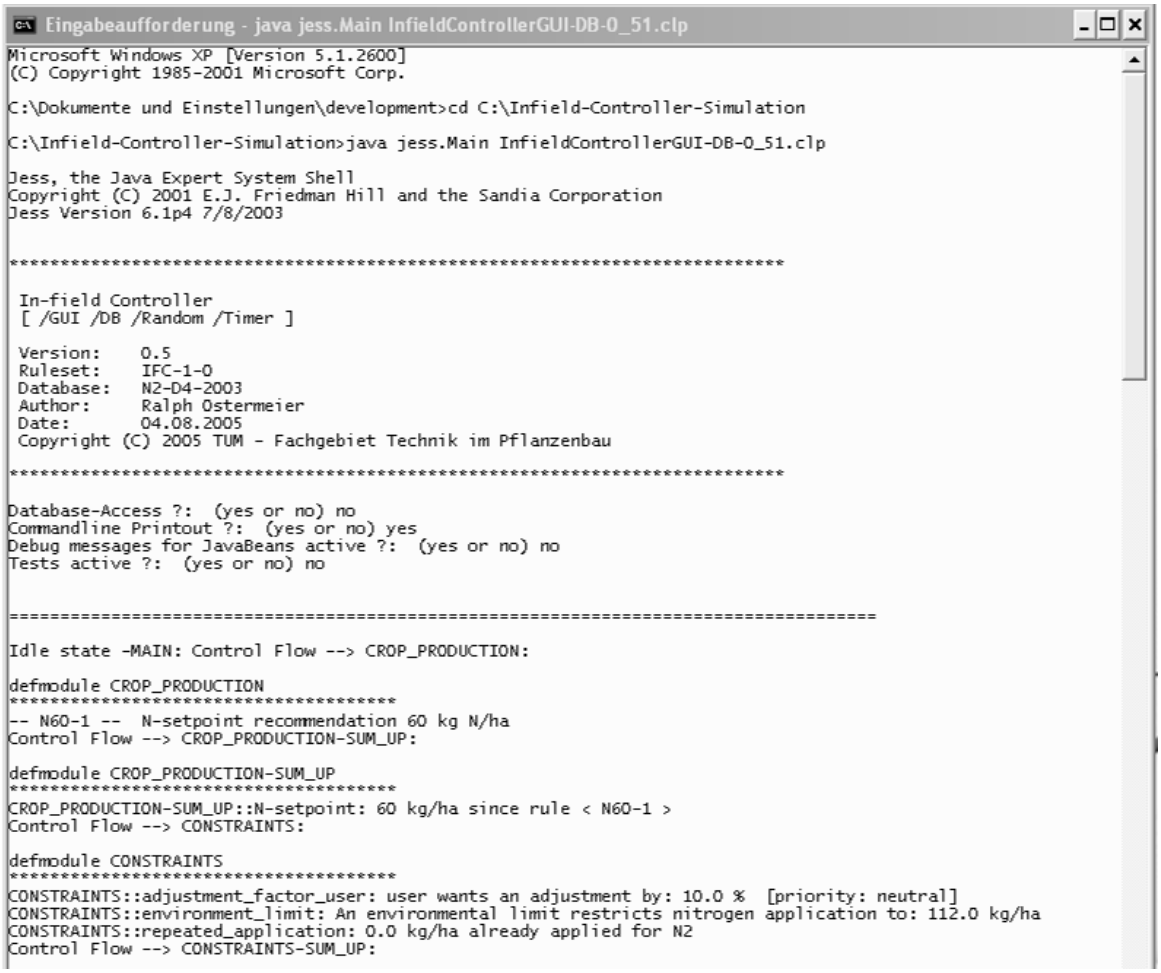


Abbildung 5.8: MMI der Simulation auf Basis der JESS-Kommandozeile

Nach Programmstart wird der Benutzer, normalerweise betrifft dies nur den Experten oder Wissensingenieur, auf der JESS-Kommandozeilen-Benutzerschnittstelle in sequentieller Abfolge nach der An- oder Abwahl folgender Konfigurationsmöglichkeiten gefragt:

- Verbindungsherstellung mit der Datenbank mit den Datensätzen der Feldversuche sowie einer Datenbank zur Speicherung von Ergebnissen der Test- und Validationsphase („*Debug-Ergebnisse"*),
- Ausgabe von „*Debug"*-Nachrichten der Schlussfolgerungsprozesse auf der JESS-Kommandozeile,
- Ausgabe von „*Debug"*-Nachrichten zur Simulation der Prozessumgebung auf der JESS-Kommandozeile,
- Aktivierung und Integration von Test-Funktionalität mittels gezieltem Einbringen von Fakten in den Schlussfolgerungsprozess,
- Aktivierung der Speicherfunktion der ausgewählten „*Debug-Ergebnisse"* in der diesbezüglichen Datenbank (Default-Einstellung: inaktiv, da in Abhängigkeit von der Option der Verbindungsherstellung mit der Datenbank).

Weiterhin stehen dem Wissensingenieur die Möglichkeiten der Expertensystem-Shell JESS mit den integrierten Diagnosemerkmalen zur Verfügung. Ebenso kann er von dem

Grundprinzip eines regelbasierten Expertensystems Gebrauch machen, zur Laufzeit neue Regeln definieren, Fakten modifizieren, neu einbringen oder entfernen, d.h. das *working memory* manipulieren, sowie auch definierte Funktionen im Stile einer Skriptsprache aktivieren. Dies ließ sich bei der evolutionären Entwicklung und dem Testen sehr hilfreich einsetzen, um Testfakten zur Laufzeit zu erzeugen, Fahrspursimulationen, die mit Daten aus der Datenbank hinterlegt werden, „*on the fly*“ zu generieren, Regeländerungen und -erweiterungen zu testen oder aber auch nur um GUI-Elemente zu modifizieren und neu zu organisieren. Bei der durchgeführten Implementierung ist jedoch eine Einschränkung zu beachten. Die Eingabemöglichkeit der Kommandozeile ist nur zugänglich solange die GUI noch nicht initialisiert ist. Die Ausgabemöglichkeiten der Kommandozeile sind davon nicht betroffen und grundsätzlich ließe sich die Eingabe über die GUI ermöglichen und an die Kommandozeilenebene übergeben.

Als hilfreiche Diagnosefunktionen erwiesen sich die Möglichkeiten den *match-recognize-act-cycle* verfolgen zu können. Hierzu dienen vor allem die `(watch)`-Funktionen, die eine Debug-Ausgabe erzeugen, falls ein spezifisches Ereignis in JESS stattfindet. Dies ist beispielsweise die Anzeige einer Regel, die gerade aktiviert wurde bzw. bereit zu feuern ist, bei der Wahl von `(watch activations)`, die Anzeige der aktuell vorliegenden Fakten mittels `(watch facts)` oder eines Modulfokuswechsels über `(watch focus)`. Das `(view)`-Kommando erlaubt einen „Schnappschuss“ von dem aktuellen RETE-Netzwerk in grafischer Baumstruktur-Darstellung, bei dem die einzelnen Knoten für weitere Informationen mit der „Computer-Maus“ angewählt werden können. Neben den Diagnosefunktionen besteht auch die Möglichkeit zur Konfiguration des Schlussfolgerungsprozesses, z.B. mit der Wahl der Suchmethode zur Konfliktauflösung mit `(set-strategy depth)` für eine Tiefensuche oder mit `(set-strategy breadth)` für eine Breitensuche. Auch die Wahl zwischen Vorwärts- und Rückwärtsverkettung lässt sich kommandieren.

5.2.4 Testen

Im Kapitel zum Stand der Technik wurde für das Thema Testen die Unterscheidung in Verifikation und Validation aufgezeigt und hinsichtlich der Expertensystementwicklung auf die Problematik der Verifikation hingewiesen. Daher lag der Schwerpunkt bei der Verifikation hauptsächlich im Test der korrekten Funktion und Implementierung der eindeutig spezifizierten Systembestandteile. Dies umfasste vor allem die Ablaufsteuerung, den Mechanismus zur Umsetzung der Entscheidungsbäume in Regeln, die Simulation der Prozessumgebung mit der zugehörigen Realisierung der Faktenbasis, die

Datenbankanbindung und die Mensch-Maschine-Schnittstelle. Für die vorliegende Arbeit hat der Autor sowohl die Entwicklungs- als auch Verifikationsarbeit übernommen. Dabei wurden die spezifizierten Eigenschaften der Funktionen und Softwarekomponenten über funktionsorientierte Tests der einzelnen Schnittstellen überprüft. Zur prinzipiellen Überprüfung der Funktionsfähigkeit von Regelwerk-Konstrukten wurde mit gezieltem Einbringen oder Entfernen von Testfakten gearbeitet. Im Mittelpunkt des Testens stand für diese Arbeit jedoch die Validation des Expertensystems- bzw. *Data Fusion*-Anteiles mit den Schwerpunkten Problemlösungsfähigkeit und informationstechnische Eignung und Leistung gemäß [Sri97]. Der dritte Aspekt, die psychologische Eignung, ausgedrückt in einfacher Bedienbarkeit, guter Erlernbarkeit und Verständlichkeit (der Erklärungskomponente), wurde nur nachrangig behandelt, da der Einsatz nur im Rahmen des Projektes bei Fachexperten zumeist im Beisein des Wissensingenieur geplant war und durchgeführt wurde. Dabei zeigte sich, dass die Bedienung über die GUI den Nutzern nach kurzer Einweisung sehr einfach gefallen ist. Für den mit der Thematik vertrauten Nutzer war auch das Nachvollziehen des Schlussfolgerungsprozesses über die im *„Explanation (SUM_UP & DETAILS) Panel"* aufgelisteten Erklärungen oder der Ausgabe der „Debug"-Nachrichten auf der Konsolen-Schnittstelle ausreichend. Für weitergehende Informationen über das implementierte Regelwerk nutzten sie vor allem die im *„Html Panel"* abfragbaren *Mind-Maps*. Auf dem Abschluss-Symposium der IKB-Dürnast DFG-Forschergruppe am 11. und 12. Oktober 2005 am Wissenschaftszentrum Weihenstephan der TU München wurde die Erfahrung gemacht, dass den Interessenten der Zugang zur Simulation intuitiv möglich war und nur wenig einleitender Erklärung bedurfte.

5.2.4.1 Problemlösungsfähigkeit

Zur Überprüfung der Problemlösungsfähigkeit wurde sowohl auf manuelle Tests, als auch auf teilweise automatisierte Verfahren zurückgegriffen. Bereits im Verlauf der Wissensakquisition wurde mit den entsprechenden IKB-Dürnast-Projektpartnern und Kollegen am Fachgebiet das erarbeitete Wissen diskutiert und somit auf Ebene der Formalisierung validiert. Nach erfolgten Implementierungsschritten wurde dies wiederholt und anhand der Simulation für unterschiedliche Testfälle manuell durchgespielt und die gefundene Lösung sowie der Einfluss von Eingangsparameteränderungen diskutiert. Jedoch ist für das *„CROP_PRODUCTION"*-Wissensmodul die implementierte Lösung direkt abhängig von der Problemlösungsfähigkeit, die über das von WEIGERT (2006) [Wei06] angewandte Verfahren zur automatisierten Generierung von Entscheidungsregeln bereit

gestellt wird. Für die Validation der Genauigkeit, Präzision, Breite und Tiefe des Lösungsspektrums wird daher auf die Erörterungen dieser Arbeit referenziert. Auf dem IKB-Dürnast-Abschluss-Symposium wurde das System der wissenschaftlichen Gemeinschaft vorgestellt und konnte von dieser getestet werden. Hierbei stand die interaktive Eingabewerteinstellung mit nachfolgender einzelner Auslösung des Inferenz-Prozesses im Fokus der Interessenten, da in dieser Betriebsvariante die Entscheidungsfindung am besten nachvollzogen und erörtert werden konnte.

Neben diesen rein manuellen Aktivitäten wurde das Testen mit den beiden folgenden zumindest teilautomatisierten Verfahren unterstützt. Vom IKB-Dürnast-Teilprojekt 12 - Projektpartner wurde dafür nicht nur das Wissen für das „*CROP_PRODUCTION*"-Wissensmodul beigesteuert, sondern auch die zu erwartenden teilflächenspezifischen Sollwertempfehlungen für den ausgewählten Versuch (Versuchsfeld D4 der TU München, 2003, 2. N-Gabe, Winterweizen). Dieser Zielwert-Vektor wurde ebenfalls in der Datenbank hinterlegt. In einem automatisierten Prozess wurde damit ohne weitere Übersteuerungen durch den Nutzer das Testfeld sozusagen virtuell bearbeitet und die gefundene Lösung für den teilfächenspezifischen Sollwert mit dem Zielwert auf Übereinstimmung geprüft. Das im nachfolgenden Unterkapitel beschriebene Testen der informationstechnischen Leistung wurde zugleich auch für ein halbautomatisiertes Testverfahren zur Leistungsfähigkeit der Problemlösungsfähigkeit genutzt. Durch die dabei angewandte Generierung von Testfällen mittels der zyklischen Ablaufsteuerung mit zufällig generierten Eingangsdaten, jedoch noch mit manuellen Nutzereingaben über die GUI, wurde eine breite Abdeckung der Eingangsparameter und der gefundenen Lösungen erzielt, die im Diagnose-Teil der Datenbank abgelegt wurden. Stichprobenartig wurden Testfälle ausgewählt und der Lösungsweg und die gefundene Sollwerteempfehlung anhand der dokumentierten Werte nachvollzogen. Zugleich wurde überprüft, ob die zusammengefassten Erklärungstexte den angewandten Lösungsweg korrekt wiedergaben. Weiterhin wurde für die Überprüfung auf Wiederholbarkeit eine Sammlung von Testfällen mit einer Bandbreite von 0 - 90 kg N/ha für den Applikationssollwert mit entsprechenden Eingangswertkombinationen hinterlegt und in zufälliger Reihenfolge in den zyklischen Ablauf anstatt der mit dem Zufallsgenerator erzeugten Eingangswerte eingespeist. Die wiederholbare und zuverlässige Ergebnisfindung wurde wiederum nach dem Testdurchlauf auf Basis der in der Datenbank dokumentierten Daten überprüft. Es zeigte sich, dass für jeden Schlussfolgerungsvorgang stets eine eindeutige Lösung vom System präsentiert wurde.

5.2.4.2 Informationstechnische Eignung und Leistung

Aufgrund der generellen Zielrichtung einer Real-time-Prozessführung konzentrierte sich die Untersuchung der informationstechnischen Eignung und Leistungsfähigkeit auf die Bestimmung der Laufzeit des Schlussfolgerungsprozesses und des Datenbankzugriffes. Zusätzlich wurde die dabei vorliegende CPU- und Hauptspeicherauslastung einer näheren Untersuchung unterzogen. Ein robustes Laufverhalten und die Portierbarkeit wurden beiläufig betrachtet, soweit sich dies aus der Durchführung der vorherig beschriebenen Kriterien und dem Testen der Problemlösungsfähigkeit ergab.

Tabelle 5.2 gibt einen Überblick über die vier unterschiedlichen Testplattformen, auf denen die Simulation installiert und getestet wurde. Dabei handelt es sich durchwegs um Notebooks, die zur Laufzeit des IKB-Teilprojektes (2002-2005) beschafft wurden oder im privaten Besitz des Autors waren. Die technischen Angaben entstammen der Anzeige des (Diagnose-)Programms *„Systemsteuerung/Systemeigenschaften/Allgemein“* des jeweiligen Betriebssystems.

Tabelle 5.2: Testplattformen

Index	CPU	CPU-Taktrate	Hauptspeicher	Betriebssystem	Notebook
I	Intel® Pentium® III-M	1,06 GHz	640 MB	Microsoft Windows 2000 Professional	Kontron ReVolution
II	Mobile Intel® Celeron®	0,79 GHz	256 MB	Microsoft Windows XP Home Edition	Sony PCG-R600MX
III	Mobile Intel® Celeron®	1,50 GHz	256 MB	Microsoft Windows XP Home Edition	Toshiba Satellite 1110
IV	Intel® Pentium® M	1,86 GHz	1 GB	Microsoft Windows XP Home Edition	Samsung X20

Bei Testplattform I handelte es sich um eine zum damaligen Zeitpunkt dem Stand der Technik entsprechende Ausführung eines Notebooks für den rauen Außeneinsatz mit energiesparender Technik für eine lange batteriebetriebene Laufzeit. Die Testplattformen II und III waren Notebooks aus dem Bereich des preiswerten Privatanwender-Segments mit entsprechend schwacher Prozessor- und Hauptspeicherausrüstung. Testplattform IV war ein zum Beschaffungszeitraum leistungsfähig ausgestattetes Notebook für den professionellen Einsatz in Büroumgebung. Auf jeder der vier Plattformen ließ sich die Simulation problemlos aufsetzen. Weder beim manuellen noch beim automatischen Testen ergaben sich auf einer der

vier Plattformen Probleme hinsichtlich eines robusten Laufverhaltens.
Es folgen zuerst Ausführungen über die Bestimmung der CPU- und der Speicherauslastung. Anschließend wird das Kapitel mit den Ergebnissen der Untersuchung von Laufzeiten abgeschlossen.
Wie erwartet, wurden die höchsten Auslastungswerte auf der leistungsschwächsten Testplattform, der Plattform II, gemessen. Da bei eingebetteten Rechnersystemen für Agrarelektronik in der Regel leistungsschwächere Systemkonfigurationen als bei Rechnern für den Büroeinsatz zum Einsatz kommen, ist dieser sogenannte schlechteste Fall (*worst case*) der vorliegenden Testplattformen von besonderem Interesse und wird im Folgenden dokumentiert. Die Testplattform wurde derart konfiguriert, dass soweit möglich nur nötige Systemprozesse aktiv waren. Außer der Simulation und dem MySQL-Datenbankserver-Dienst wurde keine andere Applikation gestartet. Die Antivirussoftware wurde desaktiviert und es wurde überprüft, dass keine *Firewall*-Einstellung eine störende Wirkung auf die Simulation ausübte. Die Wahl des Energieschemas „Dauerbetrieb“ desaktivierte sämtliche Energiesparoptionen und auch auf eine Bildschirmschonerfunktion wurde verzichtet. Mit diesen Voreinstellungen sollte gewährleistet sein, dass der Prozess „*java.exe*“ hinsichtlich der CPU-Auslastung für die Testdauer der dominierende Prozess war. Bei der Java-Umgebung („*Java Virtual Machine*“) wurde die Default-Einstellung 64 MB für den *Heap-Size* beibehalten. Für die Simulation wurde der zyklische Ablaufsteuerungsmodus mit 2 Hz Wiederholrate und der Eingangswertgenerierung per Zufallszahlgenerator gewählt. Der Tester nahm während des Testlaufes von Zeit zu Zeit Eingaben im *„User Controls Panel“* vor, „navigierte“ in den Anzeigen des *„HTML Panel”* und verschob *(„scrollte“)* die Textausgaben des „*Explanation (SUM_UP & DETAILS) Panel”* zeilen- oder passagenweise. Der Datenbank-Serverdienst war aktiviert, die Option der Datenbankanbindung jedoch nicht ausgewählt worden. Die Bestimmung der CPU- und Speicherauslastung wurde über das Programm „*Windows Taskmanager*“ durchgeführt, das Bestandteil des Betriebssystems ist und der Anzeige und Verwaltung von Programmen und Prozessen dient. Als Aktualisierungsgeschwindigkeit wurde die Stufe „Normal“ ausgewählt.
Die somit gewonnenen Messergebnisse für die CPU- und Speicherauslastung für den Prozess „*java.exe*“, der die Simulation ausführt, sind in Tabelle 5.3 zusammengestellt. Dabei werden fünf verschiedene Nutzungskonfigurationen betrachtet, die sich vor allem in der Anzeige der Mensch-Maschine-Schnittstelle und der „Debug-Funktionalität“ unterscheiden. Mit „GUI im Vordergrund“ wird ausgedrückt, dass die GUI in voller Größe im Vordergrund auf dem Notebookdisplay sichtbar und bedienbar ist. Hingegen bedeutet „GUI im Hintergrund“, dass

die GUI ausgeblendet bzw. nach MS Windows-Terminologie minimiert wurde. „Konsole im Vordergrund" sagt aus, dass die Konsolenschnittstelle, d.h. das MMI auf Basis der JESS Kommandozeile, auf dem Notebookdisplay sichtbar ist und es weitestgehend ausfüllt. Wohingegen „Konsole im Hintergrund" für eine Ausblendung bzw. Minimierung steht. Die „Nutzung der „*HTML Panel*"-Funktionalität" umschreibt, dass der Tester die interaktiven Möglichkeiten dieser GUI-Komponente nutzt und die bereitgestellten HTML-Dateien anwählt und wechselt. „Debug-Nachrichten aktiviert" weist darauf hin, dass die Ausgabe der Debug-Nachrichten auf der Konsolenschnittstelle aktiviert wurde.

Tabelle 5.3: CPU- und Speicherauslastung durch den Prozess „java.exe" für unterschiedliche Nutzungsoptionen - Testplattform II (Sony PCG-R600MX)

Index	Konfiguration	CPU-Auslastung	Speicherauslastung (typisch)
I	GUI im Vordergrund Konsole im Hintergrund	7 % - 15 %	9,5 MB - 30 MB
II	GUI im Vordergrund Konsole im Hintergrund Nutzung der „*HTML Panel*" - Funktionalität	7 % - 30 %	9,5 MB - 30 MB
III	GUI im Vordergrund Konsole im Hintergrund Debug-Nachrichten aktiviert	9 % - 18 %	9.5 MB - 30 MB
IV	GUI im Hintergrund Konsole im Vordergrund	2 %	9.5 MB - 30 MB
V	GUI im Hintergrund Konsole im Vordergrund Debug-Nachrichten aktiviert	4 %	9.5 MB - 30 MB

Während sich die Speicherauslastung für alle fünf Nutzungsoptionen gleich verhielt, traten bei der CPU-Auslastung doch merkbare Unterschiede auf. Konfiguration IV zeigt die geringste CPU-Last und spiegelt hauptsächlich die Belastung durch die Ablaufsteuerung, den Schlussfolgerungsprozess, die Aktualisierung der Faktenbasis und die entsprechende Simulation der Prozessumgebung wider.

Die Verdoppelung der Last in Konfiguration V ist Folge der zeitaufwändigen Ausgabe der „Debug"-Textnachrichten auf der Konsolenschnittstelle. Eine deutliche Zunahme der CPU-Auslastung wird durch die Einblendung der GUI verursacht. Minimal führt dies zu einer Erhöhung um 5 %. Je nach Nutzung der Funktionalität der GUI-Komponenten schwankt die zusätzliche Last in einer Bandbreite von etwa 8 %. Ein weiterer spürbarer Lastanstieg wird

durch die interaktive Nutzung der „*HTML Panel*“-Funktionalität erzeugt, was bis zu 15 % Zusatzauslastung führt.

Diese Messergebnisse legen eine starke Abhängigkeit der CPU-Last vom Einsatz und Nutzung der GUI offen. Auch die Speicherauslastung steht in einem direkten Zusammenhang mit der Funktionalität der GUI. Hierbei zeigt sich im Gegensatz zur CPU-Auslastung für alle Konfigurationen eine gleichartige Nutzung des Arbeitsspeichers. Die untere Grenze konnte bei etwa 9,5 MB identifiziert werden. Typischerweise steigt die Auslastung Schritt für Schritt an und wird in regelmäßigen Abständen auf etwa 9,5 MB automatisch zurückgesetzt. Typischerweise erfolgte diese Umbildung der Speicherauslastung in den meisten Fällen vor dem Erreichen von ca. 30 MB. Jedoch erreicht sie auch in einigen Fällen fast die maximale Java *Heap Size*-Größe. Das jeweils kontinuierliche Anwachsen der Speicherauslastung konnte durch Beobachtung direkt mit der Nutzung der Textausgabe-Funktionen in den *scroll*-baren GUI-Elementen des „*Explanation (SUM_UP & DETAILS) Panel”* in Verbindung gebracht werden. Die zugehörigen GUI-Elemente verfügen über einen Puffer zur Speicherung der Textausgaben, der auch dann befüllt wird, wenn die GUI im Hintergrund ist. Daher führen die unterschiedlichen Konfigurationen zu keinem unterschiedlichen Verhalten hinsichtlich der Speicherauslastung.

Neben dem „*java.exe*“-Prozess wurde auch die Auslastung durch drei weitere Prozesse überprüft. Dies waren zum einen der Kommandozeileninterpreter des Betriebssystems („cmd.exe“), dessen CPU-Auslastung bei 0 % und die Speicherauslastung bei maximal 3 MB lag. Zum anderen waren dies der „*Windows Taskmanager*“ („tskmgr.exe“) mit 1 % - 2 % CPU- und maximal 3 MB Speicherauslastung, sowie der Server-Dienst für die MySQL-Datenbank („mysqld.exe“) mit 0 % CPU- und maximal 5,5 MB Speicherauslastung.

Bei den vorliegenden zyklisch ablaufenden Testverfahren war der Datenbankzugriff nicht aktiviert. Mittels des Simulationsmodus „*DB-Query*“ wurde die Anwendung sozusagen im Einzelschrittverfahren untersucht, was hinsichtlich der Speicherauslastung kein geändertes Verhalten zeigte und bei normaler Aktualisierungsgeschwindigkeit des „*Windows Taskmanager*“ keine messbare CPU-Auslastung zeigte. Bei der Testaktivität zur Bestimmung von Laufzeiten konnte der Einfluss von Schreibzugriffen auf die Datenbank beobachtet werden. Bei der gegebenen Genauigkeit der CPU-Auslastungsmessung ließ sich eine Zusatzbelastung von 0 – 1 % erkennen.

Für die bereits mehrfach angesprochene Bestimmung der Laufzeit des Schlussfolgerungsprozesses und des Datenbankzugriffes wurden Funktionen zur Laufzeitermittlung in die Simulationsimplementierung auf JESS-Ebene integriert. Dabei

wurde die grundsätzliche Vorgehensweise gewählt, dass sowohl vor Eintritt der entsprechenden Programmaktivität als auch nach deren Ende die aktuelle Systemzeit ausgelesen wird. Mittels Differenzbildung wird die gesuchte Laufzeit ermittelt. Für die Bestimmung der aktuellen Systemzeit wird von JESS aus mit `(call System currentTimeMillis)` auf eine standardmäßig von Java bereitgestellte Funktionalität zurückgegriffen. Laut der Java-API-Dokumentation liefert diese Methode die momentane Systemzeit auf Millisekunden-Basis. Jedoch wird auch darauf hingewiesen, dass die Auflösung von dem zugrundeliegenden Betriebssystem abhängt und folglich eine gröbere Auflösung aufweisen könnte. Der Zeitbedarf für diesen Funktionsaufruf und der anschließenden Differenzbildung wurde im Vergleich zu der zu messenden Größe als vernachlässigbar eingeschätzt.

Auf dieser Grundlage wurde die Laufzeit für den Schlussfolgerungsprozess `time-diff-inference` und die Laufzeit für den lesenden Datenbankzugriff `time-diff-dbQuery` ermittelt. In der dieser Arbeit zugrunde liegenden Simulationsimplementierung schließt `time-diff-inference` zusätzlich zur Laufzeit der definierten Funktion `(run-system)` auch die Textausgabe in den beiden Ausgabefeldern des *„Explanation (SUM_UP & DETAILS) Panels"* mit ein. Bei `time-diff-dbQuery` umfasst die Zeitmessung die Durchführung der Funktion `(sql-read-data)`, die für jeden Eingangswert, d.h. achtmal, nacheinander lesend auf die Datenbank zugreift und dann stets mit den eingelesenen Werten die Prozessumgebungssimulation sowie die GUI-Elemente *„Sliders for Maps"* und *„Slider Online Sensor REIP"* aktualisiert.

Wie bereits bei den Ausführungen zur CPU- und der Speicherauslastung werden nachfolgend vorrangig die Testergebnisse der leistungsschwächsten Testplattform II beschrieben, da dort die längsten Laufzeiten gemessen wurden. Die Testplattform wurde identisch zu der Konfiguration aufgesetzt, die für die Messungen der CPU- und Speicherauslastung genutzt wurde. Wiederum wurde der zyklische Ablaufsteuerungsmodus mit 2 Hz Wiederholrate und der Eingangswertgenerierung per Zufallszahlgenerator gewählt. Die Interaktion des Testers mit der Simulation bestand in Eingaben im *„User Controls Panel"*, der Navigation in den Anzeigen des *„HTML Panel"* und einem zeilen- oder passagenweise Verschieben der Textausgaben des *„Explanation (SUM_UP & DETAILS) Panel"*. Die GUI wurde stets im Vordergrund, die Konsolenschnittstelle im Hintergrund gehalten. Die Option zur Speicherung der Diagnosedaten in der Datenbank wurde aktiviert, von der Option zur Ausgabe der „Debug"-Nachrichten auf der Konsolenschnittstelle wurde kein Gebrauch gemacht. Mit dieser Versuchsanstellung wurden, abgesehen von der Nutzerinteraktion, zahlreiche automatisierte

Messungen durchgeführt. Das nachfolgend dokumentierte Messergebnis für einen Testdurchlauf mit 5028 Schlussfolgerungszyklen steht stellvertretend für die Menge der durchgeführten Tests. Abbildung 5.9 gibt in Form eines Histogramms einen Einblick in gemessene Laufzeiten für einen Schlussfolgerungszyklus und deren Häufigkeitsverteilung.

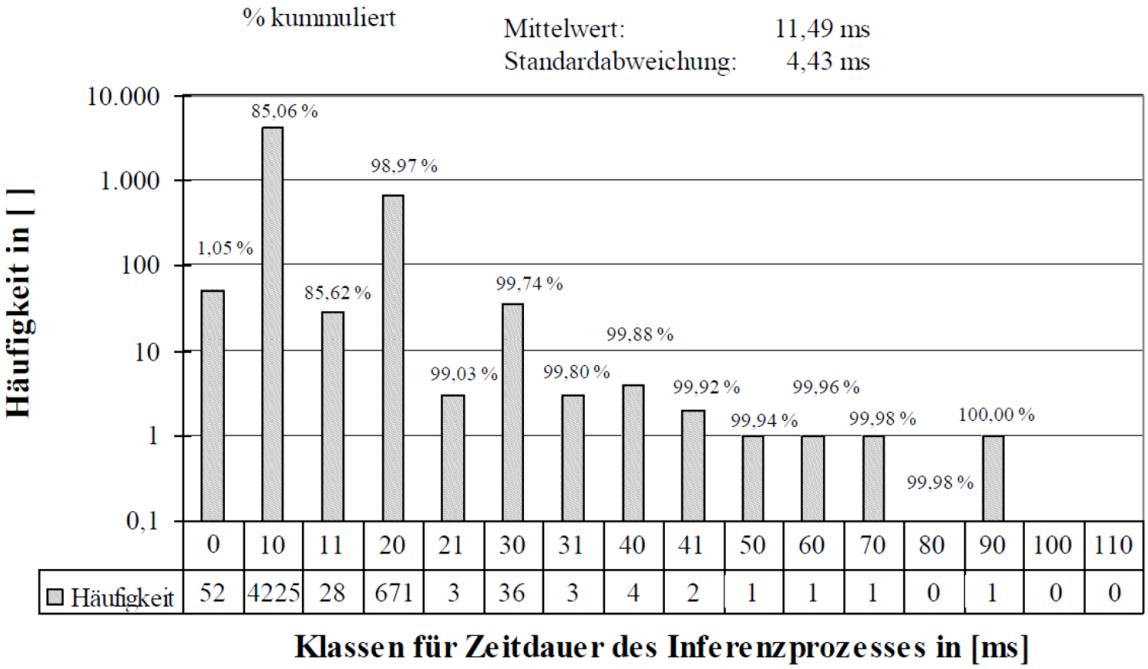


Abbildung 5.9: Histogramm zur Laufzeitmessung für den Schlussfolgerungsprozess auf der Testplattform II

Auf der Abszisse sind die einzelnen Klassen für die gemessenen Laufzeiten aufgetragen. Aus den gemessenen Daten war ersichtlich, dass bis auf den Wertebereich größer 90 ms keine Bereiche sondern diskrete Messwerte für die Klasseneinteilung gewählt werden konnten. Auf der Ordinate ist die Häufigkeit der einzelnen Klassen angegeben. Aufgrund der großen Spanne in der ermittelten Häufigkeitsverteilung wurde eine logarithmische Skalierung gewählt. Unterhalb der Abszisse ist neben der Klasseneinteilung auch noch die ermittelte Häufigkeit jeder einzelnen Klasse angegeben. Mit Blick auf die dreizehn Klassen ist klar ersichtlich, dass die gemessene Laufzeit von 10 ms mit einer Häufigkeit von 4225 bei einer Gesamtzahl von 5028 dominiert. Zusammen mit den Klassen 0 ms und 11 ms stehen diese drei Klassen für 86 % aller Schlussfolgerungsprozesse. Wird die zweitstärkste Klasse von 20 ms hinzugenommen so sind bereits 99 % aller Durchläufe abgedeckt. Mit den Klassen von 21 ms bis 41 ms ist das fehlende 1 % nahezu abgedeckt. Die Klassen 50 ms, 60 ms, 70 ms sowie der Maximalwert von 90 ms treten nur einmal auf. Größere Laufzeiten wurden nicht gemessen. Im Rahmen der Messgenauigkeit lässt sich für diesen Testlauf zur Bestimmung der

Laufzeit des Schlussfolgerungsprozesses ein Mittelwert von 11 ms und eine Standardabweichung von 4 ms errechnen.

Für diesen Test wurden für das „*CROP_PRODUCTION“*-Modul alle Entscheidungsbaumäste der Option mit einem REIP-Wert, abgedeckt. Das alternative Regelwerk ohne Berücksichtigung des REIP-Wertes stand nicht im Mittelpunkt des herangezogenen Testlaufes. Daher liegt für diesen Fall eine weitreichende aber keine vollständige Abdeckung der entsprechenden Entscheidungsbaumäste vor. Eine Häufigkeitsverteilung hinsichtlich der Testabdeckung wurde nicht erstellt. Da die maximalen Laufzeiten (*worst case*) für die Beurteilung der Echtzeitfähigkeit von besonderem Interesse sind, verdienen sie eine nähere Betrachtung. Den Schlussfolgerungsprozessen, die zu den Laufzeiten von 60 ms und 70 ms geführt haben, liegen schlussendlich die Auswahl des manuellen Betriebsmodus und zugleich einer Übersteuerungsanforderung von 80 % bzw. 120 % zugrunde. Die Laufzeiten von 50 ms und der *worst case* von 90 ms wurden bei Prozessen gemessen, für die der automatische Modus gewählt worden ist. Beide Male führten unterschiedliche Eingangswertkombinationen dazu, dass im „*CROP_PRODUCTION“*-Modul die Regel <N0-2> zur Anwendung kam. Eine weitere Analyse für die Laufzeiten aller Fälle, bei denen diese Regel gefeuert wurde, ergab, dass auch die kürzeren Laufzeiten 10 ms, 11 ms, 20 ms und 31ms gemessen wurden. Der Schwerpunkt lag dabei bei 10 ms.

Im Rahmen des Testens wurden Messungen der Laufzeit nicht nur für die Plattform II sondern auch für die restlichen drei Plattformen durchgeführt. Tabelle 5.4 listet dabei die jeweiligen maximalen Mittelwerte und Maximalwerte für die gesamte Menge der ausgeführten Tests unterschieden nach Testplattform auf.

Tabelle 5.4: Zusammenfassung der Mittel- und Maximalwerte der Laufzeit für einen Schlussfolgerungsprozess für verschiedene Plattformen

Plattform	Laufzeit Schlussfolgerungsprozess	
	Mittelwert	Maximalwert
I	10 ms	60 ms
II	11 ms	90 ms
III	10 ms	60 ms
IV	0 ms	16 ms

Für die Plattformen I-III liegen mit 10 ms bis 11 ms nahezu gleiche Ergebnisse für die Mittelwerte vor. Hingegen ergibt sich für die Plattform IV ein Mittelwert von 0 ms, der

dahingehend zu interpretieren ist, dass eine Laufzeit kleiner 1 ms vorliegt, was vom implementierten Messverfahren jedoch nicht besser aufgelöst werden kann. Die gemessenen Maximalwerte liegen für die Plattform I und III auf gleichem Niveau von 60 ms. Für die Plattform II wurde mit 90 ms der absolute Maximalwert aller Tests gemessen. Der geringste Maximalwert wurde auf der leistungsstärksten Plattform IV mit 16 ms gemessen und liegt damit schon fast im Bereich der Mittelwerte der leistungsschwächeren Plattformen.

Bei den beschriebenen Laufzeitmessungen war die Option zur Ausgabe von „Debug"-Nachrichten auf der Konsolenschnittstelle nicht aktiviert. Wurde diese Option zugeschalten, führte dies nur zu einer geringen zusätzlichen CPU-Auslastung, die Auswirkung auf die Laufzeit war jedoch erheblich. Wurde auf der Testplattform II die entsprechende Programmoption gewählt, die Konsolenschnittstelle jedoch noch nicht zur Anzeige gebracht, so führte dies zu einer im Mittel um 20 ms verlängerten Laufzeit für einen Schlussfolgerungsprozess. Wird zugleich die Konsolenschnittstelle zur Anzeige („Konsole im Vordergrund") gebracht, verlängert sich die Laufzeit um einen Wert aus dem Bereich von 500 ms bis 550 ms. Dies gilt für eine Konsolenschnittstelle, die den Bildschirm in XGA-Auflösung[46] fast vollständig ausfüllte. Die verlängerte Laufzeit verringerte sich proportional in dem Maße, in dem die Konsolenschnittstelle verkleinert oder durch die GUI teilweise überdeckt wurde.

Nicht nur die Schlussfolgerungsprozesse wurden einer Laufzeitmessung unterzogen, auch der lesende Zugriff auf die Datenbank wurde untersucht, um die minimal nötigen Rechenprozesse für den Sensor-Ansatz mit Kartenüberlagerung erfassen zu können. Dies ist der Zugriff auf Kartenmaterial, stellvertretend hierfür der Datenbankzugriff, und die anschließende Schlussfolgerung und Entscheidungsfindung. Der lesende Datenbankzugriff wurde anhand des Testszenarios mit den realen Testdaten für den dieser Arbeit zugrunde liegenden Feldtest durchgeführt. Die Gesamtzahl der hierzu genutzten Teilflächen- bzw. Rasterelemente war 208. Für die Laufzeit `time-diff-dbQuery` wurde ein Mittelwert von 34 ms bei einer Standardabweichung von 8 ms ermittelt. Der gemessene Maximalwert lag bei 90 ms.

[46] XGA entspricht einer Auflösung von 1024 Pixel horizontal x 768 Pixel vertikal.

6 Einordnung und Diskussion

Nachdem in den vorangegangenen Kapiteln die Ergebnisse ausführlich dargelegt wurden, sollen sie anschließend kritisch reflektiert und eingeordnet werden. Diese Einordnung und Diskussion folgt dabei wiederum der Gliederung nach den drei aufeinander aufbauenden Zielsetzungen:

- Ableitung einer Analyse- und Entwurfsmethode für eine auf MSDF basierende Real-time Prozessführung in einem mobilen landwirtschaftlichen BUS-System,
- Überprüfung auf Anwendbarkeit, Förderung des Verständnisses und Analyse der Auswirkungen auf landwirtschaftliche BUS-Systeme am Beispiel der Real-time Prozessführung in sensorgestützten Düngesystemen nach dem Sensor-Ansatz mit Kartenüberlagerung für intensive Stickstoffdüngung,
- Realisierung der theoretisch abgeleiteten Lösung in Form einer Software-Simulation mit intuitiver Interaktionsmöglichkeit und Test relevanter Kenngrößen der Fusionslösung.

Abgerundet wird das Kapitel mit einem generellen Blick auf die Herausforderungen und Grenzen von MSDF und der Frage, welche Erkenntnisse sich daraus für die Agrarsystemtechnik gewinnen lassen.

6.1 Analyse- und Entwurfsmethode

Bisher wurde im Bereich der Agrarsystemtechnik weder national noch international eine durchgängige Analyse- und Entwurfsmethode für eine Real-time Prozessführung für mobile Applikationssysteme unter Nutzung von MSDF dargelegt. Auch ist die besondere Berücksichtigung der mobilen landwirtschaftlichen bzw. landtechnischen BUS-Systeme bei dieser Thematik bisher nicht erfolgt. Die vorliegende Arbeit schließt mit dem Vorschlag des MSDF-Framework, einem Vorgehensmodell, diese Lücke.

Die Anwendbarkeit dieses abstrakten Vorgehensmodells wurde am Beispiel der Real-time Prozessführung in sensorgestützten Düngesystemen nach dem Sensor-Ansatz mit Kartenüberlagerung gezeigt. Zugleich bietet es einen anschaulichen Weg zur Durchdringung der rein theoretisch abgeleiteten Modellierung und kann als Muster für andere praktische Anwendungen herangezogen werden. Mit der Realisierung der Software-Simulation wurde (zumindest) für den ausgewählten Anwendungsfall der Beweis für die Durchgängigkeit der Methodik und die Umsetzbarkeit der Konzepte bis hin zu einem realen Prototypensystem erbracht.

Zur Themenstellung dieser Arbeit haben OSTERMEIER ET AL. (2003) [OAD03] und OSTERMEIER UND AUERNHAMMER (2004) [OA04] erstmals gezielt die theoretischen

Grundlagen bezüglich MSDF erarbeitet und als Terminologie eingeführt. AUERNHAMMER UND SCHUELLER (1999) [AS99] verwendeten bei dem Vorschlag für den Sensor-Ansatz mit Kartenüberlagerung für eine Real-time Prozessführung für teilflächenspezifische Applikation weder den Ausdruck MSDF oder *Sensor Fusion* noch forderten sie deren Einsatz. Der Ergebnisteil hat jedoch gezeigt, dass diese Forderung inhärenter Bestandteil des Sensor-Ansatz mit Kartenüberlagerung ist, und legt damit offen, dass die Wissenschaft die Relevanz des Themas früh erkannt hat. LINSEISEN ET AL. (2000) [LSHWSD00] haben das Thema in einem umfangreichen Verbundforschungsprojekt (IKB-Dürnast) aufgegriffen. Den Schritt vom stationären System auf das mobile System haben sie jedoch noch nicht vollzogen und den Ansatz unter dem Aspekt eines Informationssystems und nicht explizit als Aufgabenstellung der MSDF bearbeitet.

Die publizierten Arbeiten zu Ortung und Navigation (vgl. Kap. 2.2.1), zu Umfelderfassung (vgl. Kap. 2.2.2) sowie zu Zustandsüberwachung und Teleservice (vgl. Kap. 2.2.4) untersuchten und dokumentierten jeweils gezielt eine spezifische Fusionslösung bzw. die Gegenüberstellung weniger alternativer Algorithmen. Für den Bereich der Prozessführung für teilflächenspezifischen Pflanzenschutz vertraten RAMON ET AL. (2006) [RMBVD06] die Ansicht, dass ein „*Multisensor Fusion*"-Ansatz gegenüber einem Ein-Sensorsystem Vorzüge bei der Identifizierung von Pflanzenkrankheiten und Stresssymptomen aufweisen könnte. Frühe Ansätze für teilflächenspezifischen Pflanzenschutz [Bil03] wie auch aktuelle Arbeiten haben genau diesen Weg eingeschlagen [KTRM08]. Die Relevanz und Aktualität des Themas der MSDF im Präzisen Ackerbau spiegeln neue Recherche- [ARSS11] und Forschungsarbeiten [BBMK11] wider.

Auffällig ist, dass bisher keine eindeutige Terminologie in der Agrarsystemtechnik verwendet wird. Die aktuellen Arbeiten sprechen von „*Sensor Fusion*", ältere Veröffentlichung teilweise von „*Multisensor Fusion*" oder nur von einem „Multisensorsystem". Bei vielen der vorgestellten Arbeiten trifft aufgrund ihrer engen Zielsetzung die Nutzung einer Namensgebung zu, die nur eine Teilmenge der MSDF erfasst. Wenn jedoch der Inhalt der Veröffentlichung eindeutig über die in der Überschrift genannte „*Sensor Fusion*" hinausgeht, wie z.B. in den beiden angeführten Arbeiten jüngeren Datums, wird der Bedarf nach einem gemeinsamen Verständnis und der Nutzung einer eindeutigen Terminologie in der Agrarsystemtechnik deutlich erkennbar. Die vorliegende Arbeit hat dies in der Begründung für die allgemeine Zielsetzung aufgeführt und eine Lösung ausgehend von dem Begriff *Data Fusion* dafür angeboten. Ein Forschungsprojekt wie iGreen, mit der Zielsetzung der Konzeption und Realisierung eines standortbezogenen Dienste- und Wissensnetzwerks zur

Verknüpfung verteilter, verschiedener, öffentlicher, wie auch privater Informationsquellen für den Agrarbereich (vgl. [BBMK11]), fordert bereits die nächste Ausbaustufe der Zielsetzung dieser Arbeit und stößt die Diskussion über *Information Fusion* an. Laut STEINBERG UND BOWMAN (2001) [SB01] ist diese Fusionsart eine Teilmenge von *Data Fusion* ebenso wie *Sensor Fusion* und besitzt mit Letzerer eine Schnittmenge. Die Diskussion wird an dieser Stelle nicht weitergeführt, sondern konzentriert sich auf die Zielsetzung und Ergebnisse dieser Arbeit.

Es ist unstrittig, dass es sich bei dem MSDF-Framework um keine neue Erfindung sondern um die Auswahl und Komposition von bekannten Modellen handelt. Neu ist jedoch die Anpassung an die Anforderungen der Agrarsystemtechnik und die Demonstration, dass es für dieses Fachgebiet ein gangbarer Lösungsweg ist. Dieses Vorgehen ist nicht unüblich für ein multidisziplinäres Fachgebiet, das die theoretischen Arbeiten in einem Wissenschaftsbereich nicht ursprünglich entwickelt und angetrieben hat, jedoch zu einem bestimmten Zeitpunkt die Relevanz und Eignung für eigene Fragestellungen erkennt. Die treibenden Kräfte für MSDF wurden im Stand der Technik (vgl. Kap. 2.1.1) benannt. Agrarsystemtechnik ist ein typischer „*follower"*, der von der in anderen Anwendungsbereichen bewährten Technik profitiert bzw. profitieren kann, aber seine eigenen technischen, sozialen und wirtschaftlichen Rahmenbedingungen berücksichtigen muss. Ein Vergleich mit den Anfängen der Landwirtschaftlichen BUS-Systeme macht das Vorgehen anschaulich. Das von der Fa. Robert Bosch GmbH, Stuttgart in den 1980er-Jahren entwickelte CAN-Feldbussystem, das erste vielversprechende herstellerspezifische Anwendungen in der Automobiltechnik fand, wurde von der Agrarsystemtechnik als Grundlage für LBS und auch ISOBUS ausgewählt (vgl. Kap. 2.4). Mit einer Verzögerung verglichen mit dem Start der LBS-Normung hatte auch die amerikanische Nutzfahrzeugindustrie die Möglichkeiten erkannt und mit der Normung des SAE-Standards [SAEJ1939] begonnen. Im Jahr 1993 wurde die ISO-Norm [ISO11898] für den Automobilbereich auf Basis der CAN-Spezifikation veröffentlicht. CAN erlangte seine schon fast zwei Jahrzehnte andauernde herausragende Stellung für die Datenkommunikation im KFZ-Serienfahrzeug und damit eine weitreichende Marktdurchdringung. Auch in modernen Traktoren und Landmaschinen ist der CAN-Bus nicht mehr wegzudenken. Mittlerweile wurden umfangreichste standardisierte Protokolldefinitionen für das Zusammenspiel von dynamisch konfigurierbaren Traktor-Geräte-Kombinationen definiert, die den Vergleich mit der Automobiltechnik und dem Nutzfahrzeugbereich nicht scheuen müssen bzw. bereits deren Komplexität weit übersteigen. So hatte die Agrarsystemtechnik bei der Entwicklung und Umsetzung der grundlegenden Technik die Rolle eines *„follower"* inne,

jedoch hat sie frühzeitig deren Potential und Eignung für eigene Aufgabenstellungen erkannt. Konsequenterweise stand sie daher bei der Initiierung von Standardierungsaktivitäten an der Spitze und hat für die eigene Fachdisziplin auf nationaler und internationaler Ebene einen Standard entwickelt, der die komplexen Anwendungsszenarien (der Prozesstechnik) der Aussenwirtschaft adressieren kann.

Bevor die einzelnen Elemente des Vorgehensmodells kritisch betrachtet werden, soll noch ein kurzer Blick auf die Einordnung im Gesamtkomplex einer Prozessführung geworfen werden.

An erster Stelle steht eine Gesamtsystemanalyse. Nur wenn daraus eine potentielle Vorzüglichkeit für den Einsatz von MSDF gefolgert werden kann, ist eine Begründung für den effizienten Einsatz dieser Technik gegeben. MSDF ist stets nur ein Teil einer Automatisierungs- bzw. Prozessführungslösung. WALTZ UND HALL (2001) [WH01] drückten dies sehr prägnant aus: „*... there is no such thing as a data fusion system. Instead, there are applications to which data fusion techniques can be applied. ...*“[47] Daher muss diese Technik sich in das Gesamtsystem einfügen und löst meistens Herausforderungen auf niedriger Fusionsebene (z.B. vorverarbeitete Maschinen- und Prozessdaten) oder erweitert die Systemlösung um einen „intelligenten Faktor“, zumeist in Richtung wissensbasierte Lösung (vgl. Kap. 2.3.2.1.3). Daraus folgt, dass das MSDF-Framework sich in den Entwicklungsprozess des Gesamtsystems einzufügen hat. Das vorgeschlagene Vorgehensmodell hat hierzu einen Weg eingeschlagen, der unterschiedliche Abstraktionsebenen und Betrachtungsebenen für die einzelnen Schritte des Vorgehensmodells nutzt und hiermit eine gewisse Flexibilität bei der Integration in den Entwicklungsprozess des Gesamtsystems bietet. Die grundsätzlichen Fragen des Funktionalen Modells („*Was“*), des Prozessmodells („*Wie*“) und der Systemarchitektur („*Wodurch*“) finden sich in der Mehrzahl aller Entwicklungsprozesse (vgl. Kap. 2.5) wieder. Durch das Bestreben, stets eine möglichst abstrakte Antwort zu erzielen, die erst in späten Phasen der Systemarchitektur und des Feinentwurfes einen spezifischen Implementierungsvorschlag definiert, wurde Flexibilität, Konzentration auf das Wesentliche und Zukunftssicherheit in die Entwurfs- und Analysemethode eingebaut.

Das MSDF-Framework geht im Grunde vom Ideal des „*top down”* Entwurfs aus und fügt sich damit nahtlos in die bisher in der Agrarsystemtechnik dominierenden Entwicklungsprozesse Wasserfall- oder V-Modell ein. Verglichen mit dem V-Modell, das derzeit für die Einhaltung der funktionalen Sicherheit die erste Wahl ist (vgl. Kap. 2.5), kann folgende Zuordnung

[47] A. a. O., S. 2

getroffen werden. Das Funktionale Modell ist der Ebene der Anforderungsdefinition und Validation zugeordnet. Das Prozessmodell lässt sich im Rahmen des Grobentwurfs und der Verifikation des Problemlösungsparadigma auf abstrakter Ebene nutzen. Auch die Systemarchitektur ist auf Ebene des Grobentwurfs und der Verifikation angesiedelt. Die Phasen des Feinentwurfs, die Modulimplementierung und die zugehörige Verifikation folgen der Ursprungsdefinition des V-Modelles.

Das MSDF-Framework sperrt sich jedoch nicht gegen Anwendung von eher „*bottom up"* orientierten Ansätzen auf Ebene der Systemarchitektur und der nachfolgenden Umsetzung. Wird in Zukunft ein Zustand erreicht, dass MSDF-Lösungen für die Agrarsystemtechnik „baukastenartig" vorliegen, so wird das Modell des „*System Architecting"* oder ein hybrider Ansatz bevorzugt zur Anwendung gebracht werden. Hinsichtlich des Einsatzes von „*System Engineering"* oder „*System Architecting*" kann festgestellt werden, dass bei Erstem der Entwurfsaspekt des MSDF-Framework überwiegt, während bei Zweitem der Analyseaspekt in den Vordergrund tritt, um die relevanten „*use cases"* und die dazu passenden Implementierungs-„Bausteine" identifizieren zu können. Die Forderung nach dem „*Agile Software Development*" [LM05] fördert ein Vorgehen nach dem hybriden Ansatz. Jedoch kommt auch dieser „*Agile Approach*" nicht ohne ein stabiles Grundgerüst aus, in das in schneller Iterationsfolge neue Produktmerkmale eingebaut und dann getestet und gewartet werden. Gerade für die Entwicklung des Grundgerüstes kommt auch dieser Ansatz nicht ohne grundlegende Analyse- und Designschritte mit Ableitung einer Systemarchitektur aus. Ein rein evolutionäres Entwicklungsmodell zur Serienentwicklung wurde für die Agrarsystemtechnik bisher noch nicht vorgestellt. Der Einsatz des MSDF-Framework sollte auch hierbei helfen, Irrwege früh aufzuzeigen und damit die Iterationsanzahl zu verringern, d.h. trotz des evolutionären Grundprinzips für Zielorientierung und Effizienz sorgen. Was diese Arbeit jedoch klar zeigt, ist, dass die MSDF-Anforderung in der Ableitung eines wissensbasierten Problemlösungsparadigma resultieren kann und somit im Rahmen eines Gesamtsystem-Entwicklungsprozesses nach Wasserfall- oder V-Modell für die Phasen Feinentwurf und Modulimplementierung auch die Anwendung eines evolutionären Vorgehens erfordern kann. Wie sich eine so komplett andere Vorgehensphilosophie dennoch möglichst widerspruchsfrei integrieren lässt, wurde im Rahmen dieser Arbeit bei der Realisierung der Software-Simulation erprobt (vgl. Kap. 5.1.3.1 und Kap. 6.3.1).

Durch die Abstraktion spielen sowohl beim Funktionalen Modell als auch beim Prozessmodell landwirtschaftliche BUS-Systeme keine Rolle. Erst ab dem Punkt, an dem grundsätzliche Problemlösungsformen abgeleitet wurden, erfolgt auf Ebene der

Systemarchitektur die Anpassung an die konkreten Rahmenbedingungen für die Real-time Prozessführung in einem mobilen landwirtschaftlichen BUS-System. Während die als optimal abgeleitete Problemlösungsform als Konstante angesehen werden kann, ist es möglich, die Umsetzung flexibel an geltende Standards, die Verfügbarkeit und den bisweilen rasanten technischen Fortschritt bei Sensoren, Aktoren und informationsverarbeitender Technik anzupassen. Neben der Existenz einer singulären Lösung oder von Lösungsalternativen kann dies auch zur Erkenntnis führen, dass eine Norm geändert bzw. neu definiert werden sollte oder der herrschende Stand der Technik die Umsetzung der Problemlösungsform nicht ermöglicht oder unrentabel macht. Die Handlungsempfehlungen hängen vom Einzelfall ab.

Dem Aspekt der funktionalen Sicherheit wurde keine besondere Rolle bei der Definition der Entwurfs- und Analysemethode zuteil. Ähnlich wie bei der Berücksichtigung der landwirtschaftlichen BUS-Systeme wird angenommen, dass die Beachtung der funktionalen Sicherheit über die Anpassung in der Systemarchitektur und/oder der Qualitätssicherung erfolgen kann. Sollte diese Annahme sich als zu optimistisch erweisen, wäre die erste Fragestellung, ob der MSDF-Anteil in der Prozessführung überhaupt sicherheitskritisch ausgelegt werden muss bzw. auf welchen sicherheitskritischen Kernbereich er sich verringern lässt.

Als alternativer Ansatz für das vorgeschlagene Vorgehensmodell für MSDF für Real-time Prozessführungen kommt am ehesten die Arbeit von BLANK ET AL. (2011) [BBMK11] in Frage, da das Flussdiagramm von ADAMCHUK ET AL. (2011) [ARSS11] zur „*Data integration in precision agriculture*“ im weitesten Sinn nur als eine Teilmenge eines funktionalen Modells fungieren könnte. BLANK ET AL. (2011) [BBMK11] hingegen gehen zur Ableitung ihres „*Modular Sensor Fusion Approach for Agricultural Machines*“ von einer abstrahierten Datensicht aus und entwerfen dann eine Systemarchitektur, die die Ebenen „*signal alignment*“, „*signal level fusion*“ und „*state classification*“ realisiert. Dabei wird großes Gewicht darauf gelegt, dass die Fusion mit den begrenzten Ressourcen (Rechnerleistung, Speicher und Übertragungsgeschwindigkeit) aktueller ECU‘s und der dynamischen Rekonfiguration von Traktor-Geräte-Kombinationen zurechtkommt. Im Grunde beschreiben die Ebenen eine Einteilung in Funktionen des *JDL Level 1 und 2 Processing* sowie Maßnahmen zu *JDL Level 4 Processing* für die Rekonfigurationsfähigkeit. Ein Prozessmodell kommt nicht zum Einsatz, sondern MSDF-Algorithmen werden im Rahmen der Systemarchitektur nach oben angeführten Kriterien ausgewählt, am besten vergleichbar mit der Anwendung von Hall's Taxonomie (vgl. Kap. 2.1.5.1). Die einzelnen abgeleiteten Techniken für die systemübergreifend konsistente Maschinen- und Prozessdatenaufbereitung

(*JDL Level 1 Processing)* oder die Zustandsbestimmung (*JDL Level 1 und 2 Processing)* sind durchaus vielversprechend. Ein durchgängiges Vorgehensmodells stellt der Ansatz jedoch nicht dar, auch ist er nicht auf die Real-time Prozessführung und die landwirtschaftlichen BUS-Systeme ausgerichtet, wenngleich im Ausblick darauf hingewiesen wird, dass eine Anpassung und Anwendung auf den ISOBUS interessant erscheint.

Abschließend soll noch das Funktionale Modell, das Prozessmodell, die Systemarchitektur und der Feinentwurf einzeln erörtert werden.

Da das Funktionale Modell an oberster Stelle des Abstraktionsniveaus steht, ist es möglich, dass ein breiter Nutzerkreis die Anforderungen und die Rahmenbedingungen an ein System definieren kann, aber auch bereits auf dieser Ebene den Nutzen und die grundsätzliche Leistungsfähigkeit überprüfen kann. Die Begründung für die Auswahl des „*Revised JDL data fusion model*“ wurde in Kapitel 3.1 gegeben. Es bleibt festzuhalten: die Erstellung des Funktionalen Modells für einen konkreten Anwendungsfall ist die wertvolle Basis und Vorarbeit zu der Ableitung einer Problemlösungsform über die Anwendung des vorgeschlagenen Prozessmodells. Ebenfalls in Kapitel 3.1 wurde darauf hingewiesen, dass das Prozessmodell nach Antony andere Modelle mit einschließt bzw. deren Funktionalität mit abdeckt. Dem Ausgangspunkt, menschliche und tierische Fähigkeiten in dem Bereich MSDF mittels eines technischen Systems nachzubilden, liegt die Methode zugrunde, ein erfolgreiches Praxisbeispiel zu kopieren und daraus theoretische Grundprinzipien abzuleiten. Der Ansatz Fusionsprozesse auf unterschiedliche Wissensarten zurückzuführen, hat den Vorteil, dass er sehr allgemein, und damit auch zeitlos und weitgehend unabhängig vom Stand der Technik ist.

Als Mangel des Prozessmodells ist festzustellen, dass es sich um keinen formalen informationstheoretischen Ansatz handelt. Verglichen mit den alternativen Prozessmodellen bietet er jedoch eine strukturierte Vorgehensweise über die Ableitung von Fusionsklassen und kanonischen Problemlösungsformen an. Die reine Verwendung von Hall's Taxonomie gleicht im positiven Fall der Auswahl eines Algorithmus durch Ähnlichkeit der „*use cases“*. Im negativen Fall ist sie bestimmt von den Kenntnissen und Vorlieben des Systemdesigners bzw. den Vorgaben des Auftraggebers und gerät in Gefahr, in einem „*trial and error*“-Verfahren zu enden (vgl. auch [Ant95][48]). Es sei bemerkt, dass der beste Fall für „*System Architecting“* geeignet ist.

Spätestens wenn ein Problemlösungsparadigma hergeleitet wurde, kann der Mangel einer

[48] A. a. O., S. 27

fehlenden theoretischen Grundlage für den oder die *Data Fusion*-Algorithmen überwunden werden. Für die einzelnen Algorithmen existieren Nachschlagewerke mit einer Sammlung der mathematischen Grundlagen und einer umfangreichen Liste relevanter Literaturreferenzen [WL90], [HM04]. Selbst wenn keine (mathematisch) analytischen Modelle zugrunde liegen, sondern heuristische Modelle (vgl. [Kre91]) benötigt werden, stellt die Informatik die theoretischen Grundlagen bereit. Die Qualität des Modells wird jedoch dann direkt durch Qualität der Heuristik bestimmt, die von Fachexperten beizusteuern ist. Für den Anwendungsfall dieser Arbeit bedeutet dies, dass das Wissen der agrarwissenschaftlichen Disziplinen Pflanzenbau, -züchtung, -ernährung, Bodenkunde und Betriebswirtschaftslehre dafür entscheidend ist. Für diese wissensbasierten Lösungen kann die Agrarsytemtechnik im Idealfall eine optimale Prozessführungs-„Infrastruktur“ entwerfen. Sie ist jedoch auf das Fachwissen der anderen Disziplinen angewiesen, damit die Prozessführung auch ein optimales Ergebnis vollbringt (vgl. Kap. 6.3.2). An der Überleitung zur Systemarchitektur soll darauf hingewiesen werden, dass eine alternative Art der kooperativen Aufgaben-/Problemlösung, der Einsatz von sogenannter Schwarmintelligenz oder *intelligent agent systems* [HM04] nicht explizit in dem Prozessmodell hervorgehoben wurde.

Auch der Systemarchitekturvorschlag des MSDF-Framework beschreitet bewährte Wege und rät zur Auswahl von bekannten Architekturen. Die genannten neueren kooperativen Ansätze, die aktuell Gegenstand der Forschung (z.B. [RL11]) sind, berücksichtigt er nicht. Es spricht jedoch nichts dagegen auch diesen Architekturvarianten eine höher Bedeutung beizumessen, falls die Annahme eines lokalen Charakters einer Prozessführung nicht mehr gültig ist. In dieser Arbeit stand eine Traktor-Düngerstreuer-Kombination im Mittelpunkt, die im lokalen Einsatz mit eigenem *Data Fusion*-Prozessor die Applikation unter Echtzeitbedingungen erbringt und von extern nur über das FMIS oder ein WSN Eingangsinformation erhält. Der lokale Charakter würde jedoch verloren gehen, falls ein hochgradig verteiltes System zur Erbringung der Applikationstätigkeit angestrebt wird, z.B. bestehend aus mehreren Traktor-Geräte-Kombinationen und/oder der Nutzung von service-orientierten Strukturen, d.h. mehrere Web Services erbringen die „*In-field Controller*“-Funktionalität.

Entscheidend auf der Ebene der Systemarchitektur ist, dass sich die Ergebnisse aus dem funktionalen Modell und speziell dem Prozessmodell abbilden und wiederfinden lassen. Die Thematik der Anpassung an die Randbedingung, den Stand der Technik, der Skalierbarkeit, das Erkennen des Machbaren und der zukünftigen Weiterentwicklung wurde vorhergehend bereits aufgegriffen und diskutiert. So bleibt nur eine Bemerkung zur vorrangigen Auswahl des ISOBUS (ISO 11783) für die Randbedingung des mobilen landwirtschaftlichen BUS-

Systems. Im Kapitel 2.4 wurde aufgezeigt, dass es die derzeit einzige (veröffentlichte) Protokolldefinition ist, die von einer wachsenden Anzahl von Landmaschinenherstellern unterstützt und von der Praxis angenommen wird. Da ISOBUS (und schon LBS) bereits von Anfang an die grundlegenden Systemelemente und Systemarchitektur-Philosophien für Prozessführungen miteinschließt, ist anzunehmen, dass auch zukünftige Weiterentwicklungen im Bereich der Landwirtschaftlichen BUS-Systeme zumindest diese Grundideen weiterentwickeln und weitertragen.

Der abschließende Punkt im MSDF-Framework, der Feinentwurf und die Implementierung, wurde nahezu offen gehalten, obwohl er in der Regel den größten Teil der Entwicklungsressourcen beansprucht. Dies basiert auf der Annahme, dass es der ausführenden Stelle obliegt, die Schritte nach ihren Unternehmensleitlinien, Interessen und dem Stand der Technik durchzuführen. Dies kann eine wissenschaftliche Stelle mit Interesse an einem *proof of concept* sein, ein Unternehmen mit Gewinnerzielungsabsicht und Haftungs- und Gewährleistungspflichten oder z.B. eine (gemeinnützige) Vereinigung, die ein „*Open Source"* -Produkt generiert. Ihre Ausgestaltung dieser Phasen wird sich unterscheiden.

Die Schritte werden sich auch je nach gewähltem Problemlösungsparadigma unterscheiden. Für diese Arbeit kam folglich ein Verfahren zur Entwicklung eines wissensbasierten Systems zum Einsatz (vgl. Kap. 5.2), geleitet durch eine nicht gewinnorientierte wissenschaftliche Sichtweise.

6.2 Real-time Prozessführung

Die Begründung für die Auswahl des anwendungsorientierten Teilziels wurde bereits in der Zieldefinition ausführlich dargelegt (vgl. Kap. 3.2). Die Auffassung wird auch nach der Zusammenstellung des Standes des Wissens und der erarbeiteten Ergebnisse aufrecht erhalten. Die Gesamtsystemanalyse hat eine Eignung und Forderung nach MSDF nachgewiesen. Die Aufstellung der Rahmenbedingung ist nachvollziehbar. Auch die Bewertung der Echtzeitfähigkeit ist schlüssig, zeigt aber auch dass eine Risikobewertung durchzuführen ist, die je nach Gewichtung ebenso zu unterschiedlichen Ergebnissen führen kann. Dies gilt ebenfalls für die Abschätzung des zeitlichen Rahmens für einen MSDF-Prozess, so dass die Echtzeitbedingung eingehalten werden kann. Die gewählten Parameter sind durchaus realistisch, es ließen sich jedoch auch ungünstigere Kombinationen finden. Generell ist stets das Denken im System gefordert und auch bei der Systemauslegung zu beachten.

Mit der Anwendung des MSDF-Framework auf eine Real-time Prozessführung in

sensorgestützten Düngesystemen nach dem Sensor-Ansatz mit Kartenüberlagerung für intensive Stickstoffdüngung und der Realisierung als Simulation konnte der im Rahmen des IKB-Dürnast-Forschungsvorhabens von LINSEISEN (2001) [Lin01] noch nicht durchgeführte Schritt ins mobile System vollzogen werden. Zugleich konnte mit einem gänzlich anderen Ansatz als bei WEIGERT (2006) [Wei06] die grundsätzliche Eignung einer Heuristik zur kleinräumigen N-Düngung hergeleitet werden, die auf Entscheidungsregeln basiert. Dem von WEIGERT über Data Mining und Wissensentdeckung erarbeiteten heuristischen Wissen konnte eine optimale Prozessführungs-„Infrastruktur“ zur Seite gestellt werden.
Auf der Grundlage obiger allgemeinen Einordnung kann eine grundsätzliche Anwendbarkeit der Entwurfs- und Analysemethode angenommen werden. Das Anwendungsbeispiel ist gerade bei den abstrakten Modellen eine Hilfe den Nutzen und die Methodik besser zu verstehen. Im weiteren Verlauf wird eine tiefergehende Betrachtung der Ergebnisse auf den Ebenen Funktionales Modell, Prozessmodell und Systemarchitektur durchgeführt. Dabei deckt der Abschnitt zur Systemarchitektur auch die Analyse der Auswirkungen auf landwirtschaftliche BUS-Systeme ab. Eine knappe verfahrenstechnische Einordnung rundet das Kapitel ab.

6.2.1 Funktionales Modell

Der MSDF-Anteil der Prozesssteuerung wurde in Form einer Mischung aus Anforderungsdefinition und „*use case*“ -Beschreibung formuliert und setzt keine besondere Kenntnis von (graphischen) Modellierungswerkzeugen voraus. Daher ist beim vorgestellten Modell auf hoher Abstraktionsebene davon auszugehen, dass auch Experten aus dem Bereich der Agrarwissenschaften oder der Landtechnikindustrie ohne tiefere Kenntnisse der Automatisierungstechnik und Informatik grundsätzlich beurteilen können, wie die Lösung für den vorgeschlagenen Systemansatz strukturierbar ist, aber auch welche Abhängigkeiten und Forderungen daraus entstehen. Manch technisch orientierter Leser mag sich eine mehr formale Darstellung wünschen, aber eine weitestgehende Verständlichkeit für einen breiten differenzierten Nutzerkreis ist auf dieser Abstraktionsebene zielführend.
Unter Zuhilfenahme der Vorarbeit in dem Stand der Technik konnte der Sensor-Ansatz mit Kartenüberlagerung allen entsprechenden *Revised JDL Processing Level* und Zusatzfunktionen zugeordnet werden bzw. fand eine Entsprechung. Eine Betrachtung der Ergebnisse für die einzelnen Elemente des Modells wird im Folgenden zusammengestellt.

Informationsquellen: Selbst beim aktuellen Stand der Kommunikationstechnik ist eine lückenlose Verfügbarkeit von drahtloser (breitbandiger) Kommunikation im ländlichen

Raum nicht stets gegeben oder führt zu merkbaren Zusatzkosten. Daher wird die Annahme nicht in Frage gestellt, dass kartierte Informationen zu einer lokalen Informationsquelle gewandelt werden soll.

Level 0 Processing: Die geforderte Funktionalität auf dieser Ebene ist sehr spezifisch für jeden einzelnen Sensor oder jedes Kartenmaterial. Die Arbeit geht davon aus, dass diese Funktionalität von den Herstellern in geeigneter Form bereitgestellt wird.

Level 1 Processing: Die Zuordnungen der Anforderungen aus der Gesamtsystemaufgabenbeschreibung auf die Teilaspekte dieser Ebene weist keine überraschenden Stellen auf. Zur Förderung des Verständnisses soll jedoch ein Aspekt nochmals herausgehoben werden, der bereits in der Einleitung zum Prozessmodell (vgl. Kap. 4.3) am Beispiel des Pflanzenbedeckungsgrades besprochen wurde. Level 1 Processing-Funktionalität kann mit auf der Level 2 Processing-Ebene gewonnenem oder vorhandenem Kontextwissen unterstützt bzw. erst ermöglicht werden. Diese über Ebenen reichende Data Fusion wird im Prozessmodell mit der Eigenschaft „mehrfache Abstraktionsebenen" (multi level of abstraction) bezeichnet. Es ist anzunehmen, dass derzeit die Bestimmung der Identität beteiligter Objekte wie z.B. der Pflanzensorte und des eingesetzten Stickstoffdüngers keine Fusionsaufgabe ist, sondern mit den Auftragsdaten vom FMIS zum MICS übergeben wird. Erst der Einsatz von optischen (z.B. Barcodes) oder funkbasierten (z.B. RFID) Identifikationssystemen könnte dies ändern.

Level 2 Processing: Diese Ebene ist sicherlich das Kernstück der funktionalen Modellierung des Sensor-Ansatzes mit Kartenüberlagerung. Die dort festgelegte Unterteilung, wie die Situationsbewertung durchgeführt werden sollte, ist der Ausgangspunkt und das Gerüst zur Erstellung des Prozessmodells und weist auch schon den Weg für die Untergliederung in Wissensmodule in der Simulation.

Level 3 Processing: Die auf dieser Ebene formulierte Funktionalität kann als Blaupause für einen Ausblick und eine Weiterentwicklung des Systemansatzes gesehen werden. Es zeigt aber bereits auf dieser funktionalen Ebene die daraus resultierende Komplexitätssteigerung.

Level 4 Processing: Unter Beachtung der Festlegungen für die anderen Ebenen handelt sich um die konsequente Beschreibung der nötigen Schritte zur Prozessoptimierung. Da die Prozessoptimierung letztendlich sehr stark mit der Systemarchitektur und der konkreten Implementierung interagiert, ist anzunehmen, dass die Einhaltung der Abstraktion schwieriger als bei anderen Ebenen ist.

Level 5 Processing – bzw. Mensch-Maschine-Schnittstelle: Die Einschränkung auf die Mensch-Maschine-Schnittstelle und Ablehnung einer „Level 5 Processing (cognitive refinement)" Ebene wurde im Ergebnisteil bereits begründet. Wie beim Anwendungsbeispiel einer Real-time-Prozessführung für Applikationstätigkeit stehen bei

den Steuerungs- und Regelungsaufgaben von landwirtschaftlichen Traktoren und Maschinen ja gerade die Automatisierung der Aufgabe und die Entlastung des Nutzers im Vordergrund. Daher ist es folgerichtig, die Erweiterung als nicht nötig zu erachteten. Die vorgeschlagene einfache Strukturierung der Mensch-Maschine-Schnittstelle in einen weitgehend einfachen Nutzermodus und einem sogenannten Experten-Modus findet sich in einer Vielzahl von Produkten. Im Prinzip wurde in diesem Abschnitt des Funktionalen Modells bereits die Spezifikation für die GUI und die Kommando/Konsolen-Schnittstelle der Simulation (vgl. Kap. 5.2.3) vorab definiert.

Datenbank-Management-System: Die vorgenommene Zuordnung lässt keine Fragen offen.

Abschließend lässt sich hinterfragen, ob das angewandte Modell für das Zielobjekt nicht zu groß angelegt ist? Anderseits bietet es jedoch einen strukturierten Zugang zur Definition der Anforderungen und der Ausgestaltung des Systems auf hoher Abstraktionsebene. Es ist eher ein Vorteil des Modells, dass es, wie eben auch mit dem Anwendungsbeispiel gezeigt, die für Zukunftssicherheit nötige Skalierbarkeit aufweist und auch zukünftige verteilte Lösungen mit drahtlosen Datenübertragungen und Web-Service gestützten-Lösungen ohne Schwierigkeiten abbilden könnte.

6.2.2 Prozessmodell

Auch das Prozessmodell wurde in Textform, ergänzt um wenige anschauliche Abbildungen, formuliert und ist damit wie bereits das Funktionale Modell einem weiten Nutzerkreis zugänglich. Der Anwendungsfall war in seiner Gesamtheit auf diesem Abstraktionsniveau modellierbar und führte zur Ableitung einer geeigneten Problemlösungsform. Der Weg dorthin wurde mit den angestellten Annahmen und den getroffenen Analyse- und Entwurfsentscheidungen transparent wiedergegeben. Die Aufteilung des Vorgehens in die beiden Schritte „Zuordnung der Wissensarten“ und „Bestimmung des Problemlösungsparadigmas“ trägt zur Transparenz und Nachvollziehbarkeit bei.

Beim ersten Schritt der „Zuordnung der Wissensarten“ wird der Wert eines gut vorbereiteten Funktionalen Modells erkennbar. Beim zweiten Schritt der „Bestimmung des Problemlösungsparadigma“ ist der Lösungsweg gut dokumentiert, aber der Leser, dem die Referenzliteratur [Ant95] nicht vorliegt, könnte beanstanden, dass der Einblick in die alternativen Problemlösungsformen fehlt. Da etwa 1/3 der Referenzliteratur dem Prozessmodell und der Anwendung gewidmet sind, ist eine Aufnahme in den Anhang kein gangbarer Weg. In der folgenden Diskussion wird daher an wenigen relevanten Stellen eine alternative Lösungsform angesprochen.

Wenngleich die vorgeschlagene Methode, einen geradlinigen Weg aufzeigt, wäre es ein

Trugschluss anzunehmen, dass eine 1-1 Zuordnung zwischen der Fusionsklasse und einem Problemlösungsparadigma existiert, d.h. eine Fusionsklasse kann durch unterschiedliche Problemlösungsformen umgesetzt werden.

Im weitesten Sinne grenzt die bestimmte Fusionsklasse die möglichen adäquaten Problemlösungsparadigmen auf eine Untermenge ein. Über den zweiten Schritt der Betrachtung der erforderlichen Algorithmus-Robustheit, Kontext-Sensitivität, Effizienz und Erweiterbarkeit lässt sich daraus eine geeignete kanonische Problemlösungsform und dafür typische Algorithmen ableiten. Spätestens an diesem Punkt ist auch diese Vorgehensweise nicht mehr ganz frei von den Vorkenntnissen des Systemdesigners oder -analysten. Allgemein bleibt festzuhalten, dass dieser Vorgang von der Zielsetzung geleitet war, die Problemlösungsform mit der niedrigsten Komplexität auszuwählen, die alle gestellten Anforderungen erfüllt.

Die entscheidende und folgenreiche Festlegung, die bei der Modellierung getroffen wurde, ist die Einschränkung auf eine einfache Abstraktionsebene (*single level of abstraction*), da ein „*bottom up*“ Ansatz als gültig angenommen wird. Wie bereits aus der Ergebnisbeschreibung (vgl. Kap. 4.3) ersichtlich, ist dieses Verständnis von „*bottom up*“ Ansatz ein anderes als im Zusammenhang mit dem „*System Architecting*“-Entwicklungsprozess (vgl. Kap. 2.5). Die Einschränkung, sich auf das *Level 2 Processing* und damit auf eine *single level of abstraction*-Lösungsform zu konzentrieren, kann bereits auf Ebene des Funktionalen Modells überprüft werden. Die gegebene Begründung ist schlüssig und wurde auch im IKB-Projekt akzeptiert. Es bleibt festzuhalten, dass diese Beschränkung auch zu einer Reduktion der Komplexität nicht nur im Prozessmodell, sondern vor allem auf den Ebenen Systemarchitektur und Implementierung führt. Dies ist jedoch nicht als ursächliche Motivation akzeptabel, sondern ist nur ein weiteres Entscheidungskriterium, falls die grundsätzliche Zulässigkeit der Einschränkung nachgewiesen wurde.

Wie bereits angemerkt wurde, lassen sich beim ersten Schritt der „Wissensartenzuordnung“ die bei der funktionalen Modellierung des *Level 2 Processing – Situation Assessment* ermittelten Objekte und Beziehungen den vier Wissensarten klar zuordnen. Anhand des Kriteriums Dauerhaftigkeit von Information und der Unterscheidung, ob es sich um deklaratives, d.h. beschreibendes, oder prozedurales, d.h. den Ablauf betreffendes, Wissen handelt, gelingt die Zuordnung und wirft keine offene Fragen auf. Von Interesse ist jedoch ein Blick auf die Wissensarten für den Fall, dass die Einschränkung auf nur eine Abstraktionsebene fallengelassen wird. Dabei würde die Bedeutung des *mittelfristigen deklarativen Wissens* stark ansteigen, denn gerade über diese Wissensart werden die

Fusionszwischenergebnisse festgehalten und über mehrere (*JDL) Processing Level* für Bewertungs- und Schlussfolgerungsprozesse verfügbar gemacht. Auch wäre das *langfristige explizite prozedurale Wissen* stark betroffen, da hierbei ein mehrere Abstraktionsebenen übergreifendes prozedurales Wissen zum Einsatz kommen müsste.

Dieses *langfristige explizite prozedurale Wissen* steht im Mittelpunkt bei dem zweiten Schritt der „Bestimmung des Problemlösungsparadigmas". Da die ermittelte Fusionsklasse 15 über verschiedene kanonische Problemlösungsparadigmen realisiert werden kann, wurde schrittweise in einem Eingrenzungs- bzw. Ausschlussverfahren die *generation-based* Problemlösungsform „*canonical form IX*" und ihre typische Umsetzung in Form eines Produktionssystems mit vorwärtsverkettetem Inferenzmechanismus bestimmt. Einige Punkte in dem Auswahlprozess sollen im Folgenden näher beleuchtet werden.

Die Aufteilung von prozeduralem Wissen in einen Teil langfristig deklaratives Wissen und einen Teil Kontrollwissen (*control knowledge*) wurde bereits im Stand der Technik und im Ergebnisteil festgehalten. Über die Unterscheidung zwischen spezifischem oder generellem langfristigen deklarativen Wissen wurde die niedrigere Hälfte der kanonischen Problemlösungsformen (I-VIII) ausgeschlossen. Dabei unterscheidet sich die „*canonical form I*" von der „*canonical form IX*" in den Hauptkriterien nur durch die Nutzung von spezifischen und nicht generellen deklarativen Wissen. Ein typischer Vertreter einer „*canonical form I*" ist ein KNN, wie es WEIGERT (2006) [Wei06] in seinem Wissensentdeckungsprozess zur Ableitung der Regeln zur kleinräumigen Stickstoffdüngung angewandt hat. Nach ANTONY (2001) [Ant01][49] eignen sich KNN mit ihrer spezifischen Gewichtsmatrix und Architekturfestlegung (z.B. Knoten, Ebenen) sowie als nicht modellbasierter Ansatz wenig für breitere, d.h. generelle, Aufgabenstellungen. Auch sei die Kontextsensitivität nur sehr eingeschränkt gegeben. Mit Blick auf den *Knowledge-base Level* der Simulation (vgl. Kap. 5.2.2.4) stellt sich die Frage, worin ein Vorteil in der Nutzung eines Produktionssystems liegt, wenn es aufgrund des eingesetzten Wissens des Wissensmoduls „*CROP_PRODUCTION*" Schwächen bei der Kontextsensitivität hat und das eigene generelle deklarative Wissen, d.h. die „WENN/DANN" Regeln, nichts anderes ist als (transformiertes) spezifisches Wissen. Dem können folgende Hinweise entgegengehalten werden:

Bei der Wissensentdeckung wurde ein dreistufiges Verfahren angewendet, bis die Entscheidungsregeln als Endergebnis vorlagen.

Der erste Schritt nutzt ein KNN für eine teilflächenspezifische Ertragsprognose für variable

[49] A. a. O., S. 19

Stickstoff-Mengen.

Im zweiten Schritt wird darauf aufbauend ein Prognosemodell angewandt, um für eine bestimmte teilflächenspezifische Situation die ökonomisch optimale Stickstoff-Applikation zu bestimmen.

Und im dritten Schritt werden in einem Entscheidungsbaumverfahren die Düngeregeln abgeleitet.

Während der erste Schritt durch das KNN dominiert wird, wird beim zweiten Schritt eine Optimierungsaufgabe hinzugefügt und der dritte Schritt ermöglicht im Extremfall die Loslösung von rein spezifischem Wissen. WEIGERT (2006) [Wei06][50] hat bereits darauf aufmerksam gemacht, dass die Entscheidungsregeln gegenüber der reinen Wissensdarstellung durch das KNN der Ertragsprognose und dem ökonomischen Optimierungsprozess die Möglichkeit bieten, dass Experten diese Regeln verstehen und sie gegebenenfalls auch anpassen und erweitern können. Demzufolge können die Düngeregeln letztlich komplett unabhängig von den Ergebnissen der (automatischen) Wissensentdeckung werden und von einem Experten sowohl auf einen (spezifischen) Einzelfall als auch auf ein sehr generelles Düngungsregime ausrichtet werden. Eine weitere Möglichkeit ist auch die Kombination beider Fälle, die ein breites Anwendungsfeld mit einer Art Default-Lösung und spezifische Aufgaben individuell bedienen kann. Im Unterschied zu einem KNN hat ein Produktionssystem mit vorwärtsverkettetem Inferenzmechanismus mittels seines modellbasierten Ansatzes keine Probleme dies zu realisieren. Das Kernstück des modellbasierten Ansatzes, die Inferenzmaschine, die den „*Erkennen-Handeln-Zyklus“* (vgl. Kap. 5.1.1) umsetzt, ist unabhängig vom eingesetzten generellen deklarativen Wissen, den Entscheidungsregeln. Knapp zusammengefasst, wird dabei generelles deklaratives Wissen („WENN-Teil“) gegen spezifisches deklaratives Wissen[51] (Faktenbasis) mithilfe eines rigiden, *single level of abstraction*, *generation-based* Kontrollwissens auf Übereinstimmung getestet.

Ein weiterer Vorteil einer „*canonical form IX*“-Problemlösungsform gegenüber einer „*canonical form I*“-Problemlösungsform ist die wesentlich einfachere Erweiterbarkeit bzw. Wartung des deklarativen Wissens. Bei einem KNN ist ein neuer Lernvorgang nötig, bei einem Produktionssystem werden einfach Regeln modifiziert, hinzugefügt oder entfernt.

In der Simulation sind alle anderen Wissensmodule frei von dem Verdacht über die

[50] A. a. O., S. 104

[51] An dieser Stelle ist das kurzfristige, mittelfristige oder langfristige deklarative Wissen und nicht der deklarative Teil des langfristigen prozeduralen Wissens gemeint.

Anwendung einer niedrigeren kanonischen Problemlösungsform erzeugt worden zu sein. Ihre Komplexität ist geringer als in dem „*CROP_PRODUCTION*“-Wissensmodul, aber dafür zum Teil sehr generell formuliert, wie z.B. die Ermittlung von Minimal- und Maximalwerten oder das Zusammenfügen von Textstücken. Abschließend ist festzuhalten, dass obige Diskussion bereits sehr durch das zweite Teilziel, die konkrete Simulationsimplementierung geleitet war. Wesentlich einfacher ist die Antwort auf die Fragestellung, wenn der Fall nur auf der abstrakten Prozessmodellebene hinterfragt wird. Dort wird hinsichtlich des *langfristigen expliziten prozeduralen Wissens* abstrakt von einer Beziehung bzw. Prozedur zur pflanzenbaulichen und agrarsystemtechnischen Situationsbewertung gesprochen. Die Art und Weise zur Ermittlung des heuristischen Wissens und dessen Charakter ist nicht vorgegeben. Für Expertensysteme wurden die grundsätzlichen Wege der Wissensakquisition in Kapitel 5.1.2 zusammengestellt. Das Grundprinzip ist jedoch auf eine Vielzahl der wissensbasierten Systeme für das Ingenieurwesen übertragbar.

Die Auswahl einer Problemlösungsform aus den höheren Klassen, d.h. „*canonical form X-XVI“*, erfolgte über die Charakterisierung des Kontrollwissens nach rigide oder flexibel und *single level of abstraction* gegenüber *multiple levels of abstraction*. Die Entscheidung für eine rigide Form wurde im Ergebnisteil über die Kontextsensitivität begründet, die durch den deklarativen Wissensanteil bereits gegeben ist. Ein rigides und damit nicht kontextsensitives Kontrollwissen bedeutet, dass der Fusionsalgorithmus nicht dynamisch in seinem Wirkmechanismus bzw. Inferenz-Prozess geändert wird. Angewandt auf die ausgewählte Problemlösungsform, wird starr mit einem Produktionssystem geschlussfolgert und nicht, um im Expertensystemkontext zu bleiben, variabel auf die Methode des Fallbasierten Schließens oder des Schließens nach „Fuzzy Theorie“ oder der Hierarchischen Problemzerlegung gewechselt.

Damit stellt sich noch die Frage, wie ein Vertreter einer Problemlösungsform aussieht, der in der Lage ist, über mehrere Abstraktionsebenen *Data Fusion* zu betreiben. Von der „*canonical form IX*“-Problemlösungsform unterscheidet sich die Form „*canonical form XI*“ einzig in der Eignung für *multiple levels of abstraction*. Ein derartiges Problemlösungsparadigma muss in der Lage sein, über mehrere Abstraktionsebenen, d.h. *JDL Processing Level*, hinweg sowohl in der Richtung „*top down*“ als auch „*bottom up*” schlussfolgern zu können. Dieses Paradigma kann durch ein Produktionssystem mit einem hierarchisch aufgebauten Regelwerk, d.h. dem langfristigen deklarativen Anteil des prozeduralen Wissens, emuliert werden. Ein Blick auf die Simulation wirft die Frage auf, ob die getroffene Wissensaufteilung nicht bereits ein hierarchisches Regelwerk darstellt. Grundsätzlich ist festzuhalten, dass das Regelwerk nur

auf *JDL Level 2 Processing* Ebene definiert ist und daher selbst als hierarchisches Regelwerk keine Fähigkeit zu *multiple levels of abstraction* besitzen würde. Das Regelwerk der Simulation weist eine Aufteilung des Wissens in vier Pfaden von Wissensmodulen auf, die weitestgehend gleichwertig nebeneinanderstehen. Die SUM-UP-Module sind dem jeweiligen Hauptmodul nur nachgeschaltet und fassen die Zwischenergebnisse bzw. Ergebnisse auf gleicher Abstraktionsebene zusammen. Ein hierarchisches Regelwerk wäre hingegen beispielsweise in der Lage, den im Ergebnisteil angesprochenen Aspekt des Bodenbedeckungsgrades für die Pflanzenzustandsbewertung in das *JDL Level 1 Processing* einfließen zu lassen. Die Kenntnis des Bodenbedeckungsgrades könnte ein Wissensmodul mit „*top down*"-Sicht aufgrund von Erfahrungswerten für das aktuelle Wachstumsstadium und einer aus Stichproben erstellten Boniturkarte des Feldes auf *JDL Level 2 Processing* ableiten. Diese Schlussfolgerung könnte es mit einem hierarchisch niedriger stehenden Wissensmodul mit „*bottom up*"-Sichtweise teilen, so dass dieses eine robustere Vegetationsindexbestimmung auf *JDL Level 1 Processing*-Ebene durchführen kann. Dieser Vegetationsindex würde wiederum mit einem höher stehenden pflanzenbaulich ausgerichteten Wissensmodul auf *JDL Level 2 Processing*-Ebene geteilt und dort zur Ableitung eines Stickstoffapplikationswertes genutzt. Für den Datenaustausch über hierarchische Grenzen hinweg bietet sich z.B. ein *Blackboard System* an [Sri97].

So steht zum Abschluss des Kapitels noch ein Blick auf die Einstufung als *generation based,* d.h. datengetriebene, Problemlösungsform aus. Während auf Prozessmodell-Ebene die Begründung aus der funktionalen Modellierung abgeleitet wurde, zeigt der Simulationsteil dieser Arbeit dies für den Anwendungsfall sehr anschaulich (vgl. Kap. 5.2.2.3) und wird daher an dieser Stelle nicht mehr näher erläutert. Die hypothesengetriebene (*hypothesis-based)* Alternative zur ausgewählten Problemlösungsform ist die „*canonical form X*". Auch sie kann über ein Produktionssystem realisiert werden, jedoch mit einem rückwärtsverkettetem (*backward chaining*) Inferenzmechanismus (vgl. Kap. 5.1). Diese Form ist typisch für medizinische Diagnosesysteme oder z.B. für die Identifikation von Pflanzenkrankheiten. Dabei wird ausgehend von einer Hypothese erforscht, ob die Fakten in der Wissensbasis diese Hypothese bestätigen oder widerlegen. Aus Sicht des Produktionssystems heißt dies, dass zum einen nach einer Übereinstimmung zwischen dem Ziel und dem Aktionsteil der Produktionsregeln gesucht wird. Zum anderen wird der Bedingungsteil dann als (neues) Teilziel eingesetzt, dessen „Richtigkeit" es zu beweisen gilt.

6.2.3 Systemarchitektur

Im Ergebnisteil zur Systemarchitektur wurde für den Sensor-Ansatz mit Kartenüberlagerung gezeigt, wie die Ergebnisse und Erkenntnisse aus dem Funktionalen Modell und dem Prozessmodell auf eine mit einem landwirtschaftlichen BUS-System im Einklang stehende Systemarchitektur abgebildet werden können. Zugleich wurden in diesem Zusammenhang die Voraussetzungen und Auswirkungen auf den De-facto-Standard für landwirtschaftliche BUS-Systeme, die ISO 11783, hinsichtlich einer auf MSDF basierenden Real-time Prozessführung analysiert. Das angewandte Abbildungs- und Analyseverfahren wurde wiederum in aufeinander aufbauende Schritte gegliedert. Somit können die einzelnen Annahmen, Zwischenergebnisse, Forderungen sowie Endergebnisse einfach nachvollzogen werden. Wiederum wurde mit wenig formalen Modellierungswerkzeugen gearbeitet, sondern die Gedankengänge in Textform, unterstützt mit wenigen erläuternden Abbildungen, dokumentiert. Bei der Analyse der ISO 11783 wurde jedoch an manchen Stellen sowohl eine tiefere Kenntnis der Norm vorausgesetzt als auch davon ausgegangen, dass die Norm zur Einsichtnahme vorliegt. Die nachfolgende Diskussion folgt der im Ergebnisteil angewandten Gliederung in die Abschnitte Bestimmung der allgemeinen Systemarchitektur, Zuordnung der Prozessmodellergebnisse in den ISOBUS, der „*In-field Controller"*, die Integration der Online-Sensorik und der konkurrierende Zugriff auf Geräteressourcen.

Der Ausgangspunkt für die Wahl der allgemeinen Systemarchitektur ist offensichtlich. Derart ungleiche Datenquellen wie im Fall des Sensor-Ansatzes mit Kartenüberlagerung sind *noncommensurate* Eingangsgrößen, die nicht auf Rohdatenebene fusioniert werden können. Das Hauptaugenmerk lag daher auf einer MSDF auf Merkmals/Zustands- oder Entscheidungsebene.

Die grundsätzliche Eignung des rein zentralen Ansatzes („*Central Fusion Processing*") wurde im Ergebnisteil festgehalten und deckt sich mit der Einschätzung von RAMON ET AL. (2006) [RMBVD06], die MSDF für die Pflanzenschutzapplikation vorschlagen: „ …. *The optimal architecture for disease and stress detection would be one that results in a minimum information loss and is fast enough to be implemented in real time. Centralized fusion with feature vectors seems the most appropriate because it satisfies both requirements of accuracy and speed. …*"[52] Dass jedoch der rein zentrale Ansatz dem verteilten Charakter eines Landwirtschaftlichen BUS-Systems widerspricht, wurde im Ergebnisteil bereits ausführlich dargelegt und führte zur Auswahl des Systemarchitekturtyps „*Distributed Sensor/Fusion*".

[52] A. a. O., S. 283

Aufgrund der durchgängig möglichen Abbildung des Anwendungsfalles auf die ISOBUS-Struktur wurde das andere Extrem zum zentralen Ansatz, eine komplett verteilte Systemarchitektur, d.h. der „*Fully integrated*“ Typ, nicht mehr in Erwägung gezogen, da er im Fall gleicher Eignung zu einer höheren Komplexität führt. Diese Variante ist das Mittel der Wahl, wenn ein stark verteiltes System, vor allem auch räumlich verteilte Systemkomponenten, zum Einsatz kommen sollen. Auch spielt dabei eine Rolle, dass auf verschiedene Kommunikationsnetzwerke zugegriffen werden soll und eine verbindungsorientierte Struktur über *Switch*-Netzwerke genützt wird. Bei einem Landwirtschaftlichen BUS-System, das in der Grundkonzeption ein Kommunikationssystem auf Multimaster-Basis ist und ein lokal begrenztes System darstellt, ist dieser Ansatz nicht unmittelbar gefordert. Dem könnte entgegengehalten werden, dass nach ISO 11783 auch Unternetzwerke möglich sind, z.B. ein Unternetzwerk einer Anhängerspritze für die Anbindung einzelner Sensoren an jeder Spritzdüse. Dies entspricht jedoch noch nicht einer vollkommen verteilten, evtl. auch redundanten Architektur mit dynamischer Lastverteilung zwischen mehreren Fusionsprozessoren und dem Schalten zwischen einzelnen Sensoren. Sollte in Zukunft ein Landwirtschaftliches BUS-System auf einem Real-time Ethernet Protokoll aufgebaut werden, so würde durch die grundlegende Protokolldefinition der „*Fully integrated*“-Ansatz selbst bei lokaler Ausprägung des Systems wesentlich einfacher umgesetzt werden können. Bei dem aktuell anerkannten ISOBUS könnte ein erster Anknüpfungspunkt für den Typ „*Fully integrated*“ in der Anbindung der WSN als Online-Sensorik gesehen werden. Eindeutig in den Mittelpunkt des Interesses würde diese Architekturform jedoch erst treten, falls die Prozessführungslösung ihren räumlich begrenzen Charakter verliert und über drahtlose Kommunikationsverbindungen nicht nur einzelne Inputquellen (z.B. WSN, Auftragsdateien) mit integriert werden, sondern auch *Data Fusion*-Vorgänge auf räumlich verteilte Stellen verlagert werden. Das Stichwort hierzu wäre eine serviceorientierte Architektur mit Web-Services, die die MSDF oder Teile davon erbringen. Dabei würde das für die Prozessführung notwendige prozedurale Wissen bzw. Teile davon providergestützt gehalten. Auch Teile des langfristig deklarativen Wissens könnten dort oder an wiederum anderen Stellen gehalten werden. An der Erforschung und dem Aufbau eines nötigen öffentlichen-privaten Wissensmanagements wird beispielsweise im iGreen-Forschungsprojekt gearbeitet [BBMK11]. Der rasante technologische Fortschritt im IT-Bereich lässt erwarten, dass diese Lösungsansätze an Bedeutung gewinnen und dann für einen Systemarchitekturtyp „*Fully integrated*“ sprechen würden. Für diese Arbeit jedoch wurde im Funktionalen Modell und dem Prozessmodell eine den Anforderungen entsprechende Lösung mit lokalem

Charakter abgeleitet. Demnach stellt die Lösung in der Mitte der Bandreite die vorzügliche Variante dar. Auch die vorgeschlagene Systemarchitektur gewährleistet Skalierbarkeit und damit Potential für eine zukünftige Erweiterbarkeit.

Ein kurzer Blick auf die am Markt befindlichen Online-Sensoren [Rec10] soll die allgemeine Architekturdiskussion abschließen. Das derzeit geltende Verständnis über Online-Sensoren ist durch die N-Online-Sensoren geprägt und geht davon aus, dass diese Online-Sensoren aus dem reinen Sensorteil und einem N-Sollwert ableitenden Teil bestehen. Es existieren zwei Ausführungsvarianten, die Lösung als integrierter „*Smart Sensor*“ oder als Kombination aus einem eigenen Agrarcomputer oder robusten Notebook für den Außeneinsatz mit angeschlossenen Sensorköpfen. Aus Sicht der MSDF findet die Fusion auf der Rohwert-Ebene der Sensorköpfe und in manchen Fällen auch noch auf Ebene der N-Applikationsraten-Bestimmung mit „Hinterlegung von Faktorkarten“ [LSL02] statt. Einige der N-Online-Sensoren besitzen eine ISOBUS-Implementierung für das *Virtual Terminal* und können über einen proprietären Eingang des *Task Controllers* (z.B. serielle Schnittstelle nach RS 232) den N-Applikationswert einspeisen, der dann vom Task Controller an die Düngerstreuer-ECU via ISOBUS gesendet wird. Oftmals wird aber dieser Umweg nicht gegangen, sondern eine bidirektionale Kommunikationsverbindung mit dem mobilen Agrarcomputer genutzt, der den Düngerstreuer direkt ansteuert.

Diese Arbeit verfolgt ein anderes Verständnis von Online-Sensoren. Es wird nicht nur von N-Online-Sensoren ausgegangen, sondern allgemein von Sensortechnik, die bei der Überfahrt während der Applikationstätigkeit die Messgrößen erfassen und auf *JDL Level 0 und 1 Processing*-Ebene verarbeiten kann. Die Ergebnisse werden über Knotenprozessoren weiteren Systemteilnehmern zur Verfügung gestellt, die auch zu einem *JDL Level 2 Processing* in der Lage sind, um z.B. einen Stickstoff-Applikationswert abzuleiten. Diese Arbeit legt vor allem Wert auf eine Unterteilung in logische Funktionen und erst im zweiten Schritt spielt die tatsächliche Umsetzung in Hard- und Software eine Rolle. Wird diese Überlegung zu Grunde gelegt, so sind die existierenden N-Online-Sensoren eine Kombination aus einer Art „*In-field Controller”* und einem Online-Sensor. Die Systemarchitekturen am Markt verfolgen daher in ihrer aktuellen Ausprägung entweder den Weg der „*Central Fusion Processing*“- oder der „*Distributed Sensor/Fusion*“-Systemarchitektur in proprietärer Ausprägung.

Mit dem Erweiterungsvorschlag um einen *„In-field Controller“* konnte die Zuordnung aller im Prozessmodell festgestellten Wissensarten in ein ISOBUS-System durchgeführt werden. Damit ließ sich auch ein Systemdiagramm (vgl. Abb. 4.7) aufstellen, mit dem die Skalierbarkeit hinsichtlich aller drei Systemansätze zur Teilflächenbewirtschaftung

anschaulich gezeigt werden konnte. Ausgehend von der allgemeinen Systemarchitektur und dem *„In-field Controller“*-Vorschlag war eine Analyse der bisherigen Normdefinition möglich. Grundsätzlich konnte dabei festgestellt werden, dass für den Sensor-Ansatz mit Kartenüberlagerung bereits eine gute Grundlage gegeben ist und Normerweiterungen und Vorschläge jüngeren Datums in die benötigte Richtung gingen bzw. gehen. Dennoch wurden auch noch Lücken festgestellt und Verbesserungswünsche für die zukünftige Weiterentwicklung von landwirtschaftlichen BUS-Systemen identifiziert.

Bereits im Stand der Technik wurde darauf hingewiesen, dass sowohl ISOBUS als auch LBS die Teilflächenbewirtschaftung ursprünglich nur im Kontext des Kartierungsansatzes definierten und die beiden anderen Ansätze außer Acht gelassen haben. Dies wurde von OSTERMEIER ET AL. (2003) [OAD03] bereits vor mehreren Jahren erkannt und von ihnen eine Systemarchitektur mit *„In-field Controller“,* integrierter Online-Sensorik und einem Prozessdaten-Handling gefordert, das konkurrierenden Zugriff auf Geräteressourcen zulässt. Mit der Einführung der Geräteklasse (*device class*) „*Sensor systems*“ in 2007 ([ISO11783-1]) wurde die Grundlage für eine Forderung bereits erfüllt, wenngleich dabei die Systemansätze für Teilflächenbewirtschaftung nicht im Mittelpunkt standen. Mit der wachsenden Anzahl am Markt angebotener N-Online-Sensoren, die nicht vollständig in den ISOBUS integrierbar sind, wurde in 2010 dem Sensor-Ansatz auf Ebene der ISOBUS-Normung Aufmerksamkeit zuteil. Der neue Vorschlag [ISO11783-10DAM1]-Kapitel 6.6.3 „*Real time sensor based control*“ kann als eine durchaus gangbare Reaktion auf einen drängenden Bedarf bzw. Problemstellung der Praxis beim ISOBUS gesehen werden. Nicht explizit hervorgehoben wurde der Sensor-Ansatz mit Kartenüberlagerung, aber mit der Möglichkeit zur Überlagerung mit (Applikations-)Karten auf Entscheidungsebene wurde er in eingeschränkter Form mitberücksichtigt. Damit bietet dieser Normerweiterungsvorschlag eine praxisorientierte, bis zu einem gewissen Grad skalierbare Lösung an, um alle drei Systemansätze zur Real-time Prozessführung normgerecht realisieren zu können.

Der Vorschlag dieser Arbeit beschränkt sich jedoch nicht auf eine reine *Data Fusion* auf Entscheidungsebene, sondern geht einen Schritt weiter und diskutiert und umfasst Entwurfs- und Analysemethoden für eine MSDF-Lösung auf allen drei möglichen Ebenen, der Rohdaten-, der Feature/Merkmals- und der Entscheidungsebene. Daher ist er vom Ansatz her mächtiger und zeigt die Möglichkeiten und Herausforderungen für eine zukünftige Erweiterung des ISOBUS oder für die Konzeption eines zukünftigen landwirtschaftlichen BUS-Systems auf. Weiterhin weist er Herstellern den Weg für proprietäre Lösungsansätze oder hilft bei der Auslegung der N-Online-Sensorik nach derzeitigem Marktverständnis.

In den einleitenden Worten des Ergebnisteil zum zentralen *Data Fusion*-Prozessor „*In-field Controller”* wurde bereits darauf hingewiesen, dass idealerweise der bereits definierte *Task Controller* um diese Funktionalität erweitert werden sollte. Da es bisher keinerlei Normungsaktivitäten in dieser Hinsicht gibt, wurde für die Normanalyse ein eigenständiger „*In-field Controller”*-Teilnehmer angenommen. Diese Hilfsmaßnahme zeigt zugleich einen Alternativweg auf. Der *Task Controller* könnte mit dem als „*Function*“ definierten „*In-field Controller”* ein „*Working Set*“ mit dem *Task Controller* als „*Master*“ bilden. Dies würde auch die Frage beantworten, wer in dem Scenario letztendlich die Dokumentation durchführt. Diese Funktionalität ist als eine der Hauptaufgaben bereits für den *Task Controller* definiert. Für den Fall, dass die Dokumentation nicht an den *Task Controller* delegiert werden kann, könnte dessen Definition für das Datenlogging (*"DataLogValue"*) für den „*In-field Controller”* übernommen werden.

Aus der Sichtweise des zentralen *Data Fusion*-Prozessor „*In-field Controller”* weist die Norm Lücken beim Datenaustausch zwischen FMIS und MICS für *spezifisch langfristiges deklaratives Wissen* und *langfristiges explizites prozedurales Wissen* auf. Die Änderungswünsche wurden im Ergebnisteil entsprechend abgeleitet. Dennoch bleiben noch einige Punkte, die zu einer näheren Betrachtung herausfordern. Gerade für das vorgeschlagene XML-Element „Überlagerungskarte“ („*Overlay-Map (OMP)*") ist es diskussionswürdig, ob jede „Überlagerungskarte“ wirklich an einen Auftrag („*Task (TSK)*“) gebunden werden soll. Die Karten sind im Grunde eine Informationslage, die nicht nur starr einem Auftrag zugeordnet sein sollten, sondern mehrfach angewendet werden könnten. In diesem Fall wäre eine n-m Relation zwischen Aufträgen und „Überlagerungskarten“ einer 1-n Relation vorzuziehen. Diese Diskussion zeigt das Spannungsfeld auf, das dieser neue umfassende Ansatz mit sich bringt. Im Grunde wird Funktionalität, die bisher eindeutig dem FMIS (z.B. Erstellung von Applikationskarten, Datenhaltung der *Precision Farming*-Karten) zugeordnet war, teilweise auf das MICS verlagert. Demzufolge würde sich als Zielsetzung einer weiterführenden Arbeit auch ein Blick auf die Standardisierungsbemühungen zum Datenaustausch zwischen FMIS lohnen.

Der Austausch von *langfristigen expliziten prozeduralen Wissen* zwischen dem FMIS und dem „*In-field Controller”* mithilfe der W3C-Empfehlung „*RIF Production Rule Dialect*“ unterstützt das abgeleitete Problemlösungsparadigma vollständig. Soll jedoch eine größere Bandbreite an möglichen prozeduralen Wissensarten abgedeckt werden, empfiehlt sich als erster Anknüpfungspunkt eine Berücksichtigung der Arbeiten und der Standards zum Datenaustausch im Semantischen Web mit seiner *Web Ontology Language* (OWL) [W3C09].

Dass Festlegungen zum Datenaustausch für das *mittelfristige deklarative Wissen* nur für eine Nachweispflicht oder bei einem Analysebedarf erforderlich sind, wurde im Ergebnisteil umfassend erörtert. Solange kein grundlegender Wechsel der Systemarchitektur vorgenommen wird, bleibt diese Wissensart auf den „*In-field Controller"* beschränkt. Für die Diskussion ist dies jedoch ein geeigneter Punkt, um eine bei der Diskussion des Prozessmodells analysierte alternative Problemlösungsform nochmals von einer anderen Perspektive aufzugreifen. Die bisher vorgeschlagene Systemarchitektur setzt klar die Vorgaben für das funktionale und das prozedurale Modell um, bei der die interessante MSDF auf die *JDL Level 2 Processing*-Ebene eingeschränkt wurde. Somit ist auch der eine zentrale *Data Fusion*-Prozessor ausreichend und die unterstützende Datenbankfunktionalität (nach JDL Modell) lässt sich ebenfalls dort lokal realisieren. Soll jedoch eine *Data Fusion* über mehrere Abstraktionsebenen *(multiple level of abstraction)* auch unter Einbeziehung von Knotenprozessoren stattfinden, so ist ein gemeinsames Datenhaltungssystem gefordert. Für diese Zwecke bietet es sich an, den bereits genormten *File Server*-Dienst ([ISO11783-13]) als „*Support Database*" zu nutzen und auch einen Teil der „*Fusion Database*" dort zu realisieren. Die Herausforderung dieser Lösung werden die geringe Bandbreite und die Latenzzeiten des ISOBUS sowie die Verzögerungen beim Schreib- und Lesezugriff auf den mit anderen ISOBUS-Teilnehmern geteilten *File Server*-Dienst darstellen. Zugleich ist der skizzierte Vorschlag auch ein Lösungsweg wie ein gemeinsames *Task Controller* -„*In-field Controller"* „*Working Set*" interagieren könnte. Im ursprünglichen Systemdiagramm (vgl. Abb. 4.7) wurde die Funktion des *File Server* -Dienstes vorausschauend mit aufgenommen.

Anhand des Systemdiagramms lässt sich auch für den „*In-field Controller"* in abstrakter Form untersuchen, welche Auswirkungen auf ihn die am Anfang des Kapitels ins Spiel gebrachte service-orientierte Architektur hätte, ohne dabei aber komplett auf eine „*Fully integrated*"-Systemarchitektur zu wechseln. Ginge es rein um ein Update seines langfristigen prozeduralen Wissens über einen „Wissensprovider", so unterstützt das derzeitige Schema dies bereits vollständig**.** Über das FMIS wird das Wissensmodul über das Internet beim „Wissensprovider" beschafft und gemäß dem „*RIF Production Rule Dialect*"-Datenformat vom FMIS ins MICS übertragen. Anders stellt sich die Situation dar, wenn durch einen „Web-Service" die „*In-field Controller"*-Funktionalität oder Teile davon erbracht werden. In diesem Fall würde der „*In-field Controller"* in zwei Teile aufgeteilt. Ein Teil wäre das „*In-field Controller Gateway",* mit der Anbindung an den ISOBUS mit der Online-Sensorik und der Applikationstechnik, und der zweite Teil wäre der „*In-field Controller Web-Service*" mit einer drahtlosen Kommunikationsverbindung zum ersten Teil sowie mit einem Zugang zum

FMIS wegen des langfristig deklarativen Wissens. Eine Schwachstelle eines derartigen Systems ist die Qualität der drahtlosen Datenverbindung zwischen den beiden Teilen, die eine Echtzeitfähigkeit der Prozessführung garantieren muss. Diese teilweise Abkehr von der „*Distributed Sensor/Fusion*"-Systemarchitektur deckt die zentrale Herausforderung für einen „*Fully integrated*"-Ansatz auf, der in hohem Maße auf drahtlose Kommunikationsverbindungen im ländlichen Raum angewiesen ist.

An dieser Stelle wird die kritische Betrachtung zum „*In-field Controller"* beendet und der Fokus auf die Integration der Online-Sensorik gerichtet. Der entscheidende Schritt war 2007 die Einführung der neuen Geräteklasse (*device class*) „*Sensor systems*" in den ISOBUS, der das im Ergebnisteil beschriebene Dilemma gelöst und den Weg zur normkonformen Definition eigenständiger Sensorsysteme geebnet hat. Bedingt durch den Analysecharakter im Ergebnisteil ähnelt der erste Teil des Kapitels (vgl. Kap. 4.4.3) bereits einer Diskussion und Einordnung. Dabei wurden die grundsätzlichen Alternativen zum Datenaustausch im MICS aufgezeigt. Zum einen ist dies die Nutzung der Prozessdaten-Nachrichten (*process data message; PGN: 51968)*, zum anderen die Option einer Neudefinition von PGN's nach dem Vorbild der Navigationsdaten. Die Option mit den Prozessdaten-Nachrichten unterstützt eine Systemlösung mit MSDF auf Merkmals/Zustandsebene sowie auf Entscheidungsebene, da sich hier die (Zwischen-)Ergebnisse der *Data Fusion* in der Regel auf eine CAN-Nachricht zusammenfassen lassen. Wird hierfür jedoch eine sehr hohe Updaterate benötigt oder sollen doch größere Datenmengen ausgetauscht werden, so wird eine PGN-Neudefinition mit Nutzung eines der ISOBUS Transport Protokolle von Vorteil sein. Diese Form könnte bei Betrachtung des Einzelfalls auch für eine eingeschränkte *Data Fusion* auf Rohdatenebene zur Anwendung kommen. Spätestens bei bildgebenden Sensoren wird die Übertragungskapazität des ISOBUS erreicht und erlaubt nur eine MSDF auf Merkmals/Zustandsebene sowie auf Entscheidungsebene. Erst der Wechsel auf ein neues Kommunikationssystem wie z.B. Real-time Ethernet ermöglicht die ganze Breite an *Data Fusion*-Ansätzen.

Auch wurde herausgestellt, dass die Option der Prozessdaten-Nachrichten nicht nur den Datenaustausch im MICS sondern auch zwischen MICS und FMIS sehr gut unterstützt, d.h. nicht nur der Austausch der Werte ist spezifiziert, auch eine maschinenlesbare Gerätebeschreibung ist möglich. Daher lag es auf der Hand, das Zusammenspiel zwischen Online-Sensoren und dem „*In-field Controller"* als genaue Kopie des Schemas zwischen *Task Controller* und Prozessdaten-Nachrichten zu definieren bzw. zu fordern.

Ausgehend von der abgeleiteten Systemarchitektur wurde eine weitere Forderung aufgestellt. Zwei grundsätzlich unterschiedliche Online-Sensor-Klassen sollten unterscheidbar sein, der

Knotenprozessor „*internal sensor system*“ und der Knotenprozessor „*external distributed sensor system*“. Dies ist zweifelsfrei ein neuartiger Wunsch, der bisher so noch nicht veröffentlicht wurde. In einem nächsten Schritt wurde eine Art rezeptartiges Vorgehen vorgestellt, wie Online-Sensoren als Knotenprozessoren mithilfe einer ISOBUS-Gerätebeschreibung dargestellt werden können. Angewandt auf einige im Funktionalen Modell angeführte Objekte wurden die generischen Online-Sensor-Typen Pflanze, Boden und Wetter definiert und dabei offengelegt, wie fehlende Datenbankelemente (*data dictionary entity*) identifiziert und mit einem eigenen Vorschlag ergänzt werden können.

Im Hinblick auf den Sensor-Ansatz mit Kartenüberlagerung steht der Online-Sensor „Pflanze On-board“ an erster Stelle. Bereits im Ergebnisteil wurde der Wunsch nach Erweiterung um Vegetationsindizes geäußert, für die bereits eine Korrelation zu Pflanzeneigenschaften, wie z.B. N-Status, Biomasse, N-Aufnahme, nachgewiesen wurde. Einer Übersicht nach RECKLEBEN (2010) [Rec10] zufolge kommt der bisher einzig definierte Vegetationsindex NDVI nur bei einem Teil der aktuellen N-Online-Sensoren zum Einsatz (Crop Circle (Holland Scientific), Greenseeker (Trimble), ISARIA (Fritzmeier Umwelttechnik)). Im Stand der Technik (vgl. Kap. 2.3.2.2) wurde dargelegt, dass ein fahrzeuggetragener Online-Sensor „Boden“ derzeit eher Gegenstand von Forschungsvorhaben und nicht ein marktgängiges Produkt ist. Die Ausnahmen Zugkraftsensor, EM38- und „Veris 3100“-System werden typischerweise nur zur Gewinnung von Kartierungen genutzt. Trotz des Mangels an verfügbaren Online-Sensoren „Boden“ lassen sich aus systemtechnischer Überlegung die nötigen Strukturen bereits vorab definieren und die daraus resultierenden DDE‘s dafür beantragen. Gerade die Bodenfeuchtigkeit und Nährstoffgehalte werden von besonderem Interesse sein. Erfolgsversprechender im Hinblick auf eine kurzfristige Verfügbarkeit für die Praxis erscheinen die Ansätze über WSN, diese Messwerte einer Real-time Prozessführung bereit zu stellen, d.h. über die Integration eines Knotenprozessor „*external distributed sensor system*”. Diese Arbeit schlägt keine konkrete Spezifikation für die Gateway-Funktion dieses Knotenprozessortyps vor. Sie regt jedoch dazu an, bei den neu gestarteten Normierungsbestrebungen zu NWI ISO 16867; „*Wireless Communication in Agriculture*; *Standardization of wireless communication for Fleet Management, including implement synchronization“* [NWIISO16867] (vgl. Kap. 2.3.2.2) diese Gateway-Funktion nicht außer Acht zu lassen. Wichtig ist, dass bei der anstehenden Normung eine Brücke zwischen ISOBUS und der neuen ISO 16867 geschlagen wird. In [ISO11783-1] - Annex F „(normative) ISO 11783 All industry NAMEs” liegen bereits Definitionen von *Functions* vor, die die Gateway-Funktionalität eines Knotenprozessors “*external distributed sensor system*”

unterstützen könnten:

- Value: 29 – Function: *Off-Vehicle Gateway,*
- Value: 54 - Function: *Communications Cellular,*
- Value: 55 – Function: *Communications Satellite,*
- Value: 56 – Function: *Communications, Radio.*

Vorausgesetzt ein WSN ist in der Lage seine *„JDL Level 0 und/oder 1 Processing"*-Ergebnisse transparent im MICS darzustellen, blieb im Ergebnisteil dennoch die Frage unbeantwortet, wie die lokale Zuordnung des kurzfristigen deklarativen Wissens stattfinden soll. Denkbar sind zwei grundsätzliche Wege. Der Knotenprozessor sendet einen Messwertabfragewunsch (*request*) an das WSN und verwendet bei Vorliegen mehrerer Antworten, die mit der höchsten Empfangssignalstärke aufgrund der Annahme, dass das zugehörige WSN-Element lokal am nächsten liegt. Dieses Verfahren ist einfach und direkt, aber mit dem Nachteil behaftet, dass nicht in jedem Fall mit einer oder einer eindeutigen Antwort gerechnet werden kann. Der andere Weg besteht darin, vom WSN die Messwerte aller einzelnen Elemente für das Feld oder eines Teilstücks davon in einer Art „*Snapshot*" anzufordern und unter Kenntnis der WSN-Geometrie (vgl. Gerätebeschreibung von Sensorsystemen) sich im Rahmen eines *JDL Level 1 Processing* einen virtuellen Messwert bezogen auf die eigene aktuelle Position der Traktor-Geräte-Kombination zu errechnen. Dies wäre ein Ansatz ähnlich dem Verfahren zur GPS-Positionsermittlung mithilfe einer virtuellen Referenzstation (VRS), die über ein Netzwerk von echten Referenzstationen erzeugt wird [Hol09]. Allerdings könnte ein umfassendes *JDL Level 1 Processing* bereits im WSN selbst durchgeführt werden, so dass dann für jedes Attribut eine eigene Karte übergeben werden könnte. Damit würde jedoch ein Punkt erreicht, an dem sich die Frage stellt, ob die Einstufung des WSN als sozusagen lokaler Online-Sensor noch Gültigkeit besitzt oder bereits die Schwelle zum langfristig deklarativen Wissen, d.h. den *Precision Farming*-Karten, überschritten wurde. Dieser Grenzfall stellt heraus, wie wichtig und hilfreich der Ansatz des Prozessmodells ist, Wissen nach Beständigkeit bzw. Gültigkeit zu unterscheiden und dementsprechend eine Problemlösungsform abzuleiten. Diese angestellten Überlegungen zum WSN weisen auch den Weg, wie die Integration von Fernerkundungsdaten (vgl. Kap. 2.3.2.2) diskutiert werden kann, falls sie nicht nur als Quelle zur Kartierung sondern aufgrund der Weiterentwicklung im Bereich der UAV theoretisch auch als möglicher Online-Sensor betrachtet werden.

Zum Abschluss des Kapitels soll im Folgenden der Vorschlag für einen konkurrierenden Zugriff auf Geräteressourcen einer genaueren Betrachtung unterzogen werden. Der dargelegte

Vorschlag wurde in Grundzügen von OSTERMEIER ET AL. (2003) [OAD03] vor Jahren bereits vorgestellt, eine praktische Umsetzung im ISOBUS ist jedoch nicht bekannt. Technische Gründe dürften dafür nicht den Ausschlag gegeben haben. Die minimale Normerweiterung zur Ablehnung eines Sollwert-Kommandos wäre über die Nachricht „*Process Data Negative Acknowledge* (PDNACK)"[53] [ISO11783-10] realisierbar. Die bisher definierten Ablehnungsgründe sind zwar nicht passend, aber sowohl auf Bit-, Byte- als auch *Command Value*-Ebene ist noch Raum für Erweiterung einer Ablehnungsbegründung (z.B. „*Rejection due to competing setpoint with higher priority*"). Wesentlich wahrscheinlicher für die Nichtanwendung ist die Tatsache, dass bisher keine Notwendigkeit bestand. Entweder kommandierte nur die Bedienperson manuell einen N-Applikationssollwert oder der *Task Controller* bei der Umsetzung des Kartierungsansatzes bzw. beim proprietären „Durchschleifen" der N-Online-Sensor-Applikationsraten. Mit dem neuen Vorschlag zur Normergänzung [ISO11783-10DAM1] - Kapitel 6.6.3 „*Real time sensor based control*" wurde ein alternativer Weg beschrieben, der sich folgendermaßen zusammenfassen lässt:
Es wird ein *Task Controller-Working Set* mit Sollwertquellen (*setpoint value source*) und Sollwertsenken (*setpoint value user*) definiert. Sollwertsenken haben in ihrer Gerätebeschreibung *Device Process Data* (DPD)-Objekte, die als „*Settable*" definiert sind. Sollwertquellen besitzen eine Gerätebeschreibung mit gleichartigen *Device Process Data* Objekten, die eine neu definierte Eigenschaft „*Setpoint Source*" aufweisen. Die Eigenschaften „*Settable*" und „*Setpoint Source*" lassen sich nur wechselseitig setzten. Eine Überlagerung mit einer Karte erfolgt dadurch, dass eine Sollwertquelle ebenfalls *Device Process Data* (DPD)-Objekte als „*settable*" definiert haben kann. Diese „*settable*" Objekte können damit von einer anderen Sollwertquelle beschrieben werden. Beispielsweise indem ein positionsabhängiger Sollwert vom *Task Controller* zum N-Online-Sensor gesendet (, d.h. geschrieben,) wird, dort wird mit diesem Sollwert und einem spezifischen Algorithmus der Sensor-N-Sollwert modifiziert und dann als neuer endgültiger Sollwert an die finale Sollwertsenke, den Düngerstreuer, gesendet. Bei Initialisierung eines Auftrags im *Task Controller* erfolgt eine Zuweisung von Datenquellen und Datensenken mit vergleichbaren Mechanismen, die in der Norm bereits für die Auswahl zwischen manueller oder automatischer Steuerung definiert wurden.
Die Bewältigung eines konkurrierenden Zugriffs auf Geräteressourcen kann auch aus einer *Data Fusion*-Sichtweise als eine Aufgabenstellung der Fusion auf Entscheidungsebene

[53] A. a. O., S. 45: B.6 Process Data Negative Acknowledge (PDNACK) message; Command value= D_{16}

betrachtet werden. Die beiden Verfahren unterscheiden sich demzufolge darin, dass diese Arbeit die Fusion in einem „intelligenten Geräterechner“ vorschlägt, während der zweite Vorschlag die Fusion in einem *Task Controller-Working Set* verortet mit dem *Task Controller* als *Working Set Master*. Beides sind gangbare und skalierbare Wege. Der zweite Weg erscheint zielführender, da der *Task Controller* als die „intelligenteste Stelle“ des ISOBUS-Systems hinsichtlich der Auftragsbearbeitung wahrgenommen wird und auf bereits in der Norm integrierte Strukturen aufsetzt. Der Gedanke an „Intelligenz“ im Geräterechner erschließt sich nicht unmittelbar, nur wird hierbei davon ausgegangen, dass die intelligenten MSDF-Lösungen bereits in einem „*In-field Controller*“ erfolgen und dort nicht auf die Entscheidungsebene beschränkt bleiben. Da davon auszugehen ist, dass der neue Normergänzungsvorschlag die Lösung für den ISOBUS werden wird, bleibt ergänzend anzumerken, dass der „*In-field Controller*“ sich nahtlos einfügt. Er wäre eine weitere Sollwertquelle in einem *Task Controller-Working Set.*

Bei aller Unterschiedlichkeit ist beiden aufgezeigten Lösungen ein statisches Grundprinzip gemeinsam. Die Diskussion soll daher nicht enden, ohne eine flexible Lösung gegenüberzustellen. Eine im wirklichen Sinne flexible Lösung könnte sich an methodischen Ansätzen aus dem Bereich der Multiagentensysteme „*(networks of) autonomous agents*“ orientieren. Die grundlegende Idee dabei ist, dass ein intelligenter Geräterechner versucht, eine Art Kompromiss für alle anstehenden Sollwertwünsche zu finden. Hierzu sendet jeder sollwertkommandierende Teilnehmer nicht nur seinen Exaktwert sondern auch einen Sollwertbereich, der für ihn noch akzeptabel wäre. Die Ausgangssituation im LBS und ISOBUS haben SPANGLER ET AL. (2001) [SAD01] beschrieben und im Rahmen der Implementierung der *Open Source* „ISOAgLib“ (vgl. Kap. 2.4.2) auch entsprechende Beispielanwendungen skizziert. Im Zusammenhang mit dem Sensor-Ansatz mit Kartenüberlagerung und der Systemarchitektur mit einem „*In-field Controller*“ haben OSTERMEIER ET AL. (2003) [OAD03] diese Idee aufgenommen und darauf angewandt: „.... *It is a decisive basic concept that every single implement controller has “intelligence” to derive a solution from competing setpoints with regard to their value and time of arrival. In addition to the necessity of a priority algorithm, sending setpoint ranges instead of exact values, whenever possible, can facilitate effecting a compromise for the finally executed control point. ...*“[54] Unter Inkaufnahme eines erhöhten Nachrichtenaufkommens ist keine neue Definition eines Bereiches nötig, sondern es können die bereits definierten Minimal- und

[54] A. a. O., S. 517

Maximalsollwerte in Ergänzung zu jedem Exaktsollwert genutzt werden. Für ein minimal erweitertes Protokoll zur Ablehnung eines Sollwert-Kommandos bietet die angesprochene Nachricht „*Process Data Negative Acknowledge* (PDNACK)" genügend Spielraum. Für Fälle einer umfangreicheren Protokollerweiterung (z.B. Information und Bestätigung) bieten die „*Command parameter"* der Prozessdaten-Nachrichten (*process data message; PGN: 51968)* einen ausreichend großen freien Bereich.

Würde ein derart flexibles Konzept mit Sollwertanpassung umgesetzt, ließe sich damit ein extrem vereinfachter „*In-field Controller*" direkt im Geräterechner implementieren. Der *Task Controller* gibt mit einem Minimal-Maximal-Intervall die untere Schranke (*Default*-Wert) für den Düngebedarf und die umweltbedingte obere Schranke vor. Liegt die Sollwertempfehlung der N-Online-Sensorik innerhalb des Intervalls so wird sie direkt ausgebracht. Liegt der Sollwert unterhalb der minimalen *Task Controller*-Applikationsrate, so wird der *Default*-Wert appliziert. Wohingegen bei einem Wert über dem *Task Controller*-Maximalwert, die Applikation auf diese Rate begrenzt wird. Die Bedienperson kann jederzeit übersteuern. Dieser stark vereinfachte Ansatz ist somit ein weiterer Lösungsweg, der auf *Data Fusion* auf Entscheidungsebene basiert, nur dass er keine statischen Zuordnungen erfordert.

6.2.4 Verfahrenstechnische Einordnung

Die Bandbreite der Systemansätze für teilflächenspezifische Prozessführung in mobilen Applikationssystemen ist vonseiten der Wissenschaft seit geraumer Zeit erkannt und strukturiert worden. In der praktischen Anwendung ist der Kartierungsansatz der am besten unterstützte und sowohl in landwirtschaftlichen BUS-Systemen als auch in herstellerspezifischen Agarcomputer-Lösungen am meisten umgesetzte Systemansatz. Der Kartierungsansatz war stets Teil der *Task Controller* Definition des ISOBUS und so ist zu erwarten, dass mit fortschreitender ISOBUS-Marktdurchdringung jeder interessierte Betriebsleiter noch mehr Möglichkeiten haben wird, eine auf seinen Betrieb abgestimmte herstellerunabhängige Lösung zusammenzustellen.

Mit einer überschaubaren aber wachsenden Anzahl von N-Online-Sensoren ist auch der Sensor-Ansatz in der landwirtschaftlichen Praxis angekommen. Im Gegensatz zum Kartierungsansatz stehen noch keine durchgängigen ISOBUS-Lösungen zur Verfügung, sondern es ist stets eine herstellerspezifische Komponente enthalten, um ein einsatzbereites Applikationssystem zu erreichen. Mit dem neuen Ergänzungsvorschlag der ISOBUS *Task Controller*-Definition zum Sensor-Ansatz ist damit zu rechnen, dass sowohl Online-Sensor- als auch *Task Controller*-Hersteller ihre Produkte mit der Zeit um diese Funktionalität

erweitern werden. Der Kunde wird somit auf längere Sicht ein Marktangebot vorfinden, das es ihm gestattet, ein nach seinen Anforderungen zusammengestelltes System aufzubauen, ohne an einen oder wenige Hersteller gebunden zu sein.

Zugleich ermöglicht die im Normvorschlag enthaltene Option zur Überlagerung mit Applikationskarten eine Realisierung des umfassenden Sensor-Ansatz mit Kartenüberlagerung. Damit ist absehbar, dass die Landmaschinenindustrie ISOBUS-Produkte offerieren wird, die die gesamte Bandbreite von Prozessführungen zur mobilen teilflächenspezifischen Applikation ermöglichen. Diese Skalierbarkeit findet jedoch ihre technische Grenze, wenn der Sensor-Ansatz mit Kartenüberlagerung in herstellerunabhängiger Form nicht nur auf eine MSDF auf Entscheidungsebene reduziert werden soll, sondern auch aufwendige Optimierungsstrategien machbar sein sollen. Am Beispiel einer Real-time Prozessführung in sensorgestützten Düngesystemen nach dem Sensor-Ansatz mit Kartenüberlagerung für intensive Stickstoffdüngung wurden die grundsätzlichen Entwurfs- und Analyseverfahren gezeigt und zugleich offengelegt, welche zusätzlichen ISOBUS-Normfestlegungen gefordert wären.

Das gewählte praktische Beispiel verfügt über breite anwendungsorientierte Relevanz, da dieser Prozessführungsansatz im Grunde auf andere kleinräumig durchführbare Applikationstätigkeiten wie Saat, Pflanzenschutz und Bewässerung übertragen werden kann.

Vorausgesetzt, dass auch die vorgeschlagene Umsetzung des Sensor-Ansatz mit Kartenüberlagerung basierend auf umfassender MSDF-Technik Einzug in die Norm und den ISOBUS hält, dann stehen von technischer Seite aus alle Möglichkeiten einer individuellen standortangepassten Prozessführung zur Verfügung.

Der interessierte Betriebsleiter sieht sich damit jedoch weiteren oder neuen Fragestellungen gegenübergestellt und ist gut beraten, sie aus dem Blickwinkel einer durchgängigen Systemlösung zu beantworten. Grundsätzlich werden auf ihn Kosten im Bereich der Beschaffung der Online-Sensoren und einer ISOBUS-tauglichen Applikationstechnik zukommen. Auch stellt sich die Frage, ob die Applikationstechnik eine ausreichende Applikationsgenauigkeit und -dynamik vorhält. Die Auflösungs- und Erfassungsbereiche der Online-Sensorik sowie die Qualität der *Precision Farming*-Karten sind in Abstimmung mit den Fähigkeiten der Applikationstechnik zu bewerten. Da der „*In-field Controller*“ in der Regel bei einer vorhandenen *Task Controller*- und *Virtual Terminal*-Hardware nur eine weitere Softwareapplikation ist, sind die Mehrkosten von den Preismodellen (z.B. aufpreisfrei, Abonnement, Kauflizenz) der Hersteller abhängig.

All diese Überlegungen helfen ein ideales wissensbasiertes Real-time Prozessführungssystem

auszulegen und den Investitionsbedarf abzuschätzen. Ein entscheidender Aspekt für die Kosten-Nutzen-Abschätzung wird jedoch die Verfügbarkeit und die Qualität des eingesetzten prozeduralen Wissens sein. Im Stand der Technik wurde bereits darauf hingewiesen, dass die ökonomische und ökologische Bewertung auch im wissenschaftlichen Bereich noch eine herausfordernde Fragestellung ist. Ein weiterer Ansatzpunkt besteht darin dies in einem On-farm-Research-Ansatz zumindest betriebsspezifisch anzugehen, um von Herstellerangaben oder externen Versuchsergebnissen unabhängig zu sein [Gan05].

Hinsichtlich der Verfügbarkeit des prozeduralen Wissens eröffnen sich neben staatlichen Beratungsstellen auch für private Akteure in der landwirtschaftlichen Produktionskette neue Geschäftsfelder. Dienstleister bzw. Berater könnten als eine Art „Wissensprovider“ agieren. Auch eröffnen sich für Landmaschinenhersteller oder Firmen der chemischen Industrie (z.B. Düngemittel, Pflanzenschutz) neue Möglichkeiten zur Kundengewinnung und -bindung.

Ein sozialer Aspekt darf beim Einsatz einer wissensbasierten Lösung nicht vergessen werden. Ist der Kunde bzw. Betriebsleiter überhaupt gewillt einen großen Teil seiner Entscheidungshoheit an ein „intelligentes System“ abzutreten und auf fremdes Fachwissen zu vertrauen, das im Extremfall in einer zukünftigen service-orientierten Produktionsumgebung von einem rein elektronischen Dienstprogramm, d.h. E-Service, erbracht wird.

6.3 Simulation

Das angestrebte Ergebnis des zweiten Teilziels war, den Sensor-Ansatz mit Kartenüberlagerung für die intensive Stickstoffdüngung als Software-Simulation mit intuitiver Interaktionsmöglichkeit zu realisieren und zu testen. Der Schwerpunkt sollte dabei auf der MSDF liegen. Zugleich sollte auch eine angemessene Entwicklungsmethodik angewandt werden, um ein für die gesamte Arbeit durchgängiges Vorgehensmodell vorstellen und auf Eignung überprüfen zu können.

Zusammenfassend ist festzustellen, dass die Vorgabe ohne nachträgliche Abstriche bei den Anforderungen in eine voll funktionsfähige Simulation umgesetzt werden konnte. Das in vorhergehenden Kapiteln abgeleitete Problemlösungsparadigma für die MSDF für das *JDL Level 2 Processing* wurde implementiert und über eine GUI einfach verfolgbar und bedienbar gestaltet. Die Möglichkeiten und Vorteile, die die Zusammenarbeit in der multidisziplinären DFG-Forschergruppe des IKB-Dürnast-Projektes bot, wurden intensiv genutzt und waren der Garant für die gelungene Umsetzung und Eignungsprüfung.

Die nachfolgende Auseinandersetzung mit den Ergebnissen im Einzelnen folgt der Gliederung in die Kategorien Entwicklungsprozess, Realisierung der Simulation und Testen.

6.3.1 Entwicklungsprozess

Die Entwicklung der Simulation war bestimmt durch die generelle Ausrichtung „*proof of concept*“. Geleitet durch diese Grundidee ist die Beschränkung auf Erstellung eines Labor-Prototyps als Simulation zulässig, der ausschließlich Wert auf eine detaillierte Umsetzung des *JDL Level 2 Processing* legt und nur die dafür direkt nötigen weiteren Bestandteile, wie z.B. eine simulierte Prozessumgebung oder eine Datenbankanbindung, zusätzlich realisiert. Es steht außer Frage, dass eine Simulation eines kompletten Landwirtschaftlichen BUS-Systems mit einer variablen Verschiebung von Hard- und Softwaregrenzen im Sinne einer „*hardware in the loop*“-Simulation das wünschenswerte Ideal darstellt. Jedoch gilt es, stets den zusätzlichen Aufwand und Ressourcenbedarf mit dem möglichen Erkenntnisgewinn abzuwägen. Ein derartiges System war zur Projektlaufzeit am TU München Fachgebiet Technik im Pflanzenbau nicht verfügbar und der Schwerpunkt wurde bei einer für die Agrarsystemtechnik neuen Disziplin auf die theoretischen Grundlagenarbeiten zur MSDF und einer exemplarischen Algorithmus-Implementierung sowie der dafür adäquaten Entwicklungsmethodik gelegt.

Bei der im Prozessmodell abgeleiteten Problemlösungsform vom Typ Produktionssystem lag es nahe, die Realisierung der Simulation als eine Entwicklung eines Expertensystems zu sehen und auszuführen. Regelbasierte Expertensysteme zählen zu den ältesten, bekanntesten und am weitesten verbreiteten wissensbasierten Systemen [Kur89], [Lug01]. Ihre Problemlösungskomponente ist die Umsetzung eines Produktionssystems. Von dem Grundgedanken der Entwicklung eines Expertensystems geleitet, kam ein darauf abgestimmtes Vorgehensmodell zum Einsatz. Im Stand der Technik wurde aufgezeigt, dass der Anwendungsbereich dieses Vorgehensmodell nicht auf regelbasierte Expertensystem beschränkt ist, sondern als genereller Entwicklungsprozess für wissensbasierte System zum Einsatz kommt. Für die Entwicklung eines wissensbasierten *Data Fusion*-Algorithmus als *Embedded*-Komponente einer Prozessführung bedeutet dies in letzter Konsequenz, dass sich der grundsätzliche Entwicklungsprozess nicht von dem der Entwicklung eines vollständigen Expertensystems mit seinen weiteren Komponenten, wie z.B. Erklärungsteilsystem, Benutzerschnittstelle und Wissensbasiseditor, unterscheidet.

Wird der evolutionäre Entwicklungsprozess für wissensbasierte Systeme (vgl. Kap. 5.1.3) in Vergleich zu den in der Agrarsystemtechnik vorherrschenden linearen oder V-förmigen phasenorientierten Vorgehensmodellen (vgl. Kap. 2.5) gesetzt, so fällt der drastische Bruch in der Methodik sofort ins Auge. Die Diskrepanz mit den konventionellen Phasenmodellen wird

einhellig in der KI-Gemeinschaft bestätigt (z.B. [Fri03][55], [Kur89][56], [Lug01][57], [Sri97][58]). Damit lässt sich die Entwicklungsmethodik noch am ehesten mit dem Ansatz „*Agile Software Development*“, wie ihn LENZ UND MÜNCH (2005) [LM05] fordern, in Übereinstimmung bringen. Die Anforderungen zur Gewährleistung der funktionalen Sicherheit stellen hingegen eine große Herausforderung dar und bekräftigen nochmals die Handlungsempfehlung, (vgl. Kap. 6.1), äußert genau zu prüfen, ob der MSDF-Anteil der Prozessführung unbedingt sicherheitskritisch ausgelegt werden muss.

Kritisch hinterfragt, ließe sich unter Anbetracht des Bruches beim Übergang auf die Feinentwurf-Phase des MSDF-Framework ein Widerspruch mit der allgemeinen Zielsetzung ausmachen. Ein evolutionäres Vorgehen passt auf den ersten Blick nicht zur geforderten Zielorientierung, Effizienz und Durchgängigkeit bei der Umsetzung der Anforderungen hin zu einer implementierten Lösung. Mit der Einführung der Sichtweise nach SRIRAM (1997) [Sri97] einen Teil der Identifikation, die Konzeption und Formalisierung als eine Transformation zu betrachten, ist zumindest die Zielorientierung und Durchgängigkeit im Vorgehen gewahrt. Dies bedeutet nicht, dass es zu keiner Revisionsschleife kommen darf, jedoch ist dabei klar an welcher Stelle sich der Prozess im gesamten MSDF-Framework befindet. Der Punkt der Effizienz ist differenziert zu betrachten. Ein evolutionäres Vorgehen steht immer in der Gefahr, dass in nachfolgenden Versionen die komplette Konzeption überarbeitet werden muss, weil bei der Nullversion Kernanforderungen übersehen wurden, und dass die Nullversion nicht flexibel genug ist, um sich an ungeplante Evolutionspfade anzupassen [Bal98]. Aber gerade durch die umfangreiche Vorarbeit im Funktionalen Modell, dem Prozessmodell und bei der Systemarchitektur startet der Entwicklungsprozess für das wissensbasierte System nicht bei null, sondern mindestens die erste Bearbeitung des *Specification Level* bis zum *Knowledge-base Level* wurde bereits durchgeführt. Sollte bei dieser Transformation festgestellt werden, dass das abgeleitete Problemlösungsparadigma sich nicht als tragfähig erweist, so kann direkt in das Prozessmodell zurückverzweigt werden. Solange das Herantasten an die optimale Konzeption und Formalisierung auf diesen abstrakten Ebenen stattfindet, sind Zeitverluste und Ressourceneinsatz im Vergleich zu den Hardware- und Softwareimplementierungsschritten und der Produktwartungsphase nachrangig. Im Sinne von Effizienz darf auch nicht unberücksichtigt bleiben, dass die Wahl

[55] A. a. O., S. 26

[56] A. a. O., S. 83

[57] A. a. O., S. 284

[58] A. a. O., S. 547

einer nicht geeigneten Problemlösungsform die ineffizienteste Lösung darstellt, auch wenn sie mit einer konventionellen Entwicklungsmethode umsetzbar sein sollte.

Auf die außerordentliche Bedeutung der Wissensakquisition am Entwicklungsprozess eines wissensbasierten Systems wurde im Stand der Technik hingewiesen. In diesem Zusammenhang sind zwei Aspekte der vorliegenden Arbeit näher zu betrachten. Dies ist zum einen die Rolle und Verfügbarkeit von Wissensingenieur und Fachexperten, zum anderen die Frage nach der Möglichkeit der Wissenspflege. In der Literatur finden sich Stellen (z.B. [Kur89]), die zu bedenken geben, dass der Wissensingenieur und der Fachexperte nicht ein und dieselbe Person sein sollen. Gegen diese Empfehlung wurde im Rahmen dieser Arbeit bei mehreren der Wissensmodule verstoßen. Dies wurde dennoch als akzeptabel eingestuft, da für das relativ einfache agrarsystemtechnische Wissen der Autor als Fachexperte gelten kann und auch Rücksprache mit Kollegen am Fachgebiet gehalten hat. Des Weiteren war das Ziel der Entwicklung ein lauffähiger Prototyp zur Demonstration der MSDF-Technik und deren Leistungsfähigkeit, was selbst bei Regeln mit diskussionswürdiger Qualität noch möglich ist. Ebenso kritisch kann das Agieren des Autors bei der Implementierung der Simulation gesehen werden, bei der er auch die nicht wissensbasierten Anteile (z.B. GUI, Datenbankanbindung oder Prozessumgebungssimulation) codierte und zugleich auch die Verifikationstätigkeit übernahm. Diese Vorgehensweise ist den begrenzten Ressourcen in Forschungsprojekten geschuldet. Da aber eine reine Software-Simulation ohne eine Kommerzialisierungsabsicht erzeugt wurde, bestehen keine Haftungsrisiken und der Entwickler war bestrebt ein voll funktionsfähiges System zu realisieren, das Vorführungen vor unterschiedlichem Publikum standhält. Weiterhin wurde dieser Arbeit im Anhang (vgl. Kap. A.4) eine ausführliche Beschreibung der grundlegenden Implementierungskonzepte beigelegt, so dass der interessierte Leser das Design des Systems nachvollziehen kann, das zur Leistungsmessung (vgl. Kap. 5.2.4.2) verwendet wurde.

Vorbildlich gelang jedoch die Rollenaufteilung mit dem IKB-Dürnast Projektpartner des TP 12 im Fall des wichtigen Wissensmoduls „*CROP_PRODUCTION*“, bei dessen Wissensakquisition somit nie unter einem *knowledge engineering bottleneck* gelitten wurde. Die wiederholten Hinweise in der Literatur (z.B. [Kur89], [Sri97]) auf diese Problematik lassen eine typische Schwierigkeit einer Expertensystementwicklung vermuten.

Sowenig die Entwicklung von Engpässen bei der Verfügbarkeit von Wissensingenieur und Fachexperten betroffen war, so kritisch würde sich die Situation in Richtung Wartung der Wissensbasis, d.h. der Wissenspflege, darstellen. Mit dem Ausscheiden der beteiligten IKB-Dürnast-Projektbearbeiter kann das Wissen nicht nahtlos erweitert oder modifiziert werden.

Grundsätzlich macht die Aufteilung des Systems in Inferenzmaschine und Wissensbasis sowie die regelbasierte Repräsentationsform diese Modifikation wesentlich einfacher als bei einem konventionellen Softwaresystem. Da aber das System nur unter Zuhilfenahme eines Editors codiert wurde, ist nicht der Komfort gegeben, wie er bei kommerziellen Tools zur Expertensystementwicklung mittlerweile erhältlich ist, die Regeln grafisch zusammenstellen und modifizieren lassen. Weiterhin ist kein Tool zur Konsistenzprüfung des Wissens vorhanden und erfordert die Expertise des Fachexperten und eines Wissensingenieurs.

Bevor die Diskussion des Entwicklungsprozesses für den MSDF-Ansatz beendet und noch ein kurzer Blick auf die interaktive Benutzerschnittstelle geworfen wird, soll ein Hinweis in Sachen Expertensystem-Shell festgehalten werden. Obwohl diese Arbeit von der Systemarchitektur abwärts eine systemtechnische Vorgehensweise angewendet hat, wird auf *Tool Level* zum Teil ein hybrider Ansatz genutzt. Im Prinzip ist eine Expertensystem-Shell ein Ansatz, der mit *System Architecting* gut harmoniert. In der Regel entsteht eine Expertensystem-Shell aus einer abgeschlossenen Expertensystementwicklung, indem es mit „leerer“ Wissensbasis für andere Anwendungen und Projekte angeboten wird, die einen ähnlichen *use case* darstellen.

Verglichen mit der Implementierung des *JDL Level 2 Processing* wurde die Entwicklungsmethode für die Realisierung der Benutzerschnittstelle nicht mit der gleichen Aufmerksamkeit bedacht. Laut LANGMANN (1994) [Lan94] ist das Thema der Mensch-Maschine-Schnittstelle methodenorientiert sowohl im Bereich der Softwaretechnik wie auch in der Ergonomie und den kognitiven Wissenschaften angesiedelt. Zumindest bewusst wurde bei der Implementierung den Themen Ergonomie und kognitive Wissenschaften keine Beachtung zuteil. Auch im Bereich der Softwaretechnik kam kein ausgesuchter Entwicklungsprozess zum Einsatz. Die Entwicklung war an den Möglichkeiten der Grafikbibliothek Swing und von JESS orientiert und die GUI wurde in einem evolutionären Verfahren erstellt. Am besten lässt sich das Vorgehen als Entwicklung eines evolutionären Prototyps basierend auf objekt-orientierter (GUI)-Technik beschreiben, der Bestandteil des Zielsystems wurde. Diese Methode ist prinzipiell geeignet zur Erstellung von Mensch-Maschine-Schnittstellen und LANGMANN (1994) [Lan94] sieht den wichtigsten Vorteil des *Prototyping* darin, dass die Lösungen benutzergerechter werden, die Realitätsablösung der Entwickler vermieden werden und Prototypen auch für das externe Management eine wirkungsvolle Präsentationsgrundlage schaffen.

6.3.2 Realisierung der Simulation

Entsprechend dem Aufbau des zugehörigen Ergebnisteils wird nachfolgend ein kritischer Blick auf die Schritte der Transformation über fünf *Levels* hinweg und die abschließende Implementierung geworfen.

Die beiden Ebenen *Specification Level* und *Task Level* wurden in dieser Arbeit mehrfach und aus verschiedenen Blickwinkeln dargestellt und ausreichend erörtert.

Der *Problem Solving Level* hingegen bietet sich an, über einige Ergebnisse zu reflektieren. Eine der wegweisenden Entscheidungen auf dieser Ebene wurde bereits über das Prozessmodell geklärt, die Selektion einer oder mehrerer such- oder wissensbasierten Problemlösungsformen. Über das Prozessmodell wurde die Entscheidung für die Problemlösungsform, bestehend aus den beiden „starken Problemlösungsformen" Produktionssystem und Vorwärtsverkettung, getroffen. An dieser Stelle sollte sich der Wissensingenieur nochmals die Frage stellen, ob ein derartiges Expertensystem überhaupt das geeignete Mittel für die Aufgabenstellung ist. Hierzu ist es hilfreich, die Kriterien Schließen in Raum und Zeit sowie den Umfang der Problemdomäne zu betrachten. SRIRAM (1997) [Sri97] gibt hinsichtlich Expertensystem-Technik zu bedenken, dass „*It is hard to incorporate temporal reasoning and spatial reasoning (at least for many practical applications).*"[59] Der Sensor-Ansatz mit Kartenüberlagerung ist als Prozessführung für eine teilflächenspezifische Aufgabenstellung mit Wissensarten von unterschiedlicher Dauerhaftigkeit und Gültigkeit nicht frei von einem räumlichen und zeitlichen Aspekt bei der Schlussfolgerung, jedoch kann stets bezogen auf eine Teilfläche mit einem „Schnappschuss" von allen Prozesssignalen, sogenannten „eingefrorenen Prozesszuständen", gearbeitet werden. Dies ist ein bekannter Lösungsansatz für diese Problematik im Bereich der wissensbasierten Verfahren für die Automatisierungstechnik [Kre91][60].

Ein Expertensystem ist nur in einer eng beschränkten Problemdomäne leistungsfähig, werden deren Grenzen überschritten, versagt das System [Kur89], [Lug01]. Daher wird auch Alltagswissen oder der „gesunde Menschenverstand" (*common sense*) nicht von einem Expertensystem bewältigt oder bereit gestellt. Die Aufgabenstellung laut Definition im Funktionalen Modell ist genügend eng begrenzt und sollte daher grundsätzlich über ein entsprechendes regelbasiertes Expertensystem lösbar sein. Mit diesem sogenannten Oberflächenwissen oder „flachen Wissen" ist jedoch die Problematik verbunden, dass die

[59] A. a. O., S. 551

[60] A. a. O., S. 16

Menge an nötigen Regeln zu einer schwer handhabbaren Komplexität führen kann [Kre91]. Bei einem Teil der Systeme lässt sich dieser Punkt über eine Modularisierung weiter hinausschieben.

Ohne dass die vorliegende Realisierung des *JDL Level 2 Processing* bereits zu einer Komplexitätsexplosion geführt hätte, wurde das Mittel der Modularisierung zur Anwendung gebracht, um eine bessere Übersichtlichkeit zu gewährleisten, Zwischenschritte einfacher überprüfbar zu machen und die Wissenspflege prinzipiell zu erleichtern. Auch der Vorteil eines besseren Zeitverhaltens durch kleinere Suchräume wurde im Ergebnisteil bereits herausgestellt. Es wird ein Algorithmus mit hoher exponentieller Ordnung in eine Summe von Algorithmen mit niedrigerer exponentieller Ordnung gewandelt. Die Aufteilung in geeignete Module war durch die funktionale Modellierung einfach möglich. Hervorzuheben ist, dass unter Beibehaltung der Wissensmodule zur Einbeziehung von Beschränkungen, der technischen Situationsbewertung und der Zusammenfassung und Applikationssollwertempfehlung das Modul zur pflanzenbaulichen Situationsbewertung (*„CROP_PRODUCTION“*) einfach getauscht werden kann. Diese Funktion ist von entscheidender Bedeutung für die diskutierten Lösungen im Bereich der Systemarchitektur, bei denen prozedurales Wissen über Wissensprovider gewartet werden soll (vgl. Kap. 6.2.3).

Wird ein Vergleich mit den Angaben des Funktionalen Modells angestellt, fällt auf, dass das Wetter unberücksichtigt bleibt. Abgesehen von dem Einfluss auf die Applikationstechnik ist dieser Faktor vor allem für die pflanzenbauliche Bewertung von großer aber nicht einfach zu handhabender Bedeutung. In der vorliegenden Arbeit fand das Wetter daher keine explizite Beachtung, ist aber implizit im Regelwerk (vgl. *Knowledge-base Level*) integriert. Dazu ging WEIGERT (2006) [Wei06] bei dem WED-Prozess von folgender Annahme aus: „…. Die Technologie der reflexionsoptischen Messungen …. Vereinfacht werden von der Pflanze „verarbeitete“ Boden- und Klimaverhältnisse während der Vegetationsperiode („*in-season*“) erfasst. Die Pflanze wird somit als Indikator benutzt. … Anstelle von Witterungskennzahlen, die lediglich Rahmenbedingungen beschreiben, können nun direkt Informationen von der Pflanze erfasst werden, die ein Integral der Wachstumsbedingungen bis zu diesem Zeitpunkt abbilden. …“[61]

Im Bereich der Einstell- und Übersteuerungsfunktionen für die Bedienperson drängt sich die Frage auf, inwieweit dies überhaupt einen Sinn auf der wissensbasierten Ebene ergibt und ob es nicht außerhalb in der übergelagerten Steuerung besser angesiedelt wäre. Im vorherigen

[61] A. a. O., S. 60

Kapitel wurde bereits darauf verwiesen, dass Fragen der *user experience* nicht die höchste Priorität beigemessen wurden. Ein entscheidender Vorteil wurde jedoch darin gesehen, dass die Eingaben nahtlos in die Erklärungs- und Dokumentationsfunktion des Systems integrierbar waren. Gerade unter dem Aspekt der funktionalen Sicherheit ist aber die Benennung der Funktion „Notaus“ zu überdenken und besser in „*Pause/Resume*“ zu ändern.

Neben diesen grundlegenden Fragen zur Problemlösung spielen auf dem *Problem Solving Level* noch weitere Kriterien eine Rolle. Im Rahmen dieser Arbeit sind dies die Konsistenz des Wissens, der Umgang mit unsicherem Wissen (*inexactness*) und die Konfliktlösungsstrategien. Die entsprechenden angewandten Lösungen wurden im Ergebnisteil aufgezeigt.

Die Lösungen für den Bereich der Konsistenz sind formal nicht begründet und wenden nur Heuristiken an, die in den „...-*SUM_UP*“-Modulen zusammengefasst sind. In Ermangelung einer besser theoretisch fundierten Grundlage ist dies für ein wissensbasiertes System ein gangbarer Weg. Die Herausforderungen in diesem Bereich sind in der Fachdisziplin bekannt, was SRIRAM (1997) [Sri97] folgendermaßen ausdrückt: „*methodologies to deal well with inconsistent knowledge do not yet exist;*“ [62].

Der Umgang mit *inexactness* zeigt die Vorteile eines gut ausgearbeiteten Konzepts im Bereich des Funktionalen Modells und des Prozessmodells. Gerade auf *JDL Level 1 Processing*-Ebene sind eine Vielzahl von Algorithmen und Methoden (vgl. [HM04]) zum Umgang mit unsicherem Wissen bekannt. Daher ist es konsequent, soweit der grundsätzliche Ansatz es zulässt, dies von der *JDL Level 2 Processing*-Ebene fernzuhalten, die im Falle eines Produktionssystems im Wesentlichen nur die Anwendung von Konfidenzfaktor-Algebra [Lug01] oder das Schließen mit unscharfen Mengen („*fuzzy rules*“) [Fri03] kennt. Dies ist ein Gedankengang, der auch gut zu einer Schlussfolgerung von BLANK ET AL. (2011) [BBMK11] hinsichtlich ihres „*Modular Sensor Fusion Approach for Agricultural Machines*“ passt: „ …. *it eases the development of new sensor data-based components and subsystems since it offers an abstraction layer for the sensing hardware. Therefore, the designer does not have to worry about the specific origin of a piece of information anymore but can rely on a pool of aligned and reliable measured parameters.* …“[63]. Der Ansatz, dennoch noch eine *inexactness* -Funktion auf die *JDL Level 2 Processing*-Ebene zu heben, ist aus informationstechnischer Sicht nicht sonderlich interessant, aber aus Sicht der Systemansätze für teilfächenspezifische Applikation durchaus beachtenswert. Das Umschalten auf ein

[62] A. a. O., S. 551

[63] A. a. O., S. 9

alternatives Regelwerk im Falle, dass der REIP-Wert zur 2. N-Gabe ausbleibt oder zu unsicher erscheint, ist gleichbedeutend mit einem Wechsel auf den Kartierungsansatz. Somit wurde eine teilweise Skalierbarkeit der Systemansätze in die Simulation integriert. Würde noch ein alternatives Regelwerk für die Desaktivierung der Kartendaten erarbeitet und integriert, könnte mit der Simulation die gesamte Bandbreite von Systemansätzen realisiert und demonstriert werden.

Im Bereich der Konfliktlösungsstrategien kam explizit das einzige Mal ein suchbasiertes Verfahren zum Einsatz, die Strategie der Tiefensuche. Auf den ersten Blick mag es irritierend erscheinen, dass in der Arbeit von einer fünfstufigen Tiefe des Suchbaumes gesprochen wird, obwohl der Entscheidungsbaum für die Düngeregeln bis zu 9 Ebenen umfasst. Der Widerspruch löst sich auf, wenn in einem Ast alle gleichen Attribute zusammengezogen und nicht mit einer binären sondern einer mehrfachen Fallunterscheidung angetragen werden. Dies ist gleichbedeutend mit einer geringeren Tiefe, jedoch mit einem höheren Verzweigungsfaktor. Im Ergebnisteil wurde festgehalten, dass keine Prioritätsangaben für Regeln zum Einsatz kamen. Ein Blick in die Implementierung (vgl. Kap. A.4.1) verrät, dass aber ein durchgängig verwendeter Mechanismus von *salience* Gebrauch macht. Durch einen *salience*-Wert kleiner als die Default-Einstellung aller Regeln, wird nach abgeschlossener Inferenz von einem Wissensmodul auf das nächste weitergeschalten. Mit dem eigentlichen Schlussfolgerungsprozess hat dies jedoch nichts zu tun und widerspricht daher auch nicht dem guten Implementierungsstil eines regelbasierten Expertensystems.

Nicht nur der *Problem Solving Level* lädt zur Diskussion ein, auch der *Knowledge-base Level* hat Aspekte, die zu einer kritischen Betrachtung herausfordern.

Mit der Entscheidung für eine regelbasierte Wissensrepräsentation für das prozedurale Wissen und eine objekt-orientierte Form für das deklarative Wissen wurden zwei Repräsentationsformen gewählt, die in den KI-Kontext der „physischen Symbolhypothese“ nach NEWELL UND SIMON (1976) [NS76] einzuordnen sind. Im Bereich der KI wurde die Vorstellung, dass Intelligenz alleinig über ein Symbolsystem charakterisierbar ist, mittlerweile aufgegeben und nach alternativen Erklärungsformen gesucht. Als alternative Ansätze führt LUGER, (2001) [Lug01] konnektionistisches oder „neuronales“ Rechnen, agentenbasierte emergente Systeme sowie genetische Modelle des Rechnens auf und stellt fest, dass bei KNN der Brennpunkt der KI von Problemen der symbolischen Repräsentation und widerspruchsfreien Inferenzstrategien hin zu Problemen des Lernens und der Adaption verlagert wird. Folglich kamen interessanterweise zwei alternative Erklärungsmodelle für Intelligenz bei der Realisierung der Simulation des Sensor-Ansatzes mit Kartenüberlagerung

zum Einsatz. Dabei ist jedoch festzuhalten, dass sowohl für diese Arbeit als auch für die Wissensakquisition des pflanzenbaulichen Wissens eine abstrakte Beschäftigung mit dem Begriff der Intelligenz keine Rolle gespielt hat. In beiden Forschungsansätzen wurden bekannte und bewährte Methoden auf eigene Fragestellung angewandt und auf Eignung untersucht. Produktionssysteme sind ein bekannter und bewährter Weg für das *JDL Level 2 Processing*, bei dem der Steuerungs- und Regelungsaspekt einer Prozessführung im Mittelpunkt steht, während die Zielsetzung einer automatisierten Wissensentdeckung und damit Wissensakquisition zu den elementaren Herausforderungen des maschinellen Lernens gehört. Die Frage, wieso für den „*In-field Controller*“ nicht direkt das KNN genutzt wurde, war bereits ausführliches Thema in der Diskussion zum Prozessmodell (vgl. Kap. 6.2.2). Unterstützend anzumerken bleibt die in der Literatur mehrfach genannte Intransparenz des Schlussfolgerungs-Prozesses bei einem KNN und den damit verbunden praktischen Akzeptanzproblemen [Lug01], [Sri97], [Wei06].

Die Repräsentationsform des Wissens bestimmt darüber, ob ausreichende Gestaltungs- und Ausdrucksmöglichkeiten zur Beschreibung des Wissens vorhanden sind, jedoch besagt sie noch nichts über die Qualität des erworbenen Wissens. Für diese Arbeit ist die pflanzenbauliche Situationsbewertung mit der ökonomisch-ökologischen Optimierung von besonderer Bedeutung. Werden Vorstellungen des *JDL Level 2 Processing* im Funktionalen Modell und im Prozessmodell mit der Umsetzung in der Simulation verglichen, so fällt das Fehlen des ökologischen Optimierungsschrittes auf, außer das einfache Modell einer „Überlagerungskarte“ mit Begrenzungswerten wird als ausreichend betrachtet. Da diese Arbeit in jenem Bereich von der „Wissenszulieferung“ der IKB-Dürnast-Fachexperten abhängig war, bleibt nur auf die Möglichkeiten hinzuweisen, die WEIGERT (2006) [Wei06] in seiner Diskussion aufzeigt: „… Ebenfalls sind Anwendungen mit einem anderen Zielsystem denkbar, beispielsweise eine Optimierung nach ökologischen Gesichtspunkten. Auch eine ökonomisch-ökologische Optimierung wäre im Sinne einer nachhaltigen Landbewirtschaftung möglich. Die Änderung besteht prinzipiell nur im Zielsystem, das zur Auswahl der Input-Output-Kombinationen herangezogen wird. …“[64]. Weiterhin kann die Einengung auf nur einen eng begrenzten Versuch für die Qualität der Regeln sehr kritisch gesehen werden, zumal die geringe Datengrundlage durch die kleine Versuchsfläche die Ableitung von stabilen Modellen zu einem schwierigen Unterfangen machte [Wei06][65]. Auch ist eine Verallgemeinerungsfähigkeit des eingesetzten prozeduralen Wissens in keiner Weise

[64] A. a. O., S. 176

[65] A. a. O., S. 136

gegeben. Hierzu ist festzuhalten, dass dies für diese Arbeit vollkommen bedeutungslos ist, da nicht die grundsätzliche Qualität des prozeduralen Wissens im Mittelpunkt steht, sondern die Analyse und die Entwicklung einer geeigneten wissensbasierten Prozessführungstechnik, die das zugelieferte Wissen wertfrei zur Anwendung bringt. Stehen qualitativ hochwertigere Regeln zur Verfügung, können sie in gleicher Weise umgesetzt werden. Geachtet wurde jedoch darauf, dass die Umsetzung mit einer für den Sensor-Ansatz mit Kartenüberlagerung für N-Düngung typischen Menge und Art an Prozessvariablen durchgeführt wurde, die eine praktische Übertragbarkeit und Bestimmung relevanter Leistungskenngrößen gewährleistet.
In diesem Zusammenhang lohnt ein Blick auf die Eigenschaft eines Regelwerkes, dem ein Entscheidungsbaumverfahren zu Grund liegt. Dieses Verfahren kann nur unstetige Ergebnisse produzieren, auch wenn die zugrundeliegende Funktion stetig ist. Damit geht sozusagen ein Verlust an Qualität bei den Regeln im Vergleich zu stetigen Ausgangsmodellen einher [Wei06][66]. Dieser Verlust ließe sich verringern, indem der Lösungsraum wesentlich stärker unterteilt wird, was gleichbedeutend einer kleineren Schrittweite bei den Applikationsraten wäre. Es könnte anstatt der diskreten 10 kg N/ha Stufen eine Halbierung auf 5 kg N/ha vorgenommen werden. Aus IT-Sicht würde dies bei einer vollständigen Suche zu einem erhöhten Verzweigungsfaktor und damit zu einer im Mittel verlängerten Laufzeit eines Inferenz-Zyklus führen. Die Umsetzung in ein RETE-Netzwerk ist dadurch charakterisiert, wie viele gemeinsame Muster weitergenutzt werden können. Zu einer Ausweitung des Netzwerkes wird es in jedem Fall kommen. Des Weiteren stellt sich die Frage, ob die Präzision der Applikationstechnik ausreicht, um diese Verfeinerung auch tatsächlich bis zum Ablagepunkt zur Wirkung zu bringen. Kommen Düngerstreuer mit guten bis sehr guten Variationskoeffizienten (=< 10 %) (vgl. Kap. 2.3.2.4) zum Einsatz, sollte sich der höhere Aufwand im Regelwerk lohnen. Dennoch bleibt eine Schwachstelle bei den Mustervergleichen auf die Attribute bestehen. Die Grenzen zwischen unterschiedlichen Entscheidungswegen sind numerisch eindeutig („scharf“), somit kommt der Qualität der Prozesseingangswerte und -zustände eine gewichtige Rolle zu. Auf Ebene der Wissensentdeckung wurde der Versuch unternommen, diese Einflüsse mittels Sensitivitätsanalyse und *Response Surfaces* zu untersuchen. Für den Fall, dass Fachexperten die Schlussfolgerungsergebnisse als nicht ausreichend bewerten, bleibt nur, mit einer Annahme dieser Arbeit zu brechen und auf *JDL Level 2 Processing*-Ebene zu prozeduralem Wissen für das Schließen in unsicheren Situationen, d.h. bei unsicherem deklarativen Wissen,

[66] A. a. O., S. 39 & S. 104

zu wechseln. Hierfür bietet sich ein „Fuzzy-Regel“-basiertes Produktionssystem an, das nach LUGER (2001) [Lug01][67] Ingenieuren ein wirkungsvolles Werkzeug an die Hand gibt, um mit unpräzisen Messungen umzugehen.

Die damit tangierte Diskussion von Implementierungsalternativen lässt sich auch grundlegender führen. Wäre es nicht effizienter, Teile des prozeduralen Wissens konventionell auszuführen anstatt in einer wissensbasierten Form zu implementieren. Technisch wäre dies zweifelslos zu bewerkstelligen, jedoch würden dabei die Vorteile einheitliche Strukturen mit Erklärungsfunktion, eine „elegante Lösung“ und eine bessere Wartungsoption aufgegeben. Das prozedurale Wissen ist in einer dem Menschen einfach zugänglichen Art der Repräsentation ausgeführt. Das Denken in WENN/DANN-Strukturen erschließt sich einfacher als (numerische) Algorithmen der Informatik, zugleich bietet die Architektur des Expertensystems stets eine Möglichkeit, eine Erklärung für die durchgeführte Schlussfolgerung zu erhalten. Die vielfach angewandte Optimierungsstrategie einer Suche nach Minima oder Maxima mit nur zwei Regeln aus einer theoretisch nahezu unbegrenzten Faktenmenge zeigt die Mächtigkeit und „Eleganz“ des angewandten Problemlösungsparadigma auf. Im konventionellen Bereich ist dies nur mit dem Mittel der Rekursion vergleichbar. Weiterhin ist auch die im *Knowledge-base Level* mehrfach definierte Suche bzw. Prüfung auf die Nichtexistenz eines Faktes mit dem *conditional element*-Grundkonstrukt `(not)` mit nur einer Regel umsetzbar. Folglich sagt aber auch die alleinige Angabe der Anzahl der Regeln eines Expertensystems noch nicht zwangsläufig etwas über seine Mächtigkeit und das Leistungsvermögen aus. Vollständigkeitshalber sei darauf verwiesen, dass das Regelwerk der Simulation 136 Regeln umfasst (vgl. Kap. A.4.3). Der weitaus bemerkenswerteste Vorteil besteht jedoch in der Trennung von Wissensbasis und Inferenzmaschine im Hinblick auf Wartung bzw. Wissenspflege, die sogar zur Laufzeit erfolgen kann. Dadurch können Änderungen an der Komponente der Wissensbasis durchgeführt werden, ohne dass hierdurch andere Programmkomponenten beeinflusst werden [Lug01]. Dies ermöglicht erst ein neues Spektrum an Dienstleistungen und Individualisierungsmöglichkeiten von Prozessführungen, wie es in vorhergehenden Diskussionskapiteln mehrfach thematisiert wurde.

Im Gegensatz zu dem gerade erörterten Themenpunkt fand die Erklärungs- und Dokumentationsfunktionalität bisher wenig Beachtung in der Diskussion. Im automatischen Betrieb einer Real-time Prozessführung wird diese Funktionalität auch nicht den Fokus auf

[67] A. a. O., S. 364

sich ziehen. Im Rahmen der Entwicklung, dem Testen und einer Dokumentationspflicht spielen die Ergebnisse aber eine Rolle. Über die mitgeführten Erklärungstexte ist der Schlussfolgerungsweg stets in knapper Form dokumentiert. Eine interaktive Nachfrage ist jedoch nicht möglich, da die „eingefrorenen Prozesszustände“ stets in einem Zug eingelesen werden und nicht wie in dialogorientierten Expertensystemen dem Nutzer jede Eingabeaufforderung bei Wunsch auch gleichzeitig begründet wird. Erst nach Abschluss eines gesamten Schlussfolgerungsprozesses kann der Nutzer die Erklärung in zusammengefasster Form in einem GUI-Ausschnitt lesen und bei Bedarf in den ebenso verfügbaren *Mind-Maps* den Lösungsweg nachverfolgen. Dennoch stellt sich die Frage, ob *Mind Maps* dafür geeignet sind. Als alleiniges Mittel darf dies bezweifelt werden, mit den Ergebnisbeschreibungen des *Problem Solving Level* und *Knowledge-base Level* sollte diese Form der Erklärung und Dokumentation ausreichen. Dem Wissensingenieur und dem Fachexperten stehen auf der Kommandoschnittstelle mit den Optionen der *Debug*-Ausgabe der Inferenz und der Simulation der Prozessumgebung umfassende Möglichkeiten zur Verfügung, den Vorgang im Detail online zu beobachten.

Der *Tool Level* bietet nur wenige Angriffspunkte für eine tiefergehende Diskussion. Die Vorgaben der vorhergehenden *Level* konnten mit der Auswahl der Expertensystem-Shell JESS direkt umgesetzt werden. Die beiden Schwachstellen liegen zum einen in technischer Hinsicht in der fehlenden Echtzeitfähigkeit der Inferenzmaschine zum anderen aus Entwicklersicht in dem Fehlen komfortabler Werkzeuge zur Erstellung und Wartung des Regelwerkes. Der zweite Punkt wird üblicherweise bei kommerziell angebotenen Werkzeugen wesentlich besser unterstützt und wird auch von den Anbietern als Rechtfertigung für die Preise der Entwicklerlizenzen vorgebracht. Für die Realisierung der Simulation entstanden durch das Fehlen keine Nachteile bei der Implementierung hinsichtlich der Leistungsfähigkeit. Die Nachteile liegen in einer erschwerten Wartung und Wissenspflege der Simulation.

Der technische Nachteil einer fehlenden echtzeitfähigen Inferenzmaschine kann durch den Einsatz eines RETE-Netzwerkes nur gemildert, aber nicht vollkommen kompensiert werden. Wenngleich WALTZ UND LLINAS (1990) [WL90] den RETE-Algorithmus als eine wichtige Möglichkeit anführen, Echtzeitfähigkeit in regelbasierten Systemen mittels paralleler Strukturen und Architekturen zu erreichen. Merkmale und Fähigkeiten, die eine echtzeitfähige Inferenzmaschine ausmachen, sind bei [Sri97][68] zusammengestellt. Ein wichtiges Konzept

[68] A. a. O., S. 664-667

zusätzlich zu einem *Real-time Scheduler* gründet in der Einführung und Behandlung von Zeitgültigkeitsintervallen von Variablen bzw. Fakten. HALL UND MCMULLEN (2004) [HM04] verweisen in diesem Zusammenhang noch auf die Fähigkeit von einigen Expertensystem-Shells, die zeitbegrenzte Suchläufe durchführen können, indem sie einen Inferenz-Prozess selbst beobachten und die Restlaufzeit bis zur Schlussfolgerung vorhersagen. JESS bietet bis auf die inhärente Parallelität keine der angesprochenen Merkmale, jedoch zeugen die gemessenen Zykluszeiten (vgl. Kap. 5.2.4.2) für die vorliegende Anwendung nicht von einem Problem mit dem Echtzeitverhalten. Auch die Alternative zur Integration des Faktors Zeit, die Verwendung von „eingefrorenen Prozesszuständen", wurde bereits erörtert.

Eine noch offene Frage ist, ob die Plattformunabhängigkeit von JESS aufgrund seiner Java-Basis durch einen negativen Einfluss auf die Leistungsfähigkeit erkauft wurde. Hierzu verweist FRIEDMAN-HILL (2003) [Fri03] darauf, dass „.... *Independent benchmarks have shown that JESS is significantly faster than many rule engines written in the „faster" C language. For example on many problems, JESS outperforms CLIPS by a factor of 20 or more on the same hardware. ...*"[69] Es bleibt dabei aber zu beachten, dass JESS Geschwindigkeit über einen größeren Speicherbedarf erreicht, was es abhängig macht von dem Verhalten und der Leistungsfähigkeit des *Java garbage collector*.

Zum Abschluss dieses Unterkapitels richtet sich das Hauptaugenmerk auf den Implementierungsschritt. Da auch im Ergebnisteil die Simulation aus der Perspektive der Bedienperson und GUI im Mittelpunkt stand, gilt dies ebenso für die Diskussion. Abgerundet wird die Einordnung mit wenigen Bemerkungen zu der Implementierung der „hinter" der GUI stehenden Komponenten Ablaufsteuerung, Regelwerk, simulierte Prozessumgebung und Datenbankanbindung, deren Dokumentation dem Anhang vollständigkeitshalber beigelegt wurde.

Grundsätzlich finden sich die Ergebnisse der Transformation des *Specification Level* hin zum *Tool Level* in der Implementierung wieder. Die zweite Teilzielsetzung wurde damit einer vollständigen Lösung zugeführt. Mit Blick auf die GUI wurde genau der konkrete Anwendungsfall abgebildet. Dies bedeutet jedoch auch, dass ein sehr statisches Layout mit abgestimmter Anzahl der wichtigen Elemente *Panel*, *Slider* und *Button* vorliegt. Ein flexibles Gegenmodell dazu wurde mit dem *„HTML Panel"* bereits angedeutet. Zumindest in dieser Komponente ist es möglich, eine dynamische Anpassung der GUI vorzunehmen, ohne JESS- oder Java-Entwicklungsschritte unternehmen zu müssen. Der Inhalt des „*Explanation*

[69] A. a. O., S. 38

(SUM_UP & DETAILS) Panel" würde sich geradezu für eine Realisierung in dem *„HTML Panel"* anbieten. Der entsprechende Fensterinhalt könnte zur Laufzeit in HTML- oder XML-Format erzeugt und dann direkt oder auf Anfrage zur Anzeige gebracht werden. Gleichzeitig liegt der gesamte Inhalt maschinenlesbar für Dokumentationszwecke vor.

Unabhängig von grundsätzlichen Designüberlegungen offenbart die Implementierung der GUI (vgl. Kap. A.4.5) deutliche Züge eines evolutionären bzw. *Prototyping* Vorgehens. Die teilweise willkürlich erscheinende Aufteilung in Elemente, die entweder in JESS oder in Java umgesetzt wurden, ist weniger konzeptioneller Art, sondern mehr der wachsenden Erfahrung des Entwicklers im Laufe des Entwicklungsprozesses zuzuschreiben. Während der Entwicklung wurde mit der GUI-Kernfunktionalität in JESS begonnen. Im Laufe der evolutionären Entwicklung zeigte sich jedoch, dass die zusätzlichen Anforderungen an die GUI zwar in der skriptartigen JESS-Umgebung implementierbar gewesen wären, aber eine Realisierung in Java effizienter zu bewerkstelligen war. Bei einem Neuentwurf sollte daher ein Entwurf komplett in Java und damit einhergehend auch die Nutzung eines *GUI-Builders* als unterstützendes Werkzeug erwogen werden.

Vor allem die Beschreibung der statischen Programmstruktur des JESS-Programms (vgl. Kap. A.4.1) lässt den Eindruck aufkommen, dass es sich um ein einfaches sequentiell zu durchlaufendes Programmkonstrukt handelt. Dies ist der Möglichkeit geschuldet, JESS wie eine Skriptsprache mit *late binding*-Fähigkeit nutzen zu können. Im Grunde besitzt die Implementierung einen ereignisorientierten Charakter mit einem integrierten regelbasierten Inferenzmechanismus, der ein Produktionssystem umsetzt, und hat daher mit einer rein sequentiellen Programmabarbeitung wenig gemeinsam. Eine weitere Besonderheit von JESS ist, dass es auch als *Rule Engine*, d.h. als Softwarekomponente für regelbasiertes Schlussfolgern, genutzt werden kann [Fri03]. Diese *Rule Engine* lässt sich damit in ein übergeordnetes Softwaresystem einbetten, sei es in den MSDF-Anteil einer Prozessführung für praktischen Einsatz oder als Komponente in einen Web-Service. Auch die für die Simulation der Prozessumgebung genutzte Softwarekomponenten-Technik der *Java Beans* und der *Shadow facts* in JESS (vgl. Kap. A.4.2) ermöglicht eine durchgängige Überführung in ein „*Hardware in the loop*"-System oder in eine Hard- und Software-Implementierung für praktische Versuche. Rein auf die Simulation bezogen gestattet diese Struktur eine einfache Skalierbarkeit hinsichtlich der Anzahl aber auch des Detailgrades der Prozessumgebungsnachbildung. In Anbetracht des objekt-orientierten Java als Basis von JESS ist es durchaus überlegenswert, ob nicht eine Datenbank basierend auf demselben Paradigma eine bessere Wahl als die genutzte relationale Datenbank (vgl. Kap. A.4.4)

gewesen wäre. Bei der Implementierung stand jedoch die prinzipielle Anbindung an ein Datenbanksystem im Vordergrund, das zumindest ein rasterbasiertes Verfahren unterstützt. Bei einer vertieften Betrachtung und Neuimplementierung sind nicht nur die beiden angesprochenen Kriterien zu beachten, sondern auch die Entwicklungen im Bereich der Datenbanken für die Prozessdatendokumentation und der GIS (vgl. Kap. 2.3.2.5) zu berücksichtigen.

6.3.3 Systemtest (Validation)

Bereits im Stand der Technik wurde die allgemeine Schwierigkeit mit der Verifikation bei der Entwicklung eines Expertensystems thematisiert. Daher lag der Schwerpunkt der Testaktivitäten auf der Validation entsprechend der Kategorien Leistungsfähigkeit bezüglich der Problemlösung und informationstechnische Eignung und Leistung. Auch die kritisch zu beurteilende Rollenannahme des Autors als Entwickler und zugleich Tester wurde in der Diskussion des Entwicklungsprozesses bereits aufgegriffen (vgl. Kap. 6.3.1).

Bevor die Validation einer näheren Betrachtung unterzogen wird, gilt es, einem Aspekt bei der nicht vollständig zu vernachlässigenden Verifikation Beachtung zu schenken. Bei der Verifikation der (für zumindest einen Iterationszyklus) eindeutig spezifizierten Mensch-Maschine-Schnittstelle kam kein automatisiertes Testverfahren zum Einsatz. Dies muss noch nichts über die Qualität der Implementierung aussagen, führte jedoch zu keiner lückenlosen Testabdeckung. Zugleich wirkte sich dieser Schwachpunkt auch direkt auf die Möglichkeiten der Testautomatisierung bei der Validation aus. Die Endergebnisse der Eingabeaktivitäten der Bedienperson, d.h. z.B. Positionsangaben oder Modus-Auswahl, wären zwar simulierbar gewesen, aber die „*Slider*“- und *Scroll*-Bewegungen oder die Nutzung des „*HTML-Panel*“ mit ihrem Einfluss auf die IT-Leistungsfähigkeit konnten nur manuell getestet werden. Einfach zu lösen ist diese Problematik jedoch nicht. FRIEDMAN-HILL (2003) [Fri03] verweist auf die Schwierigkeit, ein entsprechendes Testwerkzeug zu finden, da sie meist entweder schwierig handhabbar, teuer oder fragil sind. Fragil bedeutet in diesem Zusammenhang, dass die Tests stets neu codiert werden müssen, falls die GUI-Applikation modifiziert wird. Statt eines kommerziellen Tools schlägt er als Kompromisslösung die Implementierung eigener automatisierter Tests basierend auf der Java-Klasse *java.awt.Robot* und den Skriptsprache-Fähigkeiten von JESS vor. Da jedoch die Validation der ersten Prototypen mit manuellen Tests zur Problemlösungsfähigkeit begonnen wurde und die Entwickler-Ressourcen sehr begrenzt waren, wurde davon abgesehen, ein Framework für automatisierte GUI-Tests zu erstellen.

Im Ergebnisteil zur Validation der Leistungsfähigkeit bezüglich der Problemlösung wurden Tests in Sachen Genauigkeit, Präzision, Breite und Tiefe des Lösungsspektrums und Wiederholbarkeit beschrieben. Dabei kamen manuelle und aufgrund der fehlenden GUI-Testautomatisierung nur teilautomatisierte Verfahren zum Einsatz. Nach einer Einstufung von KURBEL (1989) [Kur89][70] verfolgten die manuellen Tests dabei im Wesentlichen die qualitativen Validationsmethoden „*Face*"-Validation und Sensitivitätsvalidation. Die teilautomatisierten Verfahren gehören ebenfalls zur stichprobenartigen „*Face*"-Validation oder zur Vorhersage-Validation, bei der das System gegen Testfälle geprüft wird, bei denen das richtige Ergebnis bekannt ist.

Grundsätzlich lässt sich der Simulation bescheinigen, die gestellten Erwartungen an die Problemlösungsfähigkeit in vollem Maße zu erfüllen. Die Wissensrepräsentationsformen und der Inferenzmechanismus erwiesen sich als geeignet. Der Detaillierungsgrad des Wissens ist ausreichend. Die Qualität des eingesetzten pflanzenbaulichen Wissens ist jedoch sicherlich ein Bereich, der zu intensiver Diskussion herausfordert. In dieser Arbeit wurde dies bereits in der Diskussion des *Knowledge-based Level* oberflächlich erörtert, die vertiefte Einordnung der Methode und der Ergebnisse ist Teil der Dissertation von WEIGERT (2006) [Wei06]. In diesem Gesamtzusammenhang eingeordnet, ließe sich das Testen der Problemlösung für diesen wichtigen Teil des prozeduralen Wissens eher der Verifikation zuordnen, während es als Validation nur im Zusammenspiel mit der Arbeit von Weigert gesehen werden kann.

Im Hinblick auf die Wiederholbarkeit sowie Breite und Tiefe des Lösungsspektrums überrascht das positive Testergebnis nicht besonders, da bereits auf *Task Level*-Ebene eine Lösung angestrebt wurde, die zwar wissensbasiert ist, aber mit einem datengetriebenen Ansatz in einem begrenzten Lösungsraum eine weitestgehend deterministische Lösung finden sollte. Eine vollständige Testabdeckung wurde jedoch bei der exponentiellen Anzahl an Kombinationsmöglichkeiten der Prozesseingangsvariablen und -zwischenzuständen nicht durchgeführt. Dies relativiert auch das Fehlen einer automatischen GUI-Testumgebung und die nur stichprobenartige Überprüfung des Problemlösungsweges bei den teilautomatisierten Testläufen. Es stellt keine ideale Lösung dar, steht jedoch im Einklang mit der Testphilosophie des JESS-Entwicklers FRIEDMAN-HILL (2003) [Fri03]: „*.... The testing books often forget to tell you that some tests, although perhaps not as good as all possible tests, are infinitely better than no tests. Don't be discouraged by the sense that you can't completely test*

[70] A. a. O., S. 193

a rule-based application. Remember that some testing is a lot better than none at all. …“[71]. Es ist anzunehmen, dass dieser pragmatische Testansatz mit dem „*Agile Software Development*“ zusammenpasst, jedoch mit dem speziell auf die funktionale Sicherheit abzielenden V-Modell in einem spannungsreichen Verhältnis steht. Aber selbst bei einer vollständigen Testabdeckung bleibt bei der Validation eines wissensbasierten Systems die Herausforderung bestehen, dass (Fach-)Experten oft unterschiedliche Meinungen vertreten, d.h. ein Ergebnis, das der eine Experte als korrekt ansieht, kann nach Einschätzung eines anderen falsch oder unbefriedigend sein [Kur89]. So ist zu erwarten, dass Fachexperten entsprechend unterschiedlicher „Düngesystem- bzw. Reflexionsmodell-Schulen“ (z.B. Expert-N, Hermes, TUM-Weihenstephan, PROSPECT, Kieler Verfahren [AOMSSW06], [Hee07]) zu verschiedenartigen Regeldefinitionen kommen werden. Das aufgezeigte Spannungsfeld wird vermutlich eine hemmende Wirkung auf eine schnelle und umfangreiche Anwendung von wissensbasierten Systemlösungen im Bereich der Landmaschinenindustrie haben, die sich derzeit mit der Einführung von Entwicklungsprozessen zur Gewährleistung der funktionalen Sicherheit konfrontiert sieht (vgl. Kap. 2.5).

Da die Simulation die Eignung und Machbarkeit einer wissensbasierten *JDL Processing 2 Level*-Lösung für eine Real-time Prozessführung demonstrieren und untersuchbar machen sollte, kam der Validation der informationstechnischen Eignung und Leistung eine herausragende Bedeutung zu. Die entscheidende Größe zur Beurteilung der Echtzeitfähigkeit ist die ermittelte Durchlaufzeit für einen kompletten Schlussfolgerungsprozess. Da der MSDF-Prozess sich mit weiteren Rechenprozessen des Gesamtsystems die Hardware- und Softwareressourcen teilen muss, sind die CPU-Auslastung und der Speicherbedarf ebenfalls wichtige Größen für die Systemintegration. Die durchgeführten Tests erlaubten die Ermittlung der Messgrößen und ließen Trends erkennen, auch wenn nur wenige charakterisierende Größen für die Unterscheidung der vier Testplattformen herangezogen wurden. Unberücksichtigt blieben z.B. Parameter wie *Front Side Bus*-Geschwindigkeit, *Level* 1 oder 2 *Cache*, Grafikkartenarchitektur oder Festplattenschnittstelle, die sicherlich auch einen Einfluss auf die zu messenden Größen haben, aber die allgemeinen Trends nicht ändern. Die Ergebnisse der leistungsschwächsten, mittlerweile sehr veralteten Testplattform II (vgl. Kap. 5.2.4.2) geben einen guten Anhaltspunkt für die bei der reinen Rechenleistung stets zurückbleibenden Agrarelektronik verglichen zu *Consumer*- und Büro-IT-Technik. Dieser Nachteil wird durch die notwendige Robustheit gegenüber den sehr anspruchsvollen

[71] A. a. O., S. 177

Umgebungsbedingungen aufgewogen. Eine leistungsmäßige Einordnung folgt später in diesem Kapitel.

Für die Testplattform II ist deutlich auszumachen, dass die CPU-Auslastung von 2 % für den Schlussfolgerungsprozess innerhalb eines Gesamtsystems nicht besonders ins Gewicht fällt. Die Erhöhung der CPU-Last steht eindeutig in Verbindung mit den Leistungsmerkmalen der Mensch-Maschine-Schnittstelle und steigt dabei im Spektrum Konsolen-Schnittstelle, GUI und schlussendlich Nutzung des „*HTML Panel*“ auf bis zu 30 % an. Vor dem Hintergrund dieser Ergebnisse sollte der Vorschlag aus der Diskussion der GUI, das „*Explanation (SUM_UP & DETAILS) Panel"* mit einem dynamischen Design in das *„HTML Panel”* zu überführen, nochmals hinterfragt werden. Abhängig vom Verhalten des Nutzers würde diese Designänderung zu einer erheblichen Mehrbelastung des Systems führen oder aber auch eine Entlastung bewirken können. Bedient der Nutzer stets das *„HTML Panel”,* um die aktuellen Erklärungstexte beobachten zu können, würde dies mit einer erhöhten mittlere CPU-Auslastung erkauft. Im gegenteiligen Fall, in dem er nur von Zeit zu Zeit oder in einer Pause die Erklärungen überprüft, könnte sogar eine niedrigere mittlere CPU-Last in der Größenordnung der minimalen Last mit GUI-Anzeige, d.h. 7 %, erreicht werden. Das System würde davon profitieren, dass die Erklärungstexte nicht mehr im „*Explanation (SUM_UP & DETAILS) Panel"* aktualisiert würden. Da die Erklärungstexte nicht mehr direkt zur Anzeige gebracht sondern in eine Datei geschrieben bzw. ausgelagert würden, brächte dies den Vorteil mit sich, dass die Speicherauslastung zugleich sinken würde. Die Messungen zeigten, dass das Anwachsen des Speicherbedarfs über die 9,5 MB hinaus hauptsächlich durch die Puffermechanismen des „*Explanation (SUM_UP & DETAILS) Panel"* bedingt waren.

In der *Consumer*- und Büro-IT-Technik sind die gemessenen Speicherauslastungen bis zum Maximum der eingestellten *heap size*-Größe des *Java Garbage Collector* von 64 MB unproblematisch. In der Agrarelektronik oder dem Bereich der Automobilelektronik (*automotive*) wird jede einzelne Systemkomponente kostenmäßig hinterfragt und die Hauptspeicherauslegung ist in der Regel wesentlich knapper bemessen. Neben der aufgezeigten teilweisen Entkopplung der GUI besteht auch die Möglichkeit, eine Kosten/Nutzen-Abwägung zwischen der Geschwindigkeit der JESS-Inferenzmaschine und den beinflussenden Parametern des *Java garbage collector*, der *heap size*- und der *object nursery size*-Größe, vorzunehmen [Fri03].

Zusammenfassend betrachtet, liegen die gemessene CPU-Auslastung und der optimierbare Speicherbedarf für den reinen MSDF-Algorithmus, d.h. ohne eine GUI bzw. Mensch-Maschine-Schnittstelle, durchaus in einem akzeptablen Bereich in Bezug auf eine Integration

in ein Gesamtsystem. Die Mensch-Maschine-Schnittstelle ist ohnehin in die des Gesamtsystems einzufügen und eine aktive „Debug“-Konsolenschnittstelle ist nur im Entwicklermodus von Interesse.

Für die Laufzeitermittlungen wurde die Testkonfiguration II (vgl. Tab. 5.3) ausgewählt, die einerseits den *worst case* in Sachen CPU-Auslastung und Speicherbedarf erzeugt und anderseits für den Tester mittels der GUI die nötigen Überwachungs- und Eingabeoptionen bereit hält. Die vertiefte Diskussion basiert wiederum auf der leistungsschwächsten Testplattform II. Auffällig an den Messergebnissen ist die unsystematisch erscheinende Klasseneinteilung, wie sie aus der Abbildung 5.9 ersichtlich ist. Die Klassen mit einer Auflösung von 10 ms legen den Verdacht nahe, dass das Messverfahren die mögliche Standardauflösung der Java-Methode „*currentTimeMillis()*“[72] von 1 ms nicht unterstützen kann. Andererseits liegen von den Klassen 11 ms, 21 ms, 31 ms und 41 ms 36 Messwerte vor, die dieser Annahme entgegenstehen. Sonderbar ist jedoch, dass im Einer-Bereich der Zahlen nur die Zahl eins auftritt. Da keine tiefere Untersuchung dieses Phänomens stattgefunden hat, ist eine Erklärung schwierig. Weil die Anzahl der betroffenen Messergebnisse vergleichsweise gering ist, d.h. im diskutierten Fall nur 0,7 %, wird dem keine größere Bedeutung beigemessen und im Folgenden näherungsweise von einer Auflösungsfähigkeit von 10 ms ausgegangen. Für die Einordnung in Bezug auf Echtzeitfähigkeit bedeutet dies, dass mit einem gerundeten Mittelwert von 10 ms und einem Maximalwert von 90 ms für einen Schlussfolgerungsprozess auf der schwächsten Plattform gerechnet wird. Für den Datenbankzugriff wird mit einem Mittelwert von 30 ms und einem Maximalwert von ebenfalls 90 ms kalkuliert.

Im Mittel wird somit für einen Zyklus, bei dem Kartenwerte aus der lokal vorgehaltenen Datenbank gelesen werden, die Inferenz durchgeführt wird und Erklärungstexte aktualisiert werden, die Summe aus 10 ms und 30 ms, folglich 40 ms, benötigt. Im Falle, dass die ungünstigsten Bedingungen herrschen, berechnet sich die Zeitdauer aus zweimal 90 ms, d.h. 180 ms. Werden diese Werte in Bezug zu dem Ergebnis der Gleichung 4.3 (vgl. Kap. 4.1) in der Größenordnung von 920 ms gesetzt, sind im Mittel Wiederholraten (*update rate*) von 23 Hz und im *worst case* immer noch 5 Hz möglich. Mit diesen Werten sind die Empfehlungen des ISOBUS-Normvorschlag für den Sensor-Ansatz (vgl. [ISO11783-10DAM1] - Kapitel

[72] Auszug aus der Java-API-Dokumentation für die Methode „public static long currentTimeMillis()”:
“*…. returns the current time in milliseconds. Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds….*”

6.6.3 „*Real time sensor based control*“) für die Wiederholrate einer Sollwertquelle von 1 Hz bei unverändertem Sollwert und von maximal 5 Hz während einer Wertänderung bereits auf der schwächsten Testplattform erreichbar.

Gerade für den *worst case*-Fall hat bisher keinerlei Optimierung des MSDF-Algorithmus sowie des Messverfahrens stattgefunden. Der Vergleich aller Fälle, in denen die Regel <N-20> zur Anwendung kam, die für die einmalig gemessenen 90 ms Durchlaufzeit steht, legt den Schluss nahe, dass die Zeitunterschiede weniger in den Inferenz-Vorgängen des RETE-Netzwerkes, sondern mehr im Einfluss der GUI oder der *Java Virtual Machine* zu suchen sind. Eine bessere Einsicht in die exakte Zeitdauer des Inferenz-Prozesses könnte mittels eines besser aufgeschlüsselten Zeitmessverfahrens gewonnen werden, das zugleich auch noch die Auswirkung unterschiedlicher Threads mitberücksichtigt. Der Datenbank-Leseprozess sollte dahingehend optimiert werden, nicht jeden einzelnen Wert mit einer eigenen Anfrage sequentiell einzulesen, sondern mit einem Zugriff alle benötigten *Precision Farming*-Kartenwerte abzufragen. In dieser Maßnahme und der Entkoppelung der GUI von der reinen Inferenz-Aktivität liegt ein Optimierungspotential, das ausreichend Spielraum für ein umfangreicheres Regelwerk und weitere Prozesseingangswerte bieten sollte, ohne dafür eine höhere Hardwareleistungsfähigkeit zu benötigen.

Der extrem negative Einfluss auf die Laufzeiten durch Nutzung der Konsolen- bzw. Kommandozeilenschnittstelle ist unerheblich, da er nur für den Wissensingenieur und von Zeit zu Zeit für den Experten von Interesse ist, das System dann aber fast ausschließlich im zeitunkritischen Einzelschrittmodus betrieben wird.

Abschließend soll eine Gegenüberstellung der Rechnerleistung der getesteten Rechnerplattformen und von Varianten einer aktuellen *Embedded*-Prozessor-Produktlinie die durchgeführten Messungen aus einem weiteren Blickwinkel einschätzen helfen. Als Referenzplattform für einen *Embedded*-Prozessor mit X86-Architektur, der einen lüfterlosen Betrieb erlaubt und eine geringe Leistungsaufnahme aufweist, dient die „Intel® Atom™ Processor“-Baureihe des Chipherstellers Intel® Corporation [Int10], [Int11]. Für die Gegenüberstellung (vgl. Tab. 6.1) werden die Angaben des Internet-Auftritts zu der CPU-Benchmark-Software PassMark® Software herangezogen [Pas11]. Die Aufstellung der Benchmark-Ergebnisse nach dem „PassMark-CPU-Mark“-Bewertungsindex wird von PassMark® Software folgendermaßen beschrieben [Pass11]: „*This chart comparing CPU benchmarks is made using thousands of PerformanceTest benchmark results and is updated daily. The CPU you selected has been found and highlighted from amongst the high, medium*

and low end CPU charts." [73]

Für drei der vier Testrechnerplattformen konnte in der Aufstellung keine exakt passende Konfiguration gefunden werden. Daher wurde stets der Typus mit der am nächsten liegenden Taktrate derselben CPU-Variante aufgeführt. Aufmerksamkeit verdient die Tatsache, dass die schwächste Testplattform II eine anzunehmende 15 %-ige CPU-Minderleistung[74] gegenüber dem schwächsten Intel® Atom™-Prozessor gemessen nach dem PassMark® CPU-Benchmark aufweist. Für die Testplattform III stimmen die CPU-Leistungswerte des höher getakteten Mobile Intel® Celeron® mit denen des schwächsten Intel® Atom™-Prozessor genau überein.

Tabelle 6.1: Gegenüberstellung von PassMark® CPU-Benchmark-Ergebnissen

Rang	CPU	CPU-Taktrate	PassMark–CPU Mark	Testplattform (Notebook)
1	Intel® Pentium® M	1,86 GHz	-	IV (Samsung X20)
2	Intel® Pentium® M	1,73 GHz	447	-
3	Intel® Atom™ N450	1,66 GHz	319	-
4	Intel® Pentium® III-M	1,13 GHz	262	-
5	Intel® Pentium® III-M	1,06 GHz	-	I (Kontron ReVolution)
6	Intel® Pentium® III-M	1,00 GHz	242	-
7	Mobile Intel® Celeron®	1,50 GHz	186	III (Toshiba Satellite 1110)
8	Intel® Atom™ Z510	1,10 GHz	186	-
9	Mobile Intel® Celeron®	1,20 GHz	170	-
10	Mobile Intel® Celeron®	0,79 GHz	-	II (Sony PCG-R600MX)

In der Regel kommt nicht dieser schwächste Prozessor, sondern der um 50 % schneller getaktete Intel® Atom™ N450-Prozessor [Int11] in *Embedded*-Steuerungen und in *Netbooks* zum Einsatz, der eine etwa 71 % bessere CPU-Leistung als die Testplattform III und nahezu das Zweifache der Leistung der Rechnerplattform II besitzt. Dieser schnellere Intel® Atom™-Prozessor ist auch um etwa 25 % leistungsfähiger als der Intel® Pentium® III-M der Testplattform I, aber um mehr als 40 % schwächer als der Intel® Pentium® M des Testrechners IV. Der Leistungsvergleich unterschiedlicher CPU-Varianten nur anhand eines einzigen CPU-Benchmarks ohne aufgezeigte Relation der Testmethoden zu den leistungsbestimmenden Simulationselementen ist vom wissenschaftlichen Standpunkt kritisch

[73] A. a. O., S. 1

[74] Der Verlauf der Benchmark-Kenngröße für dieselben CPU-Varianten mit 1,5 GHz und 1,2 GHz ist nicht linear, daher wurde für die 0,8 GHz Variante eine vorsichtige Schätzung im Bereich von 15 % CPU-Minderleistung angenommen.

zu bewerten, als Orientierungshilfe im praktischen Einsatz hilft er jedoch Trends zu erkennen, erst recht wenn die gemessenen Unterschiede so drastisch zu Tage treten. Auf jeden Fall lässt sich festhalten, dass die Wahl der sehr leistungsschwachen Testplattform II eine gute Richtschnur abgibt, um damit die *worst case* Situation auch für aktuell verfügbare *Embedded*-CPU-Hardware abschätzen zu können. Mit einem auch zukünftig zu erwartenden Zuwachs der Leistungsfähigkeit von *Embedded*-Rechnerplattformen (vgl. [Aue11], [Eck12]) ist anzunehmen, dass die Prozessrechnertechnik kein Hindernis zu Integration von intelligenten bzw. wissensbasierten Optimierungsansätzen für die Real-time Prozessführung in sensorgestützten Düngesystemen darstellen wird.

Die Diskussion und Einordnung des Themas Testen rundet ein kurzer Blick auf die informationstechnischen Leistungs- und Eignungskriterien Portierbarkeit und Erweiterbarkeit ab. Bereits im Ergebnisteil wurde vermerkt, dass die Untersuchung auf Portierbarkeit sich auf die vier unterschiedlichen Rechnerplattformen beschränkte. Aufgrund der Tatsache, dass im *Tool Level* die Entscheidung für eine Implementierung in JESS und damit auf Basis des plattformunabhängigen Java gefallen ist, kann von einer Portierbarkeit auf eine breite Auswahl von Hardware- und Betriebssystemumgebungen ausgegangen werden. Mögliche Abstriche sind bei der Portierung der MySQL-Datenbankimplementierung zu machen, die jedoch keine bindende Voraussetzung für die Realisierung der Simulation ist. Allgemein bleibt zum Thema Portierung festzuhalten, dass der Kern der Implementierung, die Ergebnisse der Konzeption und Formalisierung, sich mit einer breiten Palette an Programmiersprachen und Werkzeugen in eine Hard- und Softwareimplementierung umsetzen lassen (vgl. Kap. 5.1.3.1) (vgl. auch [Kur89]). Die entscheidende Frage dabei ist die Effektivität und die zur Verfügung stehende Ressourcenausstattung. Die grundsätzliche Erweiterungsfähigkeit der Simulation wurde aufgrund der dafür vorteilhaften Trennung von Wissensbasis und Inferenzmaschine im Rahmen der Diskussion bereits mehrfach positiv bewertet.

6.4 Herausforderungen und Grenzen einer MSDF

In der vorliegenden Arbeit wurde das Thema MSDF sowohl aus einer sehr allgemeinen als auch aus einer in hohem Maße spezifischen agrarsystemtechnischen Betrachtungsweise analysiert und letztendlich als Simulation implementiert. Dabei wurde festgestellt, dass die Agrarsystemtechnik in der Vergangenheit nicht unter den treibenden Kräften der MSDF-Disziplin war, aber deren Vorzüge mittlerweile erkannt hat und zum integralen Bestandteil von Forschungsprojekten und Produktlösungen macht. Die Motivation für den Einsatz von

MSDF und deren Vorzüge wurden im Stand der Technik bereits ausgeführt (vgl. Kap. 2.1.1). Auch die analysierte und entwickelte Real-time Prozessführung in sensorgestützten Düngesystemen zählt zu den positiven Beispielen, die vom Einsatz dieser Technik profitieren. Dennoch stoßen auch MSDF-Ansätze an Grenzen. Daher schließt das Diskussionskapitel mit einem Blick auf die allgemeinen Grenzen und Herausforderungen und stellt die kritische Frage, welche Erkenntnisse sich daraus für den Einsatz in der Agrarsystemtechnik gewinnen lassen. Der Vorteil der Position eines *follower* ist, dass auf die Erkenntnisse und Erfahrungen der *first mover* zurückgegriffen werden kann. Die Anwendungsfelder können sehr verschieden sein, die zur Verfügung stehenden Methoden und Algorithmen sind es nicht. Daher besitzen allgemeine Aussagen auch für die Agrarsystemtechnik Gültigkeit.

So wurde in dieser Arbeit bereits mehrfach darauf hingewiesen, dass eine MSDF-Lösung kein eigenständiges System darstellt, sondern nur als Teil einer größeren Systemlösung eine Berechtigung hat und zum Einsatz kommt. Diese Überlegung sollte stets an erster Stelle stehen, falls der Einsatz dieser Technik als Lösungsweg bzw. -alternative erwogen wird. Im Folgenden werden die Ausführungen von HALL UND STEINBERG (2001) [HS01] zu den „*Dirty Secrets in Multisensor Data Fusion*“ herangezogen, um die Fallstricke (*pitfalls*) aufzuzeigen, und dann in den agrarsystemtechnischen Kontext einzuordnen. Dabei wird stets zuerst die Originalaussage wörtlich zitiert und anschließend knapp kommentiert.

„…. There is no substitute for a good sensor. No amount of data fusion can substitute for a single, accurate sensor that measures the phenomena that you want to observe. …“[75] [HS01]

Eine einfach gehaltene Aussage, aber mit kaum zu überschätzender Wichtigkeit. Eine Aussage die für *mature*, *pivotal* oder *emerging* Sensoren gleichermaßen gilt. Bezogen auf das Anwendungsbeispiel dieser Arbeit hat der Online-Sensor „Pflanze“ eine herausragende Bedeutung und die Technik der angebotenen Sensoren entstammt jahrelanger internationaler Forschungsarbeit zu Vegetationsindizes und deren Korrelation zu Pflanzeneigenschaften. Der Mangel eines vergleichbaren Online-Sensors zur Bestimmung der Bodenfeuchte ist derzeit auch durch einen *Data Fusion*-Ansatz nicht ersetzbar. Die Näherungslösung über ein WSN, bestehend aus stationären Sensoren, wird auf der Skalenebene eines Feldes gute Dienste leisten, aber auf kleinräumiger Ebene bestenfalls eine grobe Schätzung anbieten können. Die Qualität der Positionsbestimmung und des Ertragsmess-Sensors ist entscheidend für die Ertragsermittlung und damit für die Qualität der zugehörigen Ertragskarten. Hierzu liegen entsprechend viele Untersuchungen vor, die die Grenzen aufzeigen. Aber selbst der beste

[75] A. a. O., S. 9

Sensor ist von geringem Nutzen, falls er zum Einsatzzeitpunkt nur eingeschränkt verfügbar ist, wie z.B. ein durch eine Wolkendecke abgeschirmter Fernerkundungsensor.

„…. Sensor fusion can result in poor performance if incorrect information about sensor performance is used. …"[76] [HS01]

Falsche Annahmen über die Rahmenbedingungen von Eingangsinformationen eines Fusionsalgorithmus führen zwangsläufig zu einem zweifelhaften Ergebnis. So ist es z.B. kritisch, ohne Kenntnis des Geschwindigkeitssensors von einem schlupffreien Geschwindigkeitswert auszugehen. Der bedeutende Aspekt der Berücksichtigung der Sensoreigenschaften wurde erkannt und von der agrarsystemtechnischen Forschung aufgegriffen [BKB11].

„…. There is no such thing as a magic or golden data fusion algorithm. Despite claims to the contrary, there is no perfect algorithm that is optimal under all conditions.…"[77] [HS01]

Gerade diese Arbeit hat es sich zur Aufgabe gemacht, nicht von einem allgemein gültigen *Data Fusion*-Algorithmus auszugehen, sondern die Randbedingungen zu berücksichtigen und eine darauf angepasste Antwort zu finden.

„…. There will never be enough training data. …"[78] [HS01]

Diese Herausforderung ist ein starkes Argument für modellbasierte Ansätze, falls alternative Lösungsansätze existieren. Dies würde zumindest den Datenmangel weg aus der Entwurfsphase in die Phasen der Validation und Verifikation verlagern. Im Kapitel 5.2.2.4 wurde auf die Anstrengungen der IKB-Dürnast-Projektpartner hingewiesen, um das pflanzenbauliche Wissen mittels *Data Mining* gewinnen zu können. Dennoch sind die Datensätze zu klein, um eine allgemeine Gültigkeit für den Standort annehmen zu können. In der verfahrenstechnischen Einordnung wurde die Option eines langjährigen On-farm-Research-Ansatzes ins Spiel gebracht, um zumindest eine betriebsspezifische Lösung zu finden.

„…. Quantifying the value of a data fusion system is difficult. A challenge in data fusion is to quantify the utility of the system at a mission level. Although measures of performance can be obtained for sensors or processing algorithms, measures of mission effectiveness are difficult to define. …"[79] [HS01]

Mit dieser Aussage schließt sich der Kreis und kommt zur grundsätzlichen Feststellung

[76] A. a. O., S. 10

[77] A. a. O., S. 10

[78] A. a. O., S. 10

[79] A. a. O., S. 10

zurück. Die Frage steht im Raum, was und wie gut trägt der MSDF-Ansatz zur erfolgreichen Aufgabenbewältigung des Gesamtsystems bei. Für das vorgestellte Anwendungsbeispiel ist dies die interdisziplinäre Fragestellung nach einer ökonomisch-ökologischen Bewertung, die im Mittelpunkt des IKB-Dürnast-Teilprojekt 13 [Gan05] (vgl. Kap. 5.2.1) stand. Die Herausforderung ist zugleich Forderung nach einer multidisziplinären Zusammenarbeit, die im Kontext dieser Arbeit ihren Ankerpunkt im Funktionalen Modell haben sollte.

7 Schlussfolgerungen und Ausblick

Alle drei gesteckten Teilziele dieser Arbeit wurden einer Lösung zugeführt. So wurde eine durchgängige Analyse- und Entwurfsmethodik für MSDF in einem landwirtschaftlichen BUS-System für die Real-time Prozessführung in sensorgestützten Düngesystemen allgemeingültig erarbeitet und die spezifische Anwendbarkeit bis hin zu einer Implementierung als Software-Simulation gezeigt. Damit einhergehend wurde auch ein Vorschlag für eine eindeutige Terminologie für MSDF im Kontext der Agrarsystemtechnik geschaffen und kann somit zur besseren Verständigung zwischen unterschiedlichen Anwendergruppen und Akteuren im Agrarbereich beitragen. Das erarbeitete Vorgehensmodell leistet für die einzelnen Anwendergruppen verschiedene Dienste und bietet unterschiedliche Perspektiven. Die Analyse- und Entwurfsmethode gibt, wie der Name schon aussagt, sowohl dem mit Systementwurf und -entwicklung betrauten Personenkreis als auch dem analytisch tätigen Personenkreis, wie Systemanalysator, Gutachter oder Tester einen Leitfaden an die Hand. Aus dem Systementwurfs-Blickwinkel wurde ein geradliniger effizienter Weg vorgeschlagen, um ein den Anforderungen entsprechendes Systemdesign oder die möglichen Alternativen ableiten und umsetzen zu können. Aus der Perspektive der Systemanalyse liefert dieses MSDF-Framework hilfreiche Werkzeuge zur Beurteilung von am Mark eingeführten Lösungen, von Vorschlägen für zukünftige Systemansätze sowie der gezielten Testentwicklung und -durchführung. Zwei wichtige Grundsätze wurden offenkundig:

- MSDF steht nicht für sich alleine, sie ist immer nur Teil eines Gesamtsystems oder ein gewählter Ansatz. Folglich hat sich das MSDF-Framework in den Entwicklungsprozess des Gesamtsystems einzufügen, wobei die Anwendung von Abstraktion hilft.
- Es gibt nicht den einen alles umfassenden und lösenden MSDF-Algorithmus, gewissermaßen den „*golden data fusion algorithm*“. Diese Tatsache steht für die ursächliche Begründung der Notwendigkeit einer Entwurfs- und Analysemethode, wie sie in dieser Arbeit entwickelt und untersucht wurde.

Diese Arbeit und das MSDF-Framework legen das Hauptaugenmerk auf eine Eingrößen-Prozesssteuerung. Daher drängt sich die Ausweitung der Forschung in Richtung einer Mehrgrößen-Prozesssteuerung direkt auf. Ebenso liegt die Hypothese nahe, dass in der Agrarsystemtechnik eine prinzipielle Anwendbarkeit des Vorgehensmodells nicht auf Real-time Prozessführungen beschränkt ist, da dem Funktionalen Modell und dem Prozessmodell ein hoher Abstraktionsgrad zugrunde liegt sowie auf Modellelemente zurückgegriffen wurde, die sich bereits in einem weiten Feld von *Data Fusion*-Anwendungen bewährt haben. Beide

Richtungen zur Ausweitung des Anwendungsbereiches sind von grundlagenorientiertem Interesse und laden zu einer vertieften Analyse ein.

Die Diskussion und Einordnung hat offengelegt, für welche Aufgabengebiete der Entwurfs- und Analysemethode weiterer genereller Forschungsbedarf besteht. Zwei Problemstellungen ragten dabei besonders hervor:

- Dies ist zum einen der Bruch zwischen konventionellen Entwicklungsprozessen und dem evolutionären Vorgehen zur Entwicklung eines wissensbasierten Systemanteils
- Zum anderen ist es die Forderung nach Funktionaler Sicherheit, die spätestens auf Ebene der Systemarchitektur als Rahmenbedingung ins Spiel kommt. Bei der Funktionalen Sicherheit zeichnet sich gerade im Zusammenhang mit wissensbasierten Lösungen ein schwieriges Spannungsfeld ab. Dies erfordert eine intensive theoretische Durchdringung des Themas, wie das weite Feld der MSDF mit den Anforderungen der Funktionalen Sicherheit in Einklang gebracht werden kann.

Während sich diese neuen Forschungsansätze noch im Feld der MSDF bewegen, ermuntert und verlangt die Entwicklung und Untersuchung einer Informations- und Kommunikationstechnik-Infrastruktur für ein öffentliches und privates Wissensmanagement im Agrarbereich (vgl. Kap. 2.3.2.5) das Thema der *Information Fusion* zum Gegenstand einer wissenschaftlichen Untersuchung zu machen. Auf gemeinsame Schnittmengen mit dieser Arbeit sollte geprüft werden, um letztendlich zu einem umfassenderen Ansatz oder zu einer klaren Abgrenzung zu gelangen.

Neue Fragestellungen und Forschungsthemen ergeben sich jedoch nicht nur im allgemeinen Bereich, sondern sind auch in spezifischen Feldern dieser Arbeit zu finden. So kommen die Möglichkeiten und die Grenzen der Integration der Methoden „*Agile Software Development*“ und „*Systems Architecturing*“ in die Phasen Systemarchitektur und Implementierung des „*top down“*-ausgerichteten Vorgehensmodells durchaus für eine nähere Evaluierung in Betracht. Dabei wäre es lohnenswert, eine Sammlung von typischen agrarsystemtechnischen *use cases* für MSDF zusammenzustellen, die zutreffenden Problemlösungsparadigma abzuleiten und mit Algorithmen zu hinterlegen oder falls eine Systemlösung bereits existiert, die angewandten Algorithmen einer Problemlösungsform zuzuordnen und Stärken und Schwächen offenzulegen.

Auch bei dem zentralen Verfahrensschritt des Prozessmodells, der Ableitung des geeigneten Problemlösungsparadigmas, ist ein Angriffspunkt gegeben. Das Fehlen eines durchgängig formal bewiesenen Ansatzes verlangt nach einer Weiterentwicklung mit den Methoden der Informationstheorie. Weiterhin würde eine Untersuchung von Anwendungsfällen, bei denen

die Bedingungen für eine „*single level of abstraction"*-Charakteristik nicht gegeben sind, Einblick in die Bewältigung der Komplexitätssteigerung und anders gearteter Problemlösungsformen ermöglichen. Die Einbeziehung eines *JDL Level 3 Processing - Impact Assessment* - eröffnet das Feld der von Hypothesen und Planung geprägten Ansätze. Eine intensive Untersuchung und Umsetzung des *JDL Level 4 Processing - Process Refinement* - würde die Thematik der dynamischen Rekonfiguration und Leistungsanpassung zur Laufzeit adressieren. Dabei wäre die Beherrschung des graduellen Leistungszuwachses oder -abfalles von besonderem Interesse.

Um alternative Problemlösungsformen, Algorithmen oder Systemarchitekturen auf Basis quantitativer Kenngrößen vergleichbar und Zielgrößen für das *Process Refinement* verfügbar zu machen, existiert ein dringender Bedarf an Verfahren zur Leistungsmessung und Leistungsbewertung von Real-time Prozessführungen für mobile Applikationstechniken, die auf MSDF basieren. Erste Arbeiten in dieser Richtung wurden begonnen und in einem Vorschlag zur Bestimmung von „*measures of effectiveness"* und „*measures of performance"* in Grundzügen veröffentlicht [AOM05]. Die vorliegende Arbeit bestätigt die auf Erfahrung basierende Einschätzung der MSDF-Gemeinschaft, dass die Bewertung der Vorzüglichkeit einer Gesamtsystemlösung und des Einflusses der Fusionslösung daran zu den größten Herausforderungen der Fachdisziplin zählt. Diese Kosten/Nutzen-Abschätzung (*measures of effectiveness*) auf der Leitungsebene (*mission level*) bewegt sich für den Agrarbereich in einem durch Ökonomie, Ökologie und Soziologie aufgespannten Zielkorridor. Die Erforschung der entsprechenden Leistungsfähigkeit verlangt geradezu nach interdisziplinärer Zusammenarbeit und Bearbeitung im Forschungsverbund.

Die Agrarsystemtechnik und Landmaschinenindustrie ist seit langer Zeit in der Lage, Real-time Prozessführungen zu entwerfen und anzubieten, die auf analytischen Modellen und numerischer Informationsverarbeitung basieren. In dieser Arbeit wurde der Weg aufgezeigt, wie auch Prozessführungen auf der Basis von heuristischem Wissen mit einem wissensbasierten Systemansatz umgesetzt werden können. Gerade der wissensbasierte Ansatz bietet durch die Entflechtung des Experten-Wissens von der Inferenzmaschine für die Interdisziplinarität einen ungeheuren Vorteil. So können die naturwissenschaftlich, pflanzenbaulich, ökonomisch, ökologisch oder sozialwissenschaftlich orientierten Wissenschaftler sich auf ihr Wissen konzentrieren und im Grunde direkt in eine Prozessführung einfließen lassen, ohne tiefe Kenntnisse oder den Zugang zu einer prozedural implementierten Prozesssteuerung zu benötigen. Dieser Ansatz fördert das Denken in

Systemlösungen und sollte somit in vorbildlicher Weise einen interdisziplinären Forschungsansatz und gemeinsame Projektarbeit fördern.

In der Regel steht eine nachgewiesene Vorzüglichkeit einer Prozessführung in engem Zusammenhang mit der Qualität und Verfügbarkeit des prozeduralen Wissens und geeigneter Online-Sensoren. Die beste technische Prozessführungs-Infrastruktur ist nutzlos und findet keine Akzeptanz in der Praxis, falls die Qualität des Wissens und der Sensorik zu wünschen übrig lässt. Daher ist an diesen Stellen unbedingt weitere grundlagen- und anwendungsorientierte Forschungsarbeit zu erbringen.

Nichts ist wichtiger als ein guter Online-Sensor, der aber auch eine maschinenlesbare Beschreibung seiner Leistungsfähigkeit und deren Grenzen mit sich bringt. Die Weiterentwicklung im Bereich der pflanzenorientierten Online-Sensoren ist wünschenswert, die Erforschung und Markteinführung von bodenorientierten Online-Sensoren ist unerlässlich, um eine Verbesserung der Kontextsensitivität der Prozessführung für sensorgestützte Düngesysteme zu erreichen.

Für die Qualität des prozeduralen Wissens ist die Wissensakquisition und die Wissenspflege der Dreh- und Angelpunkt. Es ist anzunehmen, dass allgemein gültige Düngeregeln durchaus durch Fachexperten formuliert werden können, die standortbezogene individuelle Anpassung erfordert jedoch automatische Methoden, damit eine breite Marktdurchdringung nicht an einem *knowledge engineering bottleneck* scheitert. Die Ansätze zur Automatisierung wie bei [Wei06] sind ermutigend und sollten durch weitere Forschung vorangetrieben werden.

Auch ein altbekannter wissenschaftlicher Untersuchungsgegenstand der Teilflächenbewirtschaftung, die Genauigkeit und Vergleichbarkeit von *Precision Farming*-Karten, bleibt aktuell. Zumindest ist es unabdingbar, dass eine Information über die Genauigkeit dem Kartenmaterial als Metainformation beigelegt wird. Im Rahmen eines *JDL Level 4 Processing - Process Refinement* - könnte darauf reagiert werden und die Inferenz-Prozesse entsprechend adaptiert werden. In einem hochgradig dynamischen System wie einem Landwirtschaftlichen BUS-System mit Skalierbarkeit und Herstellerunabhängigkeit ist dies weit von einer trivialen Aufgabenstellung entfernt. Dynamische Anpassung und Optimierungsaufgaben stellen eine ernstzunehmende Herausforderung dar, die im Rahmen erster Untersuchungen angenommen wurde [BKB11].

Der Anknüpfungspunkt für die anwendungsorientierte Forschung wäre die Integration der vorgestellten Lösung in den realen Versuchsbetrieb, die somit eine mögliche experimentelle Erprobung erlaubt, aber auch ein Werkzeug zur Verbesserung von Online-Sensorik und der Qualität des Wissens bereitstellt.

Neben den allgemein gehaltenen Überlegungen hat die Arbeit in den Bereichen Systemarchitektur und Analyse der Landwirtschaftlichen BUS-Systeme, vorrangig des ISOBUS (ISO 11783), konkrete Vorschläge und Handlungsempfehlungen unterbreitet. Seitens der ISOBUS-Normung wurde mit der Definition der Geräteklasse der „*Sensor systems*“ und mit dem neuen Normvorschlag für die Integration des Sensor-Ansatzes der Kontext der Teilfächenbewirtschaftung auf alle drei möglichen Systemansätze zur Prozessführung in mobilen Applikationssystemen ausgeweitet. Die volle Leistungsfähigkeit und -breite des Sensor-Ansatz mit Kartenüberlagerung wird jedoch nur durch eine umfassende MSDF-Lösung erreicht, die nicht auf die Entscheidungsebene beschränkt bleibt. Dafür wurde die Systemarchitektur-Variante „*Distributed Sensor/Fusion*“ als geeignet identifiziert und nötige ISOBUS-Erweiterungen als Handlungsempfehlungen zusammengestellt:

- Erweiterung des Task Controller um die „In-field Controller“-Funktionalität,
- Definition eines Datenelementes „Überlagerungskarte“ („Overlay-Map (OMP)") zum Datenaustausch zwischen FMIS und MICS,
- Definition eines Datenaustausches zwischen FMIS und MICS für langfristiges explizites prozedurales Wissen nach der Spezifikation „W3C Rule Interchange Format (RIF)“,
- Definition von Datenbankelementen, d.h. entsprechende DDI, für Parameter von pflanzen-, boden- und wetterorientierten Online-Sensoren,
- Definition zweier komplementärer Klassen von MSDF-Knotenprozessoren für Online-Sensorik, um auch WSN integrieren zu können.

Inwieweit diese Verbesserungsvorschläge Eingang in die Normungsbemühungen finden werden, ist nicht absehbar. Für die zusätzlichen Datenbankelemente liegen die Hürden am niedrigsten und sie sind für Online-Sensoren unabhängig vom Systemansatz relevant. Daher bietet es sich an, sie als erstes in Angriff zu nehmen.

Sollte in Zukunft ein Landwirtschaftliches BUS-System neu konzipiert werden, so ist abgesehen von den vorausgehenden Forderungen darauf zu achten, dass die Leistungsfähigkeit und Systemarchitektur eine MSDF auf jeder der drei grundsätzlichen Fusionsebenen zulässt und somit auch die Rohwert-Ebene miteinschließt.

Sowohl bei einer Neuentwicklung als auch bei einer ISOBUS-Ergänzung nimmt die Schnittstelle bzw. die Schnittstellen zwischen dem drahtgebundenen ISOBUS-Kommunikationssystem und drahtlosen Kommunikationssystemen eine Schlüsselrolle ein. Die Tragweite dieser Schnittstelle hat nicht nur Auswirkungen auf die

Teilflächenbewirtschaftung, sondern erstreckt sich über alle Säulen des Präzisen Ackerbaus, d.h. Dokumentation, Teilflächenbewirtschaftung, Flottenmanagement und Feldrobotik (vgl. [Aue02]). Eine enge Zusammenarbeit der Normungsgremien zu ISO 11783 und zu NWI ISO 16867 ist daher unabdingbar und eine Brücke zwischen beiden Normen ist zu schlagen. Mit den drahtlosen Kommunikationsanbindungen wird eine tief ins MICS reichende Anbindung von WSN, Echtzeit-Fernerkundung und von (Web-)service-orientierten Lösungen realisierbar. Sowohl die Forschung als auch die Industrie haben das Gebiet bereits aufgegriffen. Aus dem Blickwinkel der MSDF rückt dabei die Systemarchitektur-Variante „*Fully integrated*“ in den Vordergrund, aber auch agentenbasierte Systemlösungen verdienen eine gesteigerte Aufmerksamkeit.

Die Rahmenbedingungen für eine erfolgreiche Umsetzung sind jedoch noch keineswegs dafür geschaffen. Drahtlose Kommunikation mit hoher Bandbreite, Echtzeitfähigkeit und lückenloser oder zumindest guter Abdeckung im ländlichen Raum ist noch nicht selbstverständlich. Auch gilt es, einen Standard bzw. Standards für alle im Prozessmodell identifizierten deklarativen und prozeduralen Wissensarten von unterschiedlicher Gültigkeitsdauer zu definieren. Ein möglicher Ausgangspunkt sind die Arbeiten und Standards zum Datenaustausch im Semantischen Web auf der Basis der *Web Ontology Language* (OWL) [W3C09], deren Eignung oder Anpassungsfähigkeit für den Agrarbereich zum Gegenstand einer Evaluierung gemacht werden sollte.

Ein Thema das in dieser Arbeit nur am Rande behandelt wurde, bei einer verteilten Systemlösung jedoch entscheidend die Leistungsfähigkeit einer MSDF-Lösung mitbestimmt, ist das dann verteilte *Database Management System*, mit den beiden Hauptkomponenten *Support Database* und *Fusion Database*. Diese verteilte Datenhaltung ist gekennzeichnet durch eine hochgradig heterogene Datenbasis mit großen Unterschieden in der zeitlichen und räumlichen Ausprägung und stellt besondere Anforderungen an Echtzeitfähigkeit, Konsistenz und Synchronisierungsmechanismen. Ohne eine Zusammenarbeit mit dem zugehörigen Spezialgebiet der Informatik bleibt der Weg zur erfolgreichen Bewältigung der Anforderungen versperrt bzw. hindernisreich.

Nachdem die Schlussfolgerungen und Ausblicke bisher den Forschungsaspekt betonten oder Normerweiterungen forderten, soll auch der Lehre, der Überführung in den praktischen Einsatz sowie den daraus resultierenden neuen Anforderungen an Landwirte und Betriebsleiter Raum gewährt werden.

Die realisierte Simulation könnte als Präsentationsmedium in der Lehre und der praktischen Ausbildung genutzt werden, um einen interaktiven Zugang zu Formen von „intelligenten

Prozessführungen“ zu schaffen und die Ausbildung zum Denken in Systemlösungen zu befördern. Dabei ist die Software-Simulation noch um die Integration des Kartierungsansatzes zu erweitern, so dass alle drei Systemansätze in skalierbarer Form präsentiert und verglichen werden können. Wünschenswert wäre ebenso eine Überführung in eine „*hardware in the loop*“-ISOBUS-Umgebung. Der Schwerpunkt läge dabei im Austausch der simulierten Prozessumgebung durch echte Hardware.

Der Technologietransfer in die praktische Anwendung ist schwer abschätzbar. Die theoretischen Grundlagenarbeiten sind erbracht. Der ISOBUS bietet mit der bereits beantragten Erweiterung um den Sensor-Ansatz mittelfristig einen Weg zur grundsätzlichen Umsetzung aller drei Systemansätze, wenngleich für die volle Leistungsspanne die Integration der „*In-field Controller*“-Struktur nötig wäre. Die Anwendung ist nicht nur auf die Düngung beschränkt, sondern ist prinzipiell auch auf andere Applikationstätigkeiten übertragbar. Die neue Generation von Terminals der gehobenen Ausstattungslinien mit *Virtual Terminal* und *Task Controller*-Implementierungen verfügen über genügend Leistungsfähigkeit für die Realisierung von Prozessführungen mit wissensbasierten Optimierungsstrategien zur Steuerung und Regelung. Die Hindernisse liegen somit weniger im technischen Bereich als vielmehr im Feld der Entwicklungsprozesse. Das aufzeigte Spannungsfeld zwischen Funktionaler Sicherheit und dem Entwicklungsprozess für wissensbasierte Systeme wird vermutlich einem schnellen und umfangreichen Einzug von wissensbasierten Prozessführungslösungen in die landwirtschaftliche Praxis hemmend im Wege stehen. Mehr als Mutmaßungen lassen sich nicht anstellen, wie eine verstärkte „Sogwirkung“ aus der Praxis erzeugt werden könnte. Wissenschaftliche Nachweisführung über die Vorzüglichkeit einer entsprechenden Lösung in Pilotprojekten oder erfolgsversprechende *On-farm Research*-Ergebnisse auf Betriebsebene könnten für eine Beschleunigung sorgen. Im Industriebereich verspricht noch am ehesten die Vorgehensweise des „*Agile Software Development*“ in Verbindung mit einer ambitionierten und experimentierfreudigen Kundengruppe eine beschleunigte Praxiserprobung und Produkteinführung.

Aber auch der Landwirt oder Betriebsleiter würde sich bei der Einführung einer MSDF-basierten Real-time Prozessführung in sensorgestützten Düngesystemen in einem landwirtschaftlichen BUS-System mit einer Reihe an neuen Fragestellungen und Anforderungen konfrontiert sehen. Dabei würden sich die Fragen hinsichtlich der Auslagerung von Tätigkeiten und Arbeitsverfahren an externe Dienstleister wie Lohnunternehmer oder Maschinenringe immer mehr auch auf den Bereich der betrieblichen IT-Infrastruktur und neu auf die Wissensbeschaffung, -haltung und -verwaltung ausweiten. Zu

vertrauten Betriebsfaktoren wie z.B. Pacht, Maschinen, Arbeitskraft oder Dünger kommt die neue „Produktkategorie“ Wissen und dessen Management hinzu. Auch im persönlichen Bereich fände er sich in einer geänderten Position wieder. Bisher typische Aufgaben und Verantwortungsbereiche eines Entscheiders würden vermehrt durch sogenannte „intelligente“ Prozessführungen von einer Maschine oder einem Webservice übernommen. Sein Anforderungsprofil würde sich in diesem Bereich in Richtung Beschaffung und Koordination von Wissensquellen, Planung und Durchführung der Wissenspflege oder Anpassung an die betriebsbezogenen Bedürfnisse verschieben.

Diese Veränderungen gingen einher mit sozialwissenschaftlichen Problemstellungen im Hinblick auf die Auswirkungen und die Akzeptanz dieser Art von Technik. Die Agrarsystemtechnik sollte dies als Ansporn nehmen, um sich verstärkt der Mensch-Maschine-Schnittstelle zuzuwenden und als Technikwissenschaft einen Beitrag liefern, um Ängste und Barrieren abzubauen. Wie aktuelle Entwicklungen im *Consumer*-Bereich mit *multi-touch*-fähigen *Tablet*-Computern vormachen, hilft die Konzentration auf die grundlegenden Nutzerwünsche, intuitive Bedienung und das „Verstecken“ der Systemkomplexität. Im Zusammenhang mit MSDF würde ein Übergang auf verteilte service-orientierte Systemlösungen den Landwirt oder Betriebsleiter als letzte Entscheidungsinstanz, Koordinationsstelle und Kontrollstelle benötigen. Damit wird ein „*human in loop*“ systemrelevant und erfordert richtungsweisende Forschung im Bereich eines *JDL Level 5 Processing - Cognitive refinement* - gemeinsam mit den relevanten geisteswissenschaftlichen Disziplinen.

Bisher war der Blick auf den Landwirt oder Betriebsleiter als reinen Anwender gerichtet, der Dienstleistungen nur in Anspruch nimmt. Langfristig gesehen, bieten sich für ihn aber auch neue Geschäfts- und Betätigungsfelder, wenn er als „Wissensprovider“ auftritt und Wissensbausteine anbietet oder managen hilft. Derzeit ist jedoch in Betracht zu ziehen, dass weder die service-orientierte Infrastruktur weitverbreitet vorliegt noch die im Feld vorrangig installierte Prozessrechentechnik einen wissensbasierten Prozessführungsansatz umsetzen kann.

8 Zusammenfassung

Vor dem Hintergrund einer wachsenden Weltbevölkerung und steigenden Nachfrage nach nachwachsenden Rohstoffen ist die praktische Umsetzung und Weiterentwicklung von Präzisionslandwirtschaft ein Lösungsansatz für eine umweltschonende Produktivitätsteigerung. So wurde in dem interdisziplinären Forschungsvorhaben der DFG-Forschergruppe IKB-Dürnast an der TU München an dem neuen Systemansatz Sensor-Ansatz mit Kartenüberlagerung für die teilflächenspezifische N-Düngung geforscht. Im Teilprojekt 8 wurde hierzu eine Real-time Prozessführung für sensorgestützte Düngesysteme entwickelt und getestet.

Bei dem Ansatz gilt es, einen Prozess oder ein System, hier Pflanzen und ihre nähere Umgebung, zu einem ökologischen und ökonomischen Optimum zu führen. Dies erfordert Prozess-Eingangsgrößen verschiedensten Ursprungs, z.B. Online-Sensorik und Kartenmaterial, sowie Information über den aktuellen Zustand des Prozesses zusammenzuführen und mit (Experten)Wissen daraus einen Applikationssollwert für jede einzelne Teilfläche in Echtzeit abzuleiten, d.h. unter Anwendung von Multisensor Data Fusion (Abk.: MSDF)-Technik. Für eine erfolgsversprechende praktische Umsetzung und Marktdurchdringung ist bei der Realisierung dieser Prozessführung vorrangig auf die den Stand der Technik repräsentierenden standardisierten landwirtschaftlichen BUS-Systeme zu setzen.Verglichen mit den herkömmlichen Prozessführungsansätzen führt die Einbeziehung von Daten mehrerer Systemeinheiten zwangsläufig zu einer steigenden Komplexität, bei deren Bewältigung strukturierte Analyse-, Entwurfs- und Entwicklungsprozesse helfen, um dennoch eine effiziente und zielorientierte Implementierung zu erreichen. Auch wenn der Agrarsystemtechnik Sensorfusionsansätze vor allem zur Navigation und zur Zustandsüberwachung nicht fremd sind, wurde bisher keine allgemeine Analyse- und Entwurfsmethode für MSDF dokumentiert bzw. vorgeschlagen, geschweige denn auf Landwirtschaftliche BUS-Systeme abgestimmt.

Daher wird folgendes MSDF-Framework für eine auf MSDF basierende Real-time Prozessführung in einem mobilen landwirtschaftlichen BUS-System vorgeschlagen. So sollte ein durchgängiges MSDF-Framework grundsätzlich unterschiedliche Abstraktionsebenen besitzen und eine Top-Down-Dekomposition der Anforderungen, sowie einen anschließenden strukturierten Systementwurf erlauben. Auf der höchsten Abstraktionsebene beschreibt und legt ein funktionales Modell, das *„Revised JDL data fusion model“* fest, welche

Analysefunktionalität oder -prozesse durchgeführt werden müssen. Hingegen beschreibt ein Prozess-Modell auf hoher Abstraktionsebene, wie diese Analysen oder Prozeduren geleistet werden können. Es wird auf das Prozessmodell von Antony zurückgegriffen. Auf der Basis dieser abstrakten Sichtweise von Anforderungen, Spezifikationen und Problemlösungs-Paradigmen ist dann eine ISOBUS (ISO 11783) konforme Systemarchitektur zu entwerfen und bei der weiteren Transformation in eine konkrete technische Realisierung in Hard- und Software umzusetzen.

Anhand des spezifischen Anwendungsbeispiels einer Real-time Prozessführung in sensorgestützten Düngesystemen nach dem Sensor-Ansatz mit Kartenüberlagerung für intensive N-Düngung ließ sich das MSDF-Framework auf Anwendbarkeit überprüfen, das Verständnis fördern und die Auswirkungen auf landwirtschaftliche BUS-Systeme analysieren.

Das Beispiel konnte aus dem funktionalem Blickwinkel nach dem *„Revised JDL data fusion model“* durchgehend spezifiziert werden. Der Schwerpunkt liegt jedoch auf dem *„JDL Level 2 Processing – Situation Assessment“*, d.h. einer umfassenden Situationsbewertung der aktuellen Online-Sensorik-Messwerte mit einer kontextsensitiven Interpretation. Dabei ist die Komposition von kurz-, mittel- und langfristigem deklarativem und prozeduralem Wissen gefordert. Mittels des Prozessmodells konnte hierfür ein Produktionssystem mit einem vorwärts-verkettetem Inferenzmechanismus als ein geeigneter Problemlösungs-Ansatz identifiziert werden. Auf Basis der Ergebnisse der funktionalen und prozeduralen Modellebenen wurde eine Systemarchitektur vom Typ *„Distributed Sensor/Fusion“* vorgeschlagen, d.h ein verteiltes Sensornetzwerk (kurzfristiges Wissen) (z.B. Online-Sensor, GPS-Empfänger, *wireless sensor networks*) und ein zentraler Data Fusion-Knoten, ein *„In-field Controller“*, mit mittelfristigem Wissen sowie langfristig deklarativem Wissen in Form der Überlagerungskarten und prozeduralem Wissen, dem MSDF-Algorithmus.

Auch wenn mit der ISOBUS-Definition der Geräteklasse „*Sensor systems*“ und mit dem neuen Normvorschlag für die Integration des Sensor-Ansatzes der Kontext der Teilfächenbewirtschaftung auf alle drei möglichen Systemansätze zur Prozessführung in mobilen Applikationssystemen ausgeweitet wurde, läßt sich die volle Leistungsfähigkeit und -breite des Sensor-Ansatzes mit Kartenüberlagerung nur durch eine umfassende MSDF-Lösung erreichen, die nicht nur die Entscheidungsebene sondern auch die Feature/Merkmals-Ebene abdeckt. Dafür werden aber Normerweiterungen wie der *„In-field Controller“*, ein zusätzliches Datenelement „Überlagerungskarte“ („*Overlay-Map* (OMP)"), die

Datenaustauschmöglichkeit zwischen FMIS und MICS für langfristiges explizites prozedurales Wissen, neue Datenbankelemente für Parameter von pflanzen-, boden- und wetterorientierten Online-Sensoren sowie die Definition zweier komplementärer Klassen von MSDF-Knotenprozessoren für Online-Sensorik zur Integration von *wireless sensor networks* angeregt.

Die Realisierbarkeit der MSDF-Lösung des spezifischen Anwendungsbeispiels wurde mit der Implementierung als intutiv bedienbare Software-Simulation in Form eines Expertensystems mit Produktionssystem gezeigt. Dabei kam die hybride Expertensystem-Shell JESS für das Regelwerk und Java zur Realisierung des graphischen Benutzerinterfaces und zur Nachbildung der Prozessumgebung zum Einsatz. Beim pflanzenbaulichen und ökonomischen Teil der Wissensakquisition wurde eng mit einem weiteren IKB-Teilprojekt zusammengearbeitet. In der Testphase ermittelte Durchlaufzeiten für einen MSDF-Zyklus auf einer leistungsschwachen Notebook-Hardware belegen die grundsätzliche Eignung für eine Real-time Prozesssteuerung mit gegenwärtiger mobiler Agrar-Rechnertechnik.

Der angewandte evolutionäre Entwicklungsprozess zeigt den Weg auf, wie in der Agrarsystemtechnik Real-time Prozessführungen nicht nur mit analytischen Modellen und numerischer Informationsverarbeitung entworfen und realisiert, sondern auch auf der Basis von heuristischem Wissen mit einem wissensbasierten Systemansatz umgesetzt werden können. Dabei tritt jedoch ein Bruch mit den traditionellen Entwicklungsprozessen (z.B. Wasserfall- oder V-Modell) zutage. Jedoch bietet bei wissensbasierten Systemen die Trennung von Inferenzmaschine und Wissensbasis neue Möglichkeiten für Dienstleistungen und individuell auf den einzelnen Betrieb zugeschnittene Systemlösungen.

Die vorliegende Arbeit bestätigt aber auch die auf Erfahrung basierende Einschätzung der MSDF-Gemeinschaft, dass die Bewertung der Vorzüglichkeit einer Gesamtsystemlösung und des Einflusses der MSDF-Lösung daran zu den größten Herausforderungen der Fachdisziplin zählt. Diese Kosten/Nutzen-Abschätzung (*measures of effectiveness*) auf der Leitungsebene (*mission level*) bewegt sich für den Agrarbereich in einem durch Ökonomie, Ökologie und Soziologie aufgespannten Zielkorridor. Die Erforschung der entsprechenden Leistungsfähigkeitsmessung und –bewertung auf den unterschiedlichen Fusionsebenen sowie eine Ausweitung der untersuchten Eingrößen- hin zu einer Mehrgrößen-Prozesssteuerung wäre von besonderem grundlagenorientierten Interesse.

9 Summary

One solution to address the challenges of a rising world population and emerging demand for renewable resources could be deployment and further development of Precision Agriculture in order to achieve an environmentally friendly increase of productivity. Within the integrated research project of the DFG research group IKB Duernast the new system approach *"Real-time approach with map overlay"* for site specific N-fertilizer application was investigated in detail. Therefore a real-time process control for a sensor based fertilizer application system was implemented and tested in subproject 8.

The basic idea of this approach is to lead a process or system, here plants and their surroundings, to an ecological and economic optimum. This requires to fuse information about the current state of the process and its inputs from different sources, i.e. "precision farming maps" or on-line sensor technology process data, and to derive the site specific application set point by (expert) knowledge in real-time. In brief, this requires Multisensor Data Fusion (MSDF) techniques. Successful deployment and market penetration would benefit from an implementation of the process control based on standardized Agricultural BUS-Systems which represent the state of the art. Fusion of data from several different system components results inevitably in a growing complexity compared to conventional process control approaches. Structured analysis, design and development processes help to cope with this and guarantee an efficient and goal-oriented implementation. Although sensor fusion approaches are not completely new to Agricultural Engineering, mainly applied for navigation and condition monitoring, no general analysis and design approach for MSDF has been documented or proposed respectively, not to mention that such an approach has been adapted to Agricultural BUS-Systems.

Thus, following MSDF-Framework for a MSDF based real-time process control in a mobile Agricultural BUS-System is proposed. Basically, an integrated MSDF-Framework should possess different levels of abstraction and should allow a top down decomposition of requirements as well as a following structured system design. A functional model, the *"Revised JDL data fusion model"*, should describe at the highest abstraction level what analysis functions or processes need to be performed. While a process model describes at a high level of abstraction how this analysis is accomplished. The process model defined by Antony is selected. Based on these abstract view of demands, requirements and problem-solving paradigm, an ISOBUS (ISO 11783) compliant system architecture has to be designed

and to be transformed during further steps into a specific hardware and software implementation.

Proof of concept of the MSDF-Framework, facilitation of understanding and the analysis of effects on Agricultural BUS-Systems could be conducted on the basis of the specific use case of a real-time process control for a sensor based fertilizer application system for intensive N-fertilization according to the *"Real-time approach with map overlay"* system approach.

From a functional point of view the use case could be completely specified according to the *„Revised JDL data fusion model"*. However the main focus is put on the *"JDL Level 2 Processing - Situation Assessment"*, i. e. an assessment of current on-line sensor technology measurements with context-sensitive interpretation. This task requires the composition among short-, medium-, and long-term declarative knowledge and procedural knowledge. By the means of the process model a forward-inference production rule paradigm was identified as an appropriate problem-solving approach. Based on the results of the functional and procedural models a system architecture according to the pattern *"Distributed Sensor/Fusion"* was proposed, i. e. a distributed sensor network (short term knowledge) (e. g. online-sensor, GPS-receiver, wireless sensor networks) and a central fusion node, an *"In-field Controller"*, with medium-term and long-term declarative knowledge as overlay maps and with procedural knowledge, the MSDF-algorithm.

Even though the ISOBUS definition of the device class *"Sensor systems"* and the new proposal for integration of the sensor approach into the standard broadens the scope of site specific crop management to all three system approaches for process control in mobile application systems, only a comprehensive MSDF-solution, that covers not just the decision level but also the feature level, enables the full performance and spectrum of the "Real-time approach with map overlay". Therefore, extensions of the standard as the *"In-field Controller"*, an additional data element *"Overlay-Map (OMP)"*, the data exchange possibility between FMIS and MICS for long-term explicit procedural knowledge and new data dictionary elements for plant, soil and weather attributes are suggested. Furthermore, the definition of two complementary classes of MSDF node processors for online sensors would allow the integration of wireless sensor networks.

Feasibility of the MSDF-solution for the specific use case was demonstrated by the implementation as intuitively useable software simulation comprising an expert system with integrated production system. For this the hybrid expert system shell JESS was used for

implementing the rule sets and Java for the graphical user interface implementation as well as to simulate the whole process environment. Knowledge acquisition for the crop production and economics parts were conducted in close cooperation with another IKB subproject. During testing phase measured processing times for a MSDF cycle running on a notebook of poor hardware performance prove basic real-time capability for a process control based on current mobile agricultural computing hardware.

The adopted evolutionary development process shows for Agricultural Engineering the way how to design und implement real-time process controls not only by the means of analytical models and numerical information processing but also by heuristic knowledge and by a knowledge-based system approach. However, this reveals a break with more conventional development processes (e. g. Waterfall- or V-model). On the other hand, the separation of the inference engine and knowledge base of a knowledge-based system offers new opportunities for services and system solutions customized for each farm.

The opinion of the MSDF-community can be confirmed that the exact proof and quantification of superior performance and effectiveness of a system solution due to applied MSDF-technique is one of the biggest challenges of the discipline. The cost-benefit analysis (measures of effectiveness) for agriculture on a mission level is characterized by and shall be aligned with the areas economics, ecology and sociology. Methods how to measure and to assess the effectiveness and performance on different fusion levels as well as the extension of the investigated monovariable to a multivariable process control would be of a special basic research interest.

Literaturverzeichnis

1	[AHMU04]	ADAMCHUK, V. I., HUMMEL, J. W., MORGAN, M. T., UPADHYAYA, S. K., 2004: On-the-go soil sensors for precision agriculture. Computers and Electronics in Agriculture 44, pp. 71-91.
2	[ARSS11]	ADAMCHUK, V. I., ROSSEL R. A. V., SUDDUTH K. A., SCHULZE LAMMERS, P., 2011: Sensor Fusion for Precision Agriculture. In: Sensor Fusion - Foundation and Applications, Rijeka, Croatia: InTech.
3	[Ahl08]	AHLERS, E., 2008: Industrie-Netz. Ethernet abseits des Büros. c't Heft 9, S. 202-206.
4	[AS04]	ALBERTOS, P., SALA, A., 2004: Multivariable Control Systems. London: Springer.
5	[ADH06]	ANDREE, H., DOLUD, M., HARTUNG, E., 2006: Near-Infrared Spectroscopy as an Analytical Process Technology in Manure Application for Fertilizing. In: VDI-Berichte Nr. 1958, World Congress "Agricultural Engineering for a Better World" in Bonn, pp. 345-346.
6	[Ant95]	ANTONY, R. T., 1995: Principles of Data Fusion Automation. Boston: Artech House.
7	[Ant01]	ANTONY, R. T., 2001: Data Fusion Automation: A Top-Down Perspective. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.
8	[APS93]	ARTMANN, R., PAUL, W., SPECKMANN, H., 1993: Die Bausteine von Prozeßleitsystemen. In: Elektronik und Computer in der Landwirtschaft, Stuttgart: Eugen Ulmer.
9	[Aue89]	AUERNHAMMER, H., 1989: The German Standard for Electronical Tractor Implement Data Communication. In: AGROTIQUE 89. Proceedings of the second international conference in Bordeaux, France, pp. 395-402.
10	[Aue91]	AUERNHAMMER, H., 1991: Elektronik in Traktoren und Maschinen: Einsatzgebiete, Funktion, Entwicklungstendenzen. München: BLV Verlagsgesellschaft.
11	[Aue93]	AUERNHAMMER, H., 1993: Geschichte des LBS. In: Landwirtschaftliches BUS-System - LBS. KTBL-Arbeitspapier 196.
12	[Aue01a]	AUERNHAMMER, H., 2001: Precision farming - the enviromental challenge. Computers and Electronics in Agriculture 30, pp. 31-43.
13	[Aue01b]	AUERNHAMMER, H., 2001: TP8: Entwicklung und Test einer Realtime-Prozessführung für sensorgestützte Düngesysteme. In: Fortsetzungsantrag für die Forschergruppe Informationssystem Kleinräumige Bestandesführung - Dürnast - (AU 149/1-4) an die DFG, Weihenstephan, S. 1-23. (unveröffentlicht)

14	[Aue02]	AUERNHAMMER, H., 2002: Automatische Betriebsdatenerfassung im Ackerbau und seine Nutzanwendung. In: Landtechnik-Schrift 14: Ackerbau der Zukunft, Tagungsband zur Landtechnischen Jahrestagung 2002 in Deggendorf, S. 45-58.
15	[Aue06]	AUERNHAMMER, H., 2006: Präziser Ackerbau. In: Jahrbuch Agrartechnik, Band 18, S. 35-42.
16	[Aue11]	AUERNHAMMER, H., 2011: Twenty Years of Precision Farming - more Questions than Answers?. In: ACPA2011, The 4th Asian Conference on Precision Agriculture in Obihiro, Japan (CD-ROM), pp. 1-6.
17	[AS99]	AUERNHAMMER, H., SCHUELLER, J. K., 1999: Precision Farming. In: CIGR Handbook of Agricultural Engineering, Vol. III: Plant Production Engineering, pp. 598-616.
18	[AS06]	AUERNHAMMER, H., SPECKMANN, H., 2006: Dedicated Communication Systems and Standards for Agricultural Applications. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 435-452.
19	[ADMSSW99]	AUERNHAMMER, H., DEMMEL, M., MAIDL, F. X., SCHMIDHALTER, U., SCHNEIDER, T., WAGNER, P., 1999: An On-Farm Communication System for Precision Farming with Nitrogen Real-Time Application. In: ASAE/CSAE-SCGR Annual International Meeting, St. Joseph, Paper No. 99 1150.
20	[ADMRW93]	AUERNHAMMER, H., DEMMEL, M., MUHR, T., ROTTMEIER, J., WILD, K., 1993: Yield Measurement on combine harvesters. In: ASAE Meeting, St. Joseph, Paper No. 93-1506.
21	[AOM05]	AUERNHAMMER, H., OSTERMEIER, R., MACHADO, P. P., 2005: Assessing multisensor data fusion performance of a real-time process control for a sensor based fertilizer application system. In: Book of Abstracts 5 ECPA - 2 ECPLF, pp. 35-36.
22	[AOMSSW06]	AUERNHAMMER, H., OSTERMEIER, R., MAIDL, F-X., SCHMIDHALTER, U., SCHNEIDER, T., WAGNER, P., 2006: DFG-Forschergruppe FOR 473 Informationssystem Kleinräumige Bestandesführung Dürnast IKB-Dürnast. Technische Universität München. Endbericht.
23	[Auf98]	AUFHAMMER, W., 1998: Getreide- und andere Körnerfruchtarten. Stuttgart: Verlag Eugen Ulmert.
24	[Aug01]	AUGSBURGER, C., 2001: Concept of a Cost-Accounting System for the Interpretation of Spatially Variable Data. In: Proceedings of the 3rd European Conference on Precision Agriculture, Montpellier, pp. 629-634.
25	[BA05]	BACHMAIER, M., AUERNHAMMER, H., 2005: Yield mapping based on robust fitting paraboloid cones in butterfly and elliptic neighborhoods. In: Precision Agriculture 05. Proceedings of the ECPA Conference 2005 in Uppsala, Sweden, pp. 741-750.

26	[Bal00]	BALZERT, H., 2000: Lehrbuch der Software-Technik: Bd. 1. Software-Entwicklung. Heidelberg, Berlin: Spektrum Akademischer Verlag.
27	[Bal98]	BALZERT, H., 1998: Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Heidelberg, Berlin: Spektrum Akademischer Verlag.
28	[BF81]	BARR, A., FEIGENBAUM, E., 1981: The Handbook of Artificial Intelligence. Volume 1. Los Altos: William Kaufmann Inc.
29	[Bar95]	BARTELME, N., 1995: Geoinformatik, Modelle, Strukturen, Funktionen. Berlin, Heidelberg: Springer-Verlag.
30	[BBMK11]	BARTOLEIN, C., BLANK, S., MEYER, A., KORMANN, G., 2011: Seamless Data Management for Agricultural Vehicles within the iGreen Infrastructure. In: VDI-Berichte Nr. 2124, Tagung Landtechnik 2011 in Hannover, pp. 293-298.
31	[BDCD99]	BAUER, H., DIETSCHE, K-H., CREPIN, J., DINKLER, F., 1999: Kraftfahrtechnisches Taschenbuch/Bosch. -23., aktualisierte und erw. Aufl.. Braunschweig, Wiesbaden: Vieweg.
32	[BO99]	BEDWORTH, M., O'BRIEN, J., 1999: The Omnibus model: a new model of data fusion?. In: Proceedings of the 2nd International Conference on Information Fusion in Sunnyvale (CD-ROM), pp. 1-9.
33	[Ber93]	BERG, E., 1993. Rechnergestützte Betriebsführung. In: Elektronik und Computer in der Landwirtschaft, Stuttgart: Eugen Ulmer Verlag.
34	[Bil99]	BILL, R., 1999: Grundlagen der Geo-Informationssysteme. Band 1 Hardware, Software und Daten. Heidelberg: Wichmann.
35	[Bil03]	BILLER, R. H., 2003: Das Projekt Advanced Optoelectronic System (AOS). Landtechnik 6, S. 380-381.
36	[Bla99]	BLACKMORE, S., 1999: Remedial Correction of Yield Map Data. Precision Agriculture 1, pp. 53-66.
37	[BG06]	BLACKMORE, S., GRIEPENTROG, H. W., 2006: Autonomous Vehicles and Robotics. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 204-215.
38	[BGF03]	BLACKMORE, S., GODWIN, R. J., FOUNTAS, S., 2003: The Analysis of Spatial and Temporal Trends in Yield Map Data over Six Years. Biosystems Engineering 84 (4), pp. 455-466.
39	[BKB11]	BLANK, S., KORMANN, G., BERNS, K., 2011: A Modular Sensor Fusion Approach for Agricultural Machines. In: XXXIV CIOSTA CIGR V Conference 2011 (CD-ROM), Vienna-Austria, pp. 1-10.
40	[BP02]	BLASCH, E. P., PLANO, S., 2002: JDL Level 5 Fusion Model "User Refinement" Issues and Applications in Group Tracking. Proceedings of SPIE Vol. 4729, pp. 270-279.

41	[BS01]	BOWMAN, C. L., STEINBERG, A. N., 2001: A Systems Engineering Approach for Implementing Data Fusion Systems. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.
42	[BFOS84]	BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., STONE, C. J., 1984: Classification and regression trees, Belmont, CA, USA: Wadsworth International Group.
43	[Bro03]	BRONSON, G. J., 2003: JAVA™ for Engineers and Scientists. Belmont, CA, USA: Brooks/Cole a division of Thomson Learning.
44	[Bun10]	BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG REFERAT BIOÖKONOMIE, 2010: Nationale Forschungsstrategie BioÖkonomie 2030. Bonn, Berlin.
45	[BB02]	BUZAN, T., BUZAN, B., 2002: Das Mind-Map-Buch. Die beste Methode zur Steigerung ihres geistigen Potentials. München: mvg Verlag.
46	[Cal11]	CALAPRICE, A., 2011: Einstein, Albert, 1879-1955: The ultimate quoteable Einstein. Princeton, NJ, USA: Princeton University Press.
47	[CL03]	CORVIN, D. L., LESCH, S. M., 2003: Application of Soil Electrical Conductivity to Precision Agriculture: Theory, Principles, and Guidelines. Agronomy Journal 95, pp. 455-471.
48	[DT03]	DABAS, M., TABBAH, A., 2003: A Comparison of EMI and DC methods used in soil mapping - theoretical considerations for Precision Agriculture. In: Precision Agriculture. Proceedings of the 4th European Conference on Precision Agriculture in Berlin, Germany, pp. 121-127.
49	[Das94]	DASARATHY, D., 1994: Decision Fusion. Los Alamitos, CA, USA: IEEE Computer Society Press.
50	[DN06]	DEMMEL, M., NESER, S., 2006: Verfahrenstechnik der Düngung. In: Die Landwirtschaft, Pflanzliche Erzeugung, München: BLV Buchverlag.
51	[Deß01]	DEßLOCH, S., 2001: SQL-Norm und Java. Datenbank-Spektrum 1, S. 25-32.
52	[DIN9684]	DIN, -: DIN 9684 Teil 2-5. Landmaschinen und Traktoren - Schnittstellen zur Signalübertragung. Berlin.
53	[DINIEC60050-351]	DIN IEC, 2009: DIN IEC 60050-351:2009-06. Internationales Elektrotechnisches Wörterbuch - Teil 351: Leittechnik (IEC 60050-351:2006). Berlin.
54	[DD06]	DOLESCHEL, P., DEMMEL, M., 2006: Grundlagen des Pflanzenbaus. In: Die Landwirtschaft, Pflanzliche Erzeugung, München: BLV Buchverlag.

55	[Dur99]	DURLESSER, H., 1999: Bestimmung der Variation bodenphysikalischer Parameter in Raum und Zeit mit elektromagnetischen Induktionsverfahren. Dissertation: Technische Universität München.
56	[EJO07]	EHLERT, D., JUERSCHIK, P., OTTER-NACKE, S., 2007: Advanced pendulum sensor for site specific crop production. In: VDI-Berichte Nr. 2001, Tagung Landtechnik 2007 in Hannover, pp. 519-524.
57	[EA06]	EHRL, M., AUERNHAMMER, H., 2006: X-By-Wire via ISOBUS Communication Network. In: Automation Technology for Off-Road Equipment, Proceedings of the 1-2 September 2006 Conference, Bonn, pp. 227-236.
58	[Eck12]	ECKL, T., 2012: Kommandozentrale für mobile Arbeitsmaschinen. Hanser automotive, electronics, systems 1-2, S. 48-50.
59	[EOBKR10]	ENGEL, T., OSTERMEIER, R., BARTOLEIN, C., KORMANN, G., RUSCH, C., 2010: Flexible Maschine-zu-Maschine-Kommunikation. In: Automatisierung und Roboter in der Landwirtschaft, KTBL-Schrift 480, S. 127-137.
60	[Eng91]	ENGEL, T., 1991: Entwicklung und Validierung eines Simulationsmodells zur Stickstoffdynamik in Boden und Pflanze mit Hilfe objektorientierter Programmierung. Dissertation: Technische Universität München.
61	[EIH06]	EWERS, M., ISENSEE, E., HARTUNG, E., 2006. NIR - Sensor for determination of product quality of grain while combining. In: VDI-Berichte Nr. 1958, Congress "Agricultural Engineering for a Better World" in Bonn, pp. 163-164.
62	[Fär92]	FÄRBER, G., 1992: Prozessrechentechnik: Grundlagen, Hardware, Echtzeitverhalten, technische Ausprägung. Berlin: Springer-Verlag.
63	[FAS06]	FERENTINOS, K. P., ARVANITIS, K. G., SIGRIMIS, N. A., 2006: Internet Use in Agriculture, Remote Service, and Maintenance: E-Commerce, E-Business, E-Consulting, E-Support. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 453-464.
64	[Föl06]	FÖLSTER, N., 2006: Expertensystemtechnologie für Teleservice bei Landmaschinen. Dissertation: Technische Universität Carolo-Wilhelmina zu Braunschweig.
65	[For82]	FORGY, C. L., 1982: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. Artificial Intelligence 19, pp. 17-37.
66	[Fra04]	FRADEN, J., 2004: Handbook of modern sensors: physics, designs, and applications. New York: Springer Science+Business Media.

67	[Fre03]	FREIMANN, R., 2003: Automation mobiler Arbeitsmaschinen - Gerät steuert Traktor. Dissertation: Technische Universität München.
68	[Fri03]	FRIEDMAN-HILL, E., 2003: JESS in Action. Greenwich: Manning Publications.
69	[FSHHS05]	FULTON, J. P., SHEARER, S. A., HIGGINS, S. F., HANCOCK, D. W., STOMBAUGH, T. S., 2005: Distribution Pattern Variability of Granular VRT Applicators. Transactions of the ASAE 48 (6), pp. 2053-2064.
70	[FSHDS05]	FULTON, J. P., SHEARER, S. A., HIGGINS, S. F., DARR, M. J., STOMBAUGH, T. S., 2005: Rate Response Assessment from Various Granular VRT Applicators. Transactions of the ASAE 48(6), pp. 2095-2103.
71	[GBM03]	GALLARDO, D., BURNETTE, E., MCGOVERN, R., 2003: Eclipse in Action. A Guide for Java Developers. Greenwich: Manning Publications.
72	[Gan05]	GANDORFER, M., 2005: Bewertung von Precision Farming dargestellt am Beispiel der teilflächenspezifischen Stickstoffdüngung. Dissertation: Technische Universität München.
73	[GP01]	GRIEPENTROG, H. W., K. PERSSON, K., 2001: A model to determine the positional lag for fertiliser spreader. In: Proceedings of the 3rd European Conference on Precision Agriculture, Montpellier, pp. 671-676.
74	[GBV06]	GRIEPENTROG, H. W., BLACKMORE, B. S., VOUGIOUKAS, S. G., 2006: Positioning and Navigation. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 195-204.
75	[GBM88]	GUYOT, G., BARET, F., MAJOR, D. J., 1988: High spectral resolution: Determination of spectral shifts between the red and near infrared. International Archives of Photogrammetry and Remote Sensing 11, pp. 750-760.
76	[HL01]	HALL, D. L., LLINAS, J., 2001: Multisensor Data Fusion. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.
77	[HM04]	HALL, D. L., MCMULLEN, S. A. H., 2004: Mathematical Techniques in Multisensor Data Fusion. Boston: Artech House.
78	[HS01]	HALL, D. L., STEINBERG, A. N., 2001: Dirty Secrets in Multisensor Data Fusion. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.

79	[HHT01]	HALL, M. J., HALL, S. A., TATE, T., 2001: Removing the HCI Bottleneck: How the Human-Computer Interface (HCI) Affects the Perfomance of Data Fusion Systems. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.
80	[HZN02]	HAN, S., ZHANG, Q., NOH, H., 2002: Kalman filtering of DGPS positions for a parallel tracking application. Transactions of the ASAE 45(3), pp. 553-560.
81	[Hee07]	HEEGE, H., 2007: Zur optischen Erfassung der pflanzlichen Stickstoff-Versorgung. Landtechnik 2, S. 78-79.
82	[HH09]	HIERONYMUS, P., HENNINGER, G., 2009: ISOBUS - Aktueller Stand und Herausforderungen in Entwicklung, Implementierung und praktischer Anwendung. In: Jahrbuch Agrartechnik, Band 21, S. 35-40.
83	[HH07]	HIERONYMUS, P., HOFMANN, R., 2007: Kommunikationssysteme. In: Jahrbuch Agrartechnik, Band 19, S. 35-42.
84	[HHG08]	HIERONYMUS, P., HENNINGER, G., GOTTSCHALK, K., 2008: Kommunikationssysteme. In: Jahrbuch Agrartechnik, Band 20, S. 37-42.
85	[HSS99]	HOFSTEE, J. W., SPEELMAN, L., SCHEUFLER, B., 1999: Fertilizer Distributors. In: CIGR Handbook of Agricultural Engineering, Vol. III: Plant Production Engineering, pp. 240-268.
86	[HG99]	HOFSTEE, J. W., GOENSE, D., 1999: Simulation of a Controller Area Network-based Tractor-Implement Data Bus according to ISO 11783. Journal of Agricultural Engineering Research 73(4), pp. 383-394.
87	[Hol09]	HOLPP, M., 2009: Präzision wird bezahlbar. dlz agrarmagazin 12, S. 62-67.
88	[IEC61162-3]	IEC, 2010: IEC 61162-3 Ed. 1.1. Maritime navigation and radiocommunication equipment and systems - Digital interfaces - Part 3: Serial data instrument network. Geneva.
89	[Int10]	INTEL CORPORATION, 2010: Intel® Atom™ Processor Z5xx Series Datasheet, Document Number: 319535-003US, Firmenschrift.
90	[Int11]	INTEL CORPORATION, 2011: Intel® Atom™ Processor N400 & N500 Series Datasheet - Volume 1, Document Number: 322847-003, Firmenschrift.
91	[Ise06]	ISERMANN, R., 2006: Sensors. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 32-52.
92	[ISO25119]	ISO, 2010: ISO 25119 Part 1-4. Tractors and machinery for agriculture and forestry - Safety-related parts of control systems. Geneva.

93	[ISO11783]	ISO, -: ISO 11783. Tractors and machinery for agriculture and forestry - Serial control and communication data network. Geneva.
94	[ISO11783-1]	ISO, 2007: ISO 11783-1:2007(E). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 1: General standard for mobile data communication. Geneva.
95	[ISO11783-5a]	ISO, 2001: ISO 11783-5:2001(E). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 5: Network management. Geneva.
96	[ISO11783-5b]	ISO, 2011: ISO 11783-5:2011(E). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 5: Network management. Geneva.
97	[ISO11783-7]	ISO, 2009: ISO 11783-7:2009(E). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 7: Implement messages application layer. Geneva.
98	[ISO11783-10]	ISO, 2009: ISO 11783-10:2009(E). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 10: Task controller and management information system data interchange. Geneva.
99	[ISO11783-10DAM1]	ISO, 2010: ISO 11783-10:2009 DAM 1 (Draft Amendment 1, Date: 2010-09-29). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 10: Task controller and management information systems data interface. Geneva.
100	[ISO11783-11]	ISO, 2011: ISO 11783-11:2011(E). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 11: Mobile data element dictionary. Geneva.
101	[ISO11783-13]	ISO, 2011: ISO 11783-13:2011(E). Tractors and machinery for agriculture and forestry - Serial control and communications data network - Part 13: File server. Geneva.
102	[NWIISO16867]	ISO, 2011: NWI ISO 16867. Wireless Communication in Agriculture; Standardization of wireless communication for Fleet Management, including implement synchronization. Geneva.
103	[ISO11898]	ISO, 1993: ISO 11898. Road vehicles - Interchange of digital information - Controller area network (CAN) for high speed communication. Geneva.
104	[ISOIEC7498-1]	ISO/IEC, 1994: ISO/IEC 7498-1. Information technology - Opens Systems Interconnection - Basic Reference Model: The Basic Model. Geneva.
105	[ISOIEC9075]	ISO/IEC, 1999: ISO/IEC 9075:1999. Information technology – Database languages – SQL. Geneva.

106	[KTRM08]	KLOSE, R., THIEL, M., RUCKELSHAUSEN, A., MARQUERING, J., 2008: Weedy - a sensor fusion based autonomous field robot for selective weed control. In: VDI-Berichte Nr. 2045, Tagung Landtechnik 2008 in Stuttgart-Hohenheim, S. 167-172.
107	[KFMS98]	KONDO, N., FUJIURA, T., MONTA, M., SEVILA, F., 1998: Robots in Bioproduction in Open Fields. In: Robotics for Bioproduction Systems, St. Joseph: ASAE.
108	[Kor03]	KORDUAN, P., 2003: Standardization in data management to increase interoperability of spatial precision agriculture data. In: Precision Agriculture. Proceedings of the 4th European Conference on Precision Agriculture in Berlin, Germany, pp. 323-328.
109	[Kor04]	KORDUAN, P., 2004: Metainformationssysteme für Precision Agriculture. Dissertation: Universität Rostock.
110	[Kra05]	KRALLMANN, J., 2005: Einsatz eines Multisensors für ein Condition Monitoring von mobilen Arbeitsmaschinen. Dissertation: Technische Universität Carolo-Wilhemina zu Braunschweig.
111	[Kra11]	KRATZ, H., 2011: Stand der Normungsarbeiten am 9. Mai 2011 - Normengruppe Landtechnik (NLA). In: Protokoll über die 24. Sitzung des Technischen Ausschusses „Elektronik“ am 10. Mai 2011. Frankfurt am Main: VDMA. (unveröffentlicht)
112	[KC93]	KRAUSE, P., CLARK, D., 1993: Representing Uncertain Knowledge: An Artificial Intelligence Approach. New York: Kluwer Academic Publishers.
113	[Kre91]	KREBS, V., 1991: Wissensbasierte Verfahren für die Automatisierungstechnik. In: VDI-Berichte Nr. 897: VDI/VDE-GMA-Aussprachetag Wissensverarbeitung in der Automatisierungstechnik in Langen, S. 1-25.
114	[Kur89]	KURBEL, K., 1989: Entwicklung und Einsatz von Expertensystemen. Eine anwendungsorientierte Einführung in wissensbasierte Systeme. Berlin, Heidelberg: Springer-Verlag.
115	[Lan94]	LANGMANN, R., 1994: Graphische Benutzerschnittstellen: Einführung und Praxis der Mensch-Prozess-Kommunikation. Düsseldorf: VDI-Verlag.
116	[LM05]	LENZ, J., MÜNCH, P., 2005: Software Development for Off-road Machines. In: VDI-Berichte Nr. 1895, Tagung Landtechnik 2005 in Hannover, pp. 315-322.
117	[Lew98]	LEWIS, T., 1998: Evolution of farm management information systems. Computers and Electronics in Agriculture 19, pp. 223-248.
118	[Li06]	LI, Y., 2006: Biosensors. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 52-93.

119	[Lie03]	LIEBLER, J., 2003: Feldspektrometrische Messungen zur Ermittlung des Stickstoffstatus von Winterweizen und Mais auf heterogenen Schlägen. Dissertation: Technische Universität München.
120	[LSHWSD00]	LINSEISEN, H., SPANGLER, A., HANK, K., WAGNER, P., STEINMAYR, T., DEMMEL, M., AUERNHAMMER, H., MANAKOS, I., SCHNEIDER, T., LIEBLER, J., 2000: Daten, Datenströme und Software in einem Informationssystem zur teilflächenspezifischen Pflanzenproduktion. Zeitschrift für Agrarinformatik 2, S. 36-43.
121	[Lin01]	LINSEISEN, H., 2001: Development of a precision farming informations system. In: Proceedings of the 3rd European Conference on Precision Agriculture, Montpellier, pp. 689-694.
122	[Lli01]	LLINAS, J., 2001: Assessing the Performance of Multisensor Fusion Processes. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.
123	[LSL02]	LUDOWICY, C., SCHWAIBERGER, R., LEITHOLD, P., 2002: Precision Farming: Handbuch für die Praxis. Frankfurt am Main: DLG-Verlag.
124	[Lug01]	LUGER, G. F., 2001: Künstliche Intelligenz: Strategien zur Lösung komplexer Probleme. München: Pearson Education Deutschland.
125	[Mad00]	MADAUSS, B. J., 2000: Handbuch Projektmanagement. Stuttgart: Schäffer-Poeschel.
126	[MSH04]	MAIDL, F.-X., SCHÄCHTL, J., HUBER, G., 2004: Strategies for site-specific nitrogen fertilization on winter wheat. In: Proceedings of the 7th International Conference on Precision Agriculture and Other Precision Resources Management, Conference in Minneapolis, 25-28 July, 2004, pp. 1938-1948.
127	[MMRD06]	MALEKI, M. R., MOUAZEN, A. M., RAMON, H., DE BAERDEMAEKER, J., 2006: Optimisation of Variable Rate Application System of Soil Phosphorus Using On-line VIS-NIR Sensor. In: VDI-Berichte Nr. 1958, World Congress "Agricultural Engineering for a Better World" in Bonn, pp. 295-296.
128	[Mar04]	MARTINUS, M. A., 2004: Funktionale Sicherheit von mechatronischen Systemen bei mobilen Arbeitsmaschinen. Dissertation: Technische Universität München.
129	[MH89]	MCGRAW, K. L., HARBISON-BRIGGS, K., 1989: Knowledge acquisition: principles and guidelines. Upper Saddle River, NJ, USA: Prentice-Hall.
130	[Min75]	MINSKY, M., 1975: A framework for representing knowledge. In: The Psychology of Computer Vision, New York: McGraw-Hill.

131	[Mis05]	MISTELE, B., 2005: Tractor based spectral reflectance measurements using an oligo view optic to detect biomass, nitrogen content and nitrogen uptake of wheat and maize and the nitrogen nutrition index of wheat. Dissertation: Technische Universität München.
132	[MS01]	MUNACK, A., SPECKMANN, H., 2001: Communication Technology is the Backbone of Precision Agriculture. CIGR Ejournal III, pp. 1-12.
133	[Mys11]	MYSQL, 2011: MySQL-Internetseite. http://www.mysql.de, Zugriff: 25.03.2011.
134	[NKB07]	NASH, E., KORDUAN, P., BILL, R., 2007: Optimising data flows in precision agriculture using open geospatial web services. In: Precision Agriculture '07: Proceedings of the 6th European Conference on Precision Agriculture, pp. 753-759.
135	[NS76]	NEWELL, A., SIMON, H., 1976: Computer Science as Empirical Inquiry: Symbols and Search. Communications of the ACM 19(3), pp. 113-126.
136	[NMD03]	NOACK, P. O., MUHR, T., DEMMEL, M., 2003: An Algorithm for Automatic Detection of Defective Yield Data. In: Precision Agriculture. Proceedings of the 4th European Conference on Precision Agriculture in Berlin, Germany, pp. 445-450.
137	[Oma06]	OMASA, K., 2006: Image Sensing and Phytobiological Information. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 217-231.
138	[OOS06]	OMASA, K., OKI, K., SUHAMA, T., 2006: Remote Sensing from Satellites and Aircraft. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 231-244.
139	[OLRD02]	OOMS, D., LEBEAU, F., RUTER, R., DESTAIN, M.-F., 2002: Measurements of the horizontal sprayer boom movements by sensor data fusion. Computers and Electronics in Agriculture 33, pp. 139-162.
140	[OP04]	OSTERHOLD, A., PISTOR, P., 2004: Datenbanksprache, Die SQL-Normen, Normen der Reihe ISO/IEC 9075. DIN-Mitteilungen 4, S. 27-36.
141	[OA04]	OSTERMEIER, R., AUERNHAMMER, H., 2004: Real-time process control for a sensor based fertilizer application system using multisensor data fusion. In: Engineering the future, AgEng Leuven 2004 (CD-ROM), pp. 1-8.
142	[OAD03]	OSTERMEIER, R., AUERNHAMMER, H., DEMMEL, M., 2003: Development of an in-field controller for an agricultural bus-system based on open source program library lbs-lib. In: Precision Agriculture. Proceedings of the 4th European Conference on Precision Agriculture in Berlin, Germany, pp. 515-520.

143	[OGAHG07]	OSTERMEIER, R., GALLMEIER, M., AUERNHAMMER, H., HERING, J., GLÜCK, S., 2007: "Intelligent bearing" - a new sensor for agricultural engineering applications. In: VDI-Berichte Nr. 2001, Tagung Landtechnik 2007 in Hannover, pp. 237-242.
144	[ORA06]	OSTERMEIER, R., ROGGE, H. I., AUERNHAMMER, H., 2006: Multisensor Data Fusion Implementation for a Sensor based Fertilizer Application System. In: Automation Technology for Off-Road Equipment, Proceedings of the 1-2 September 2006 Conference, Bonn, pp. 215-225.
145	[PJ07]	PAETOW, H., JÜRSCHIK, P., 2007: Datenerfassung - Notwendigkeit und Chance, DLG-Merkblatt 338.
146	[Pas11]	PASSMARK SOFTWARE, 2011: CPU Benchmark. http://www.cpubenchmark.net/cpu_lookup.php?cpu=Intel+Atom+N450+%40+1.66GHz, Zugriff: 05.05.2011.
147	[PS02]	PAUL, W., SPECKMANN, H., 2002: Measuring crop density and soil humidity by pulsed radar. In: Proceedings of the International Conference on Agricultural Engineering (Part 1), Budapest, pp. 14-15.
148	[PJS01]	PEDERSEN, H. H., JØRGENSEN, M. H., SØGAARD, H. T., 2001: Accuracy of four fertiliser spreaders for Precision Farming. In: Proceedings of the 3rd European Conference on Precision Agriculture, Montpellier, pp. 695-700.
149	[PPT10]	PETERS, O., PICKEL, P., TARASINSKI, N., 2010: Real Time Ethernet for Tractor Implement Communication. In: VDI Berichte Nr. 2111, Tagung Landtechnik 2010 in Braunschweig, pp. 285-292.
150	[PKL08]	PHELAN, J., KORMANN, G., LENZ, J., 2008: Creating Value in On-board Electronics: The Role of Sensors. In: VDI-Berichte Nr. 2045, Tagung Landtechnik 2008 in Stuttgart-Hohenheim, pp. 159-166.
151	[RMBVD06]	RAMON, H., MOSHOU, D., BRAVO, C., VRINDTS, E., DE BAERDEMAEKER, J., 2006: Sensing and Information Handling for Crop Protection. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 279-289.
152	[RFS07]	RATHMANN, S., FISCHERKELLER, R., SCHWEIKER, A., 2007: Latest Trends in Automotive Electronic Systems, Highway Meets Off-Highway. In: VDI-Berichte Nr. 2001, Tagung Landtechnik 2007 in Hannover, pp. 287-293.
153	[Rec10]	RECKLEBEN, Y., 2010: Sensorschemme - Stickstoffdüngung mit Sensoren - welche gibt es und was können sie?. Neue Landwirtschaft 4, S. 81-84.
154	[RK96]	REITZ, P., KUTZBACH, H. D., 1996: Investigations on a particular yield mapping system for combine harvesters. Computers and Electronics in Agriculture 14, pp. 137-150.

155	[RL94]	REMBOLD, U., LEVI, P., 1994: Realzeitsysteme zur Prozessautomatisierung. München, Wien: Carl Hanser Verlag.
156	[Reu97]	REUSCH, S., 1997: Entwicklung eines reflexionsoptischen Sensors zur Erfassung der Stickstoffversorgung landwirtschaftlicher Kulturpflanzen. Dissertation: Christian-Albrechts-Universität, Kiel.
157	[RHHKH08]	RISIUS, H., HAHN, J., HUTH, M., KORTE, H., HARMANN, T. L., 2008: Nah-Infrarot-Spektroskopie zur qualitätsgeleiteten Trennung des Getreidegutstroms. In: VDI-Berichte Nr. 2045, Tagung Landtechnik 2008 in Stuttgart-Hohenheim, S. 187-192.
158	[RL11]	ROBERT, M., LANG, T., 2011: Swarm Intelligence Algorithms for Agricultural In-Field Logistics. In: VDI-Berichte Nr. 2124, Tagung Landtechnik 2011 in Hannover, pp. 281-286.
159	[Rog04]	ROGGE, H. I., 2004: Expert System for Application Implement Control. Diplomarbeit: Technische Universität München, Fachgebiet Technik im Pflanzenbau, Weihenstephan.
160	[Rot06]	ROTHMUND, M., 2006: Technische Umsetzung einer Gewannebewirtschaftung als „Virtuelle Flurbereinigung“ mit ihren ökonomischen und ökologischen Potenzialen. Dissertation: Technische Universität München.
161	[RA04]	ROTHMUND, M., AUERNHAMMER, H., 2004: A web based information management system for process data designed with open source tools. In: Engineering the future, AgEng Leuven 2004 (CD-ROM), pp. 1-8.
162	[RZAD03]	ROTHMUND, M., ZIPPRICH, M., AUERNHAMMER, H., DEMMEL, M., 2003: Zugkraftmessung bei der Bodenbearbeitung als ergänzende Information zur Standortbeschreibung. In: VDI-Berichte Nr. 1798, Tagung Landtechnik 2003 in Hannover, S. 305-310.
163	[RRGAHG11]	RUSCH, C., REINECKE, M., GROTHAUS, H.-P., AUTERNMANN, L., HARTANTO, R., GEORGIEW, E., 2011: Wireless communication on the field following ISO 11783 for autonomous process planning and controlling of cooperating mobile agricultural machines. In: VDI-Berichte Nr. 2124, Tagung Landtechnik 2011 in Hannover, pp. 299-306.
164	[SAEJ1939]	SAE, 2003: SAE J1939: Recommended Practice for a Serial Control and Communications Vehicle Network. Warrendale.
165	[SAEJ1939-02]	SAE, 2006: SAE J1939-02: Agricultural and Forestry Off-Road Machinery Control and Communication Network. Warrendale.
166	[SJBGMM03]	SCHMIDHALTER, U., JUNGERT, S., BREDEMEIER, C., GUTSER, R., MANHART, R., MISTELE, B., GERL, G., 2003: Field-scale validation of a tractor based multispectral crop scanner to determine biomass and nitrogen uptake of winter wheat. In: Precision Agriculture. Proceedings of the 4th European Conference on Precision Agriculture in Berlin, Germany, pp. 615-619.

167	[Sch01]	SCHMIDHALTER, U., 2001: Geophysikalische Kartierung von Bodeneigenschaften für die teilflächenspezifische Bewirtschaftung. Landtechnik 6, S. 417.
168	[SYM06]	SCHULZE LAMMERS, P., YURUI, S., MA, D., 2006: A Combined Horizontal Penetrometer for Transient Detection of Soil Water Content and Mechanical Resistance. In: VDI-Berichte Nr. 1958, World Congress "Agricultural Engineering for a Better World" in Bonn, pp. 289-290.
169	[Sch05]	SCHUTTE, B., 2005: Bestimmung von Bodenunterschieden durch Zugkraftmessungen bei der Bodenbearbeitung. Dissertation: Universität Hohenheim.
170	[SLMPW05]	SEETHALER, C., LOVRIC, T., MICKISCH, W., PLANKENSTEINER, M., WEISSENGRUBER, M., 2005: Improved Electronic Architecture in Agricultural Vehicles with regard to the TTA-Group Steer-by-Wire Working Group. Enabling Improved Architectures with Time-Triggered Communication. In: VDI-Berichte Nr. 1895, Tagung Landtechnik 2005 in Hannover, pp. 283-290.
171	[SNS03]	SELIGE, T., NÄTSCHER, L., SCHMIDHALTER, U., 2003: Spatial detection of topsoil properties using hyperspectral sensing. In: Precision Agriculture. Proceedings of the 4th European Conference on Precision Agriculture in Berlin, Germany, pp. 633-638.
172	[Sin06]	SINDIR, K. O., 2006: Management and Decision Support Systems. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 355-367.
173	[SAD01]	SPANGLER, A., AUERNHAMMER, H., DEMMEL, M., 2001: Stimulating use of open communication standards in agriculture (DIN 9684 and ISO 11783) with capable Open Source Program Library as possible reference implementation. In: Proceedings of the 3rd European Conference on Precision Agriculture, Montpellier, pp. 719-724.
174	[SJ99]	SPECKMANN, H., JAHNS, G., 1999: Development and application of an agricultural BUS for data transfer. Computers and Electronics in Agriculture 23, pp. 219-237.
175	[Sri97]	SRIRAM, R. D., 1997: Intelligent systems for engineering: a knowledge-based approach. Berlin, Heidelberg, New York: Springer-Verlag.
176	[SB01]	STEINBERG, A. N., BOWMAN, C. L., 2001: Revisions to the JDL Data Fusion Model. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.
177	[SBW98]	STEINBERG, A. N., BOWMAN, C. L., WHITE, JR., F. E., 1998: Revisions to the JDL Data Fusion Model. In Proceedings of the 3rd NATO/IRIS Conference in Quebec City, Canada.

178	[SRA06]	STEINBERGER, G., ROTHMUND, M., AUERNHAMER, H., 2006: Agricultural Process Data Service (APDS). In: VDI-Berichte Nr. 1958, World Congress "Agricultural Engineering for a Better World" in Bonn, pp. 271-272.
179	[Ste02]	STEINMAYER, T., 2002: Fehleranalyse und Fehlerkorrektur bei der lokalen Ertragsermittlung im Mähdrescher zur Ableitung eines standardisierten Algorithmus für die Ertragskartierung. Dissertation: Technische Universität München.
180	[SMFB99]	STONE, M. L., MCKEE, K. D., FORMWALT, C. W., BENNEWEIS, R. K., 1999: ISO 11783: An Electronic Communications Protocol for Agricultural Equipment. ASAE Distinguished Lecture Series No. 23, pp. 5-17.
181	[SDK01]	Sudduth, K. A., Drummond, S. T., Kitchen, N. R., 2001: Accuracy isssues in electromagnetic induction sensing of soil electrical conductivity for precision agriculture. Computers and Electronics in Agriculture 31, pp. 239-264.
182	[Tor94]	TORNOW, W., 1994: Taschenbuch der Nachrichtentechnik: Ingenieurwissen für die Praxis. Berlin: Fachverlag Schiele & Schön.
183	[Ull03]	ULLENBOOM, C., 2003: Java ist auch eine Insel: Programmieren für die Java 2-Plattform in der Version 1.4. Bonn: Galileo Press.
184	[VDR06]	VAN LIEDERKERKE, P., DE BAERDEMAEKER, J., RAMON, H., 2006: Fertilizer Application Control. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 273-278.
185	[VV06]	VAN STRATEN, G., VAN WILLIGENBURG, L. G., 2006: Control and Optimization. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 124-138.
186	[VTB06]	VELLIDIS, G., TUCKER, M., BEDNARZ, C., 2006: A Real-Time Smart Sensor Array for Scheduling Irrigation. In: Automation Technology for Off-Road Equipment, Proceedings of the 1-2 September 2006 Conference, Bonn, pp. 281-288.
187	[W3C09]	W3C - WORLD WIDE WEB CONSORTIUM, 2009: OWL 2 Web Ontology Language Document Overview, W3C Recommendation 27 October 2009.
188	[W3C10]	W3C - WORLD WIDE WEB CONSORTIUM, 2010: RIF Production Rule Dialect, W3C Recommendation 22 June 2010.
189	[WH01]	WALTZ, E., HALL, D. L., 2001: Requirements Derivation for Data Fusion Systems. In: Handbook of multisensor data fusion, Boca Raton, London, New York, Washington, D.C.: CRC Press.
190	[WL90]	WALTZ, E., LLINAS, J., 1990: Multisensor Data Fusion. Norwood: Artech House.
191	[WHL83]	WATERMAN, D., HAYES-ROTH, F., LENAT, D., 1983: Building Expert Systems. Reading: Addison-Wesley Publishing.

192	[Wei06]	WEIGERT, G., 2006: Data Mining und Wissensentdeckung im Precision Farming - Entwicklung von ökonomisch optimierten Entscheidungsregeln zur kleinräumigen Stickstoff-Ausbringung. Dissertation: Technische Universität München.
193	[WFCHH99]	WHITE, S., FISHER, M., CATTELL, R., HAMILTON, G., HAPNER, M., 1999: JDBC™ API Tutorial and Reference, Universal Data Access for the Java™ 2 Platform. Boston, San Francisco, New York: Addison-Wesley.
194	[WK07]	WILD, K., KORMANN, G., 2007: Entwicklung eines Nah-Infrarot-Sensors für Landmaschinen. Landtechnik SH, S. 276-277.
195	[WRK98]	WOLLRING, J., REUSCH, S., KARLSSON, C., 1998: Variable nitrogen application based on crop sensing. International Fertiliser Society Proc. 423, 28 p.
196	[ZXPM06]	ZAZUETA, F. S., XIN, J., PEREIRA, L. S., MUSY, A., 2006: Information Technologies in Water Management. In: CIGR Handbook of Agricultural Engineering, Vol. VI: Information Technology, pp. 401-414.

A Anhang

A.1 Versuchsfeld D4 der TU München

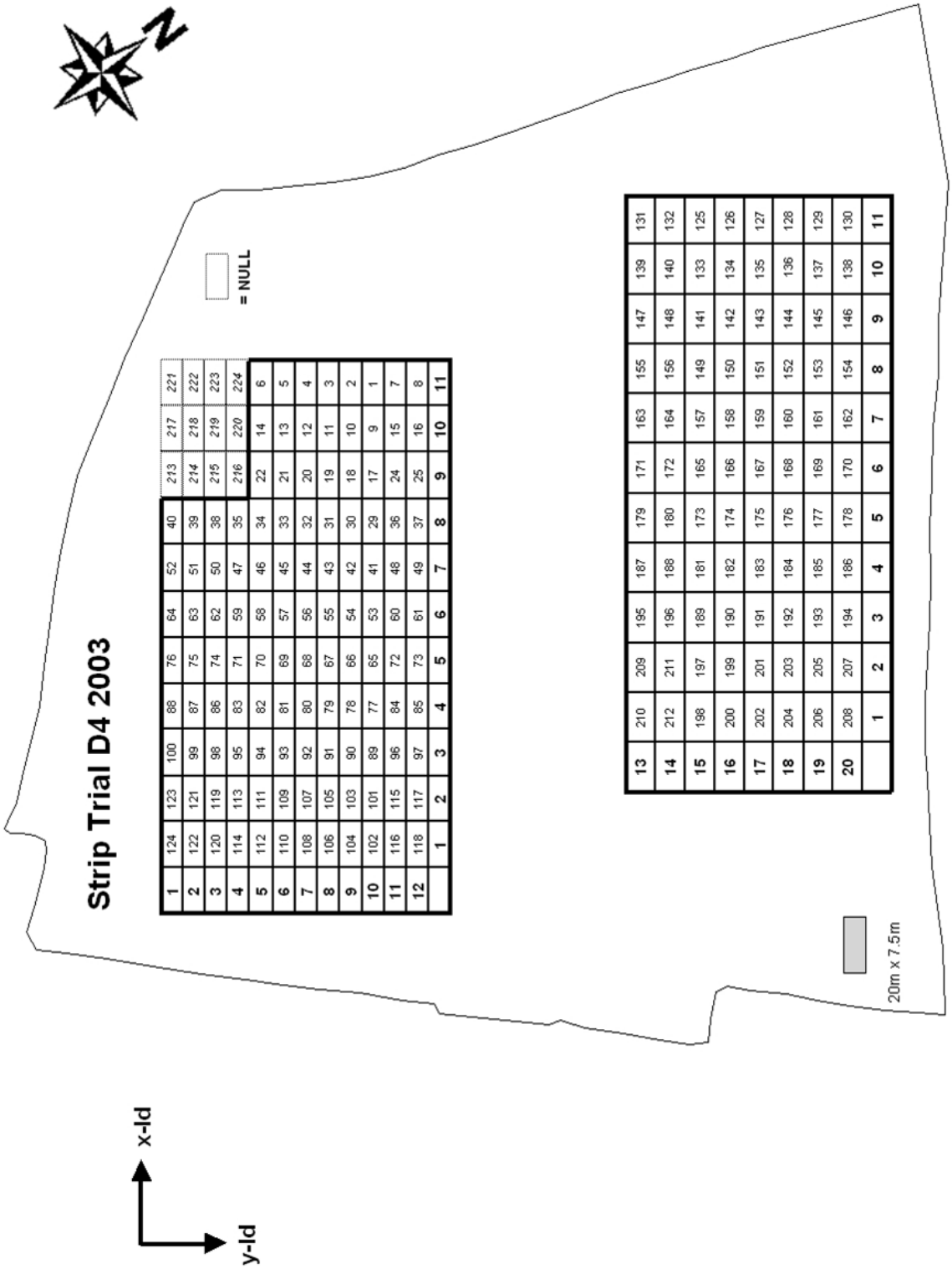


Abbildung A.1: Versuchsfeld D4 der TU München (nach [Wei06])

A.2 Entscheidungsbaum zur zweiten N-Applikation mit dem Attribut REIP_2, Versuchsfeld D4 der TU München, Versuchsjahr 2003

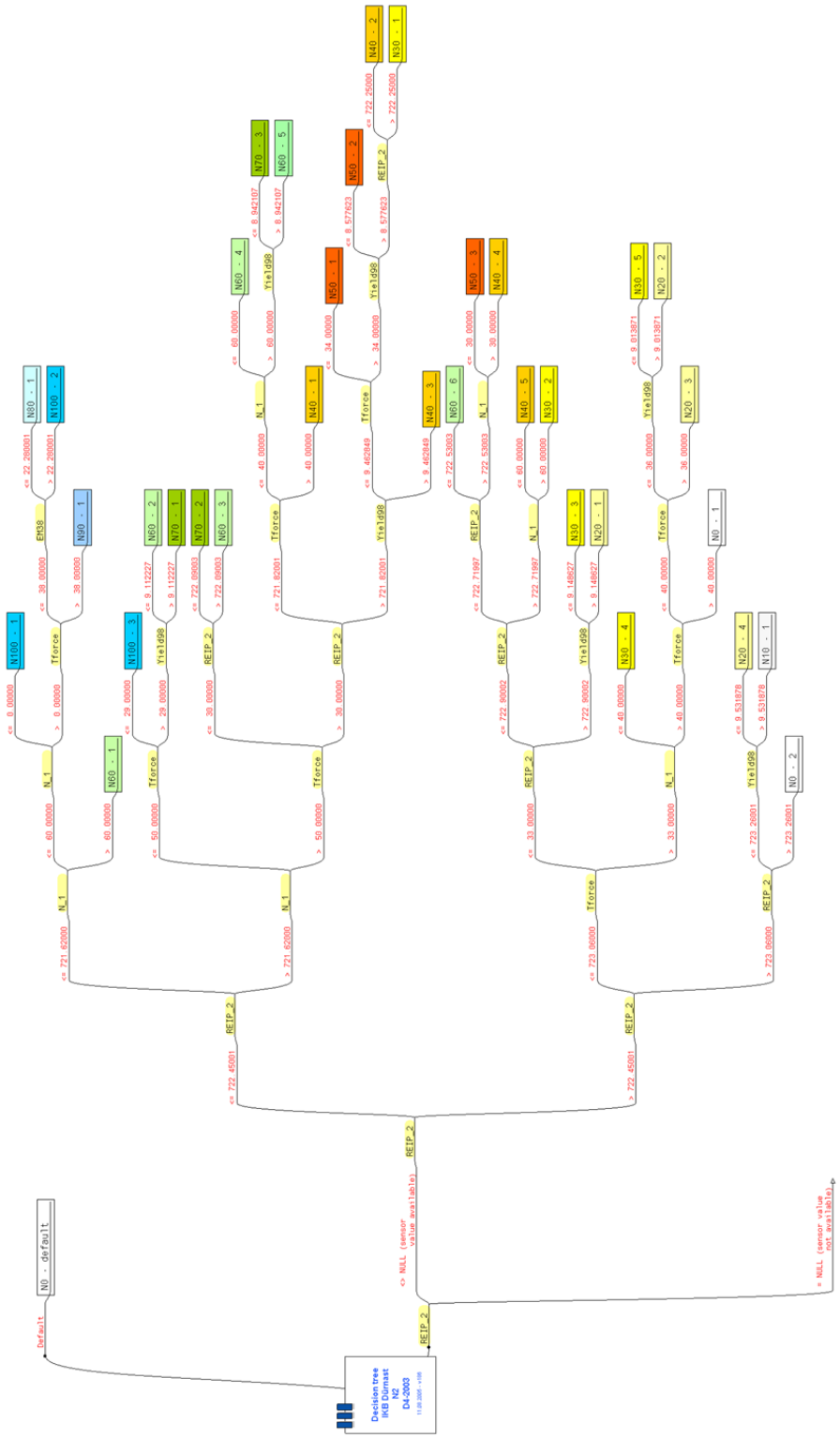


Abbildung A.2: Entscheidungsbaum zur zweiten N-Applikation mit dem Attribut REIP_2, Versuchsfeld D4 der TU München, Versuchsjahr 2003

A.3 Entscheidungsbaum zur zweiten N-Applikation ohne das Attribut REIP_2, Versuchsfeld D4 der TU München, Versuchsjahr 2003

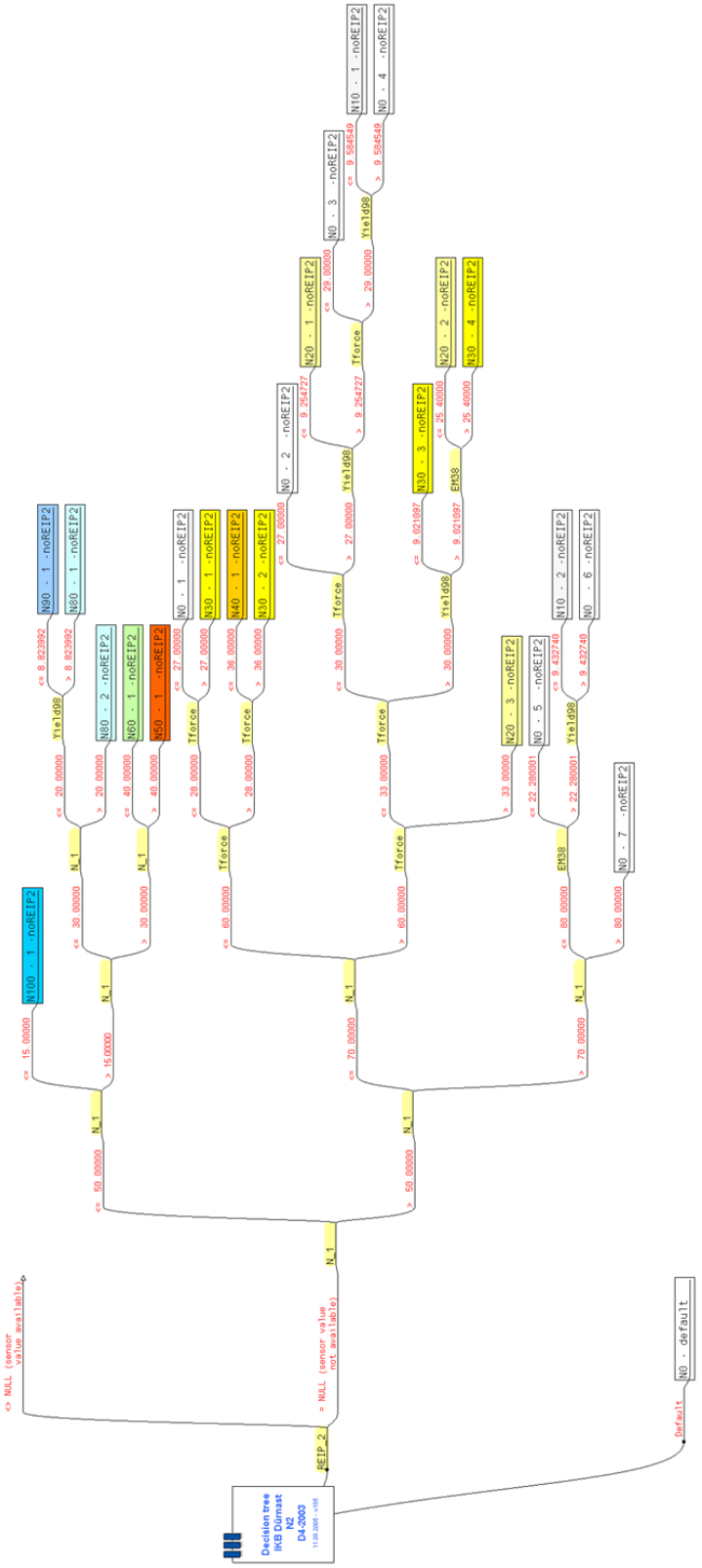


Abbildung A.3: Entscheidungsbaum zur zweiten N-Applikation ohne das Attribut REIP_2, Versuchsfeld D4 der TU München, Versuchsjahr 2003

A.4 Implementierungsdetails der Simulation - Sensor-Ansatz mit Kartenüberlagerung

A.4.1 Grundstruktur und Ablaufsteuerung in JESS

JESS kann vom Wissensingenieur auf zwei sich überlagernde Arten eingesetzt werden:

- Die eine Anwendung ist die reine Nutzung als *Rule Engine*, also eine Softwarekomponente die regelbasiertes Schlussfolgern und den Bau eines Expertensystems gestattet. Diese *Rule Engine* lässt sich in ein übergeordnetes Softwaresystem einbetten oder Grundlage eines eigenständigen Expertensystems sein.
- Anderseits kann JESS auch als universelle Programmiersprache mit direktem Zugriff auf alle Java-Klassen- und -bibliotheken genutzt werden.

Aufgrund des Interpreter-Charakters lässt sich JESS wie eine dynamische Skriptsprache nutzen und bietet sich gerade deswegen für eine evolutionäre und *Rapid Prototyping* Softwareentwicklung an. FRIEDMAN-HILL (2003) [Fri03][80] unterscheidet grob die vielfältigen Architekturmöglichkeiten, ein System mit JESS zu entwickeln und mit Java zu verschränken, in folgender Weise:

- Ausschließlich JESS-Code, kein Java-Code,
- Ausschließlich JESS-Code, aber das Programm greift auf Java APIs zu,
- Hauptsächlich JESS-Code, aber mit in Java-codierten anwendungsspezifischen JESS-Erweiterungsfunktionen,
- Zur Hälfte JESS-Code mit einem substantiellen Anteil von Java-Code, der anwendungsspezifische Funktionen und APIs bereitstellt; JESS stellt die `main()` -Funktion (Den Einstiegspunkt für das Programm) zur Verfügung,
- Zur Hälfte JESS-Code mit einem substantiellen Anteil von Java-Code, der anwendungsspezifische Funktionen und APIs bereitstellt; jedoch wird die `main()` -Funktion (Der Einstiegspunkt für das Programm) im Java-Teil angeordnet,
- Größtenteils Java-Code, der JESS-Code zur Laufzeit lädt,
- Ausschließlich Java-Code, der JESS-Funktionalität nur durch dessen Java (Bibliothek) API mit einbindet.

Für die Erstellung der Simulation wurde der Weg beschritten, dass der größere Teil in JESS-Sprache realisiert wird, die `main()`-Funktion dort angesiedelt wird, jedoch der größte Teil der GUI, die Nachbildung der Prozessumgebung und die Datenbankanbindung in Java programmiert wurden. Dieser Ansatz erlaubt zum einen die Flexibilität von JESS bei der evolutionären Programmentwicklung als auch beim Testen des Fusionsprozesses zu nutzen, anderseits ermöglicht es die effiziente Nutzung von Java-Bibliotheken zur GUI-Programmierung und zur Datenbankanbindung.

[80] A. a. O., S. 37-38

Somit liegt der bisher vor allem aus der Anwendersicht beschriebenen Simulation der in Abbildung A.4 dargestellte grundlegende Systemaufbau zugrunde.

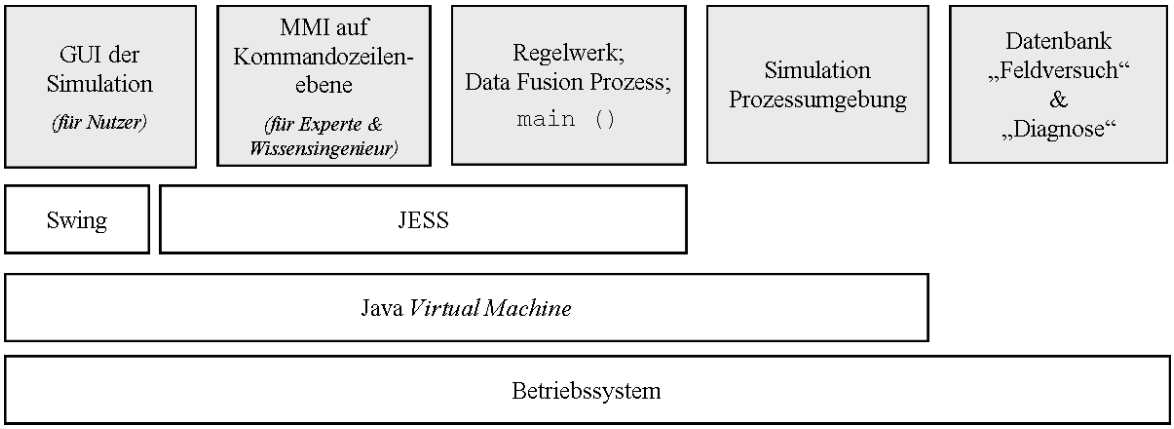


Abbildung A.4: Systemaufbau der Simulation als Schichtenmodell

Auf unterster Ebene abstrahiert ein Betriebssystem die Rechner-Hardware und bietet grundlegende Funktionen an. Als Rechner-Hardware wurden Notebooks mit einer X86-Architektur gewählt sowie eines der beiden Betriebssysteme Microsoft Windows 2000 Professional (Microsoft Corporation, USA) oder Microsoft Windows XP Home (Microsoft Corporation, USA).

Auf der nächsten Ebene kommt eine Java-Laufzeitumgebung mit einer Java™ 2 SE Virtual Machine für Windows 2000 bzw. XP zum Einsatz.

Die Java-Laufzeitumgebung stellt wiederum die Basis für die Expertensystem-Shell JESS in der Version 6.14 dar. Die Grafikbibliothek Swing für Java zum Programmieren von grafischen Benutzeroberflächen ist ebenfalls Bestandteil der Java-Laufzeitumgebung.

Auf der obersten Ebene sind die einzelnen Komponenten der Anwendung/Applikation angeordnet. So nutzt die Komponente der grafischen Benutzerschnittstelle hauptsächlich Funktionen von Swing, aber greift auch auf JESS-Funktionalität zurück. Hingegen nutzt die Kommandozeilenumgebung der Simulation nur JESS-Funktionen.

Das Kernstück der Simulation, der Datenfusionsprozess, (und die Kernelemente eines Expertensystems (wie z.B. Wissensbasis/Inferenzeinheit oder Problemlösungskomponente, Interviewkomponenten (a, GUI, b, Werte aus Datenbank), Wissenserwerbkomponente, Erklärungskomponente)) basiert vollständig auf der JESS-Expertensystem-Shell-Schnittstelle. Die Simulation der Prozessumgebung hingegen wird bis auf die entsprechende Einbindung in den Datenfusionsprozess auf Java-Basis implementiert. Auch die Datenbankfunktionalität greift nur für die Anbindung an die restlichen Anwendungskomponenten auf wenige Java- und JESS-Schnittstellen zurück. Signal- und Datenflüsse bzw. Interprozesskommunikation

werden, soweit sie für das Verständnis nötig erachtet werden, in den folgenden Unterkapiteln aufgezeigt.

Der grobe Aufbau der statischen JESS-Programmstruktur wird in Abbildung A.5 dargestellt und ist durch JESS als Interpreter-Sprache mit der Fähigkeit eines „*Late Binding"* geprägt.

1. Initialisierung des Startzustandes
2. Import von Java-Bibliotheken (z.B. Swing, AWT)
3. Definition allgemeiner Funktionen und Variablen
4. Abfrage der Betriebsparameter
5. *Timer*-Funktionen definieren und initialisieren
6. Definition und Initialisierung der GUI
7. Definition und Instanziierung der Datenbankanbindung
8. Definition des Regelwerk-Moduls MAIN mit Definition und Instanziierung der Prozessumgebung (*Shadow Facts*)
9. Definition der weiteren Regelwerkmodule • *CROP_PRODUCTION* • *CROP_PRODUCTION-SUM_UP* • *CONSTRAINTS* • *CONSTRAINTS-SUM_UP* • *AG_ENGINEERING* • *AG_ENGINEERING-SUM_UP* • *EMERGENCY_STOP* • *SUM_UP* • *CLEAN_UP* • *CLEAN_UP_DEMS*

Abbildung A.5: Statische Programmstruktur des JESS-Codes

Von einem Abdruck des Programm-*Listing* (JESS und Java-Code) in dieser Veröffentlichung wird aufgrund des Umfangs von mehr als 100 Seiten gemäß dem vorliegenden Textformat Abstand genommen und nur typische Strukturen, Funktionsweisen und angewandte Grundprinzipien aufgeführt bzw. auf die entsprechenden Unterkapitel verwiesen.

Die aufgezeigten Programmabschnitte enthalten folgende Funktionalität:

1. Zu Beginn des Programms wird ein definierter Anfangszustand erzeugt, bei dem mit `clear` sichergestellt wird, dass eine leere Faktenbasis vorliegt. Mit dem `reset`-Befehl wird dann ein `(initial fact)` erzeugt, was speziell im Zusammenhang mit der

Einbindung der Prozessumgebung von Bedeutung ist. Daraufhin wird mit der `(set-reset-globals FALSE)` Funktion verhindert, dass globale Variablen bei einem erneuten `reset` reinitialisiert werden würden. Dies hätte negative Auswirkungen auf die Datenbankanbindung und die grafische Benutzerschnittstelle.

2. Im nächsten Programmabschnitt werden die Java-Bibliotheken `javax.swing.*`, `import javax.swing.Timer`, `import java.awt.*`, `import java.awt.event.*` in JESS importiert, die für die grafische Benutzerschnittstelle und eine ereignisorientierte Ablaufsteuerung von Nöten sind.

3. In einem nächsten Schritt werden Funktionen und Variablen definiert, die von allgemeiner Bedeutung sind. Hierzu zählen global definierte `(defglobal)` Variablen für die Dokumentation und für die Zeitmessung zum Zwecke des Testens.

4. Die Funktionen `deffunction FUNKTIONSNAME ()` umfassen vor allem Funktionen für die einfache Texteingabe von der JESS –Kommandozeile, Funktionen um Betriebsparameter von der Kommandozeile abzufragen und Funktionen für Zeitmessungen.

5. Im anschließenden Programmabschnitt werden dann zuvor definierte Funktionen genutzt, um über die JESS-Kommandozeile Betriebsparameter bzw. Konfigurationsmöglichkeiten der Simulation abzufragen (und je nach Nutzereingabe entsprechend einzustellen.) Anschließend folgt die Definition und Initialisierung einer *Timer*-Funktionalität und der zugehörigen Aktionen, die bei Eintritt des zyklischen Ereignisses ausgelöst werden sollen. Im zugehörigen Aktionsteil werden Funktionen aufgerufen, die die Einstellwerte des „*User Controls Panels"* auslesen, für die anderen Prozessdaten und Kartenwerte Zufallswerte generieren, Zeitmessungen starten und stoppen sowie den Schlussfolgerungs-/Data Fusion-Prozess auslösen. Die aufgeführten Funktionen werden erst an späteren Stellen im Programmcode definiert, können jedoch aufgrund der *late binding*-Funktionalität von JESS bereits hier genutzt werden.

6. Im folgenden Programmteil wird in einer „*bottom up*"-Verfahrensweise die GUI mit ihrem statischen Aufbau und dynamischen Verhalten aufgebaut. Das Unterkapitel A.4.5 beschreibt die Strukturen und ausgewählte Funktionen.

7. Der dann folgende Programmabschnitt definiert und erzeugt die Datenbankanbindung, der ebenfalls ein eigenes Unterkapitel A.4.4 gewidmet ist.

8. Erst nach diesem Programmpunkt startet die Implementierung der regelbasierten MSDF-Prozesse, in dem die einzelnen Ergebnisse des *Problem Solving Level* und des *Knowledge-base Level* in JESS kodiert werden. Dazu erfolgt eine äquivalente Aufteilung in Module (`defmodule` MODUL-NAME). Begonnen wurde dazu mit dem Module „*MAIN"*, das grundlegende Variable definiert, die nötig sind, um die simulierte Prozessumgebung anzukoppeln. Des Weiteren wird das *Knowledge Base Template*

(*Template* für die Faktenbasis) für die Sollwertempfehlung definiert `((deftemplate N_recommendation (slot setpoint) (slot explanation))`. Es folgen Funktionsimplementierungen für Ein- und Ausgabe auf der Kommandozeilenumgebung, für die Aktivierung der Debug-Funktionalität der Prozessumgebung und für die Funktion `run-system`, die auf dem JESS-Kommando `(run)` basiert.

9. In den folgenden Programmabschnitten werden die einzelnen Module „*CROP_PRODUCTION“*, „*CROP_PRODUCTION-SUM_UP“*, „*CONSTRAINTS“*, „*CONSTRAINTS-SUM_UP“*, „*AG_ENGINEERING“*, „*AG_ENGINEERING-SUM_UP“*, „*SUM_UP“*, „*EMERGENCY_STOP“*, „*CLEAN_UP“* und „*CLEAN_UP_DEMS“* (DEMS = *Disable EMergency Stop*) implementiert.

Für die grundlegende Ablaufstruktur der Simulationsimplementierung gelten folgende Prinzipien:

- JESS in seiner Funktion als Expertensystem-Shell ist die übergeordnete Ebene, von dessen Kommandozeilenumgebung der gesamte Funktionsumfang auch zur Laufzeit stets genutzt werden kann. Zusätzlich können auf gleicher Weise auf alle bei der Simulationsimplementierung definierten anwendungsspezifischen Funktionen zugegriffen werden.
- Auf der nächsten Ebene lassen sich die allgemeinen Swing-Funktionen für grafische Benutzerschnittstellen nutzen, wie z.B. Beispiel das Verschieben oder das Schließen des Anwendungsfenster, in dem die GUI der „*In-field Controller*“-Simulation läuft.
- Eine Stufe tiefer stehen dem Nutzer jederzeit die Eingabemöglichkeiten der realisierten GUI zur Verfügung mit seinen *Sliders*, *Radiobuttons* und den Tasten für die in den *use cases* definierten Anwendungsfälle der Simulation. Einerseits werden dadurch einzelne Wertelesevorgänge und ein einmaliger Durchlauf des Schlussfolgerungsverlaufes durchgeführt, also vergleichbar einem Unterprogrammaufruf. Anderseits löst die „*Start“*-Taste auch ein zyklisches Ablaufverhalten aus, das gemäß der Default-Einstellung mit 2 Hz läuft bis es wieder mit der „*Stop“*-Taste komplett beendet wird oder durch die „*ON/OFF“*-Taste im simulierten Notaus pausiert. Diese Funktionalität basiert auf einer in Java definierten Klasse für einen zyklischen Prozess („*Timer*“) und Javas Modell zur Ereignisbehandlung. Dementsprechend wurde ein *Timer*-Objekt erzeugt und der globalen JESS Variable `?*timer1*` zugewiesen. Der „*Timer*“ wurde mit einem Wert für die Zykluszeit von 500 ms initialisiert. Weiterhin wird ein Exemplar eines sogenannten Interessenten mit umgesetzter Java-Schnittstelle (*Interface*) *ActionListener* zugeordnet, das die entsprechende durchzuführende Aktion bei Ereigniseintritt ausführt. Die Formulierung in JESS lautet:

```
(defglobal ?*timer1* = (new Timer 500
    (new  jess.awt.ActionListener timer-function (engine))))
```

Auf weitere Einzelheiten der Ereignisbehandlung wird im Kapitel A.4.5 im Zusammenhang mit den Schaltflächen noch näher eingegangen.

Eingebettet in diese übergeordneten Ablaufsteuerungsstrukturen ist eine weitere Ebene einer Ablaufsteuerung, die auf der Ebene des Regelwerkes implementiert wird. Diese gibt auch den Prozess des Schlussfolgerungsverlaufes wieder. Der Nachrichtenaustausch auf dieser Ebene zwischen den einzelnen Modulen erfolgt über globale Variablen bzw. Fakten in JESS, deren jeweilige Bezeichnung bereits in den Kapiteln *Problem Solving Level* (vgl. Kap. 5.2.2.3) und *Knowledge-base Level* (vgl. Kap. 5.2.2.4) eingeführt und benutzt wurde. Die typische Umsetzung in JESS folgt dem Schema für beispielsweise *CROP_PRODUCTION::N_recommendation (setpoint*, *explanation),* was in JESS durch `(deftemplate CROP_PRODUCTION::N_recommendation (slot setpoint) (slot explanation))` umgesetzt wird.

„`CROP_PRODUCTION::`“ bedeutet, dass es sich um ein Fakt des Moduls „*CROP_PRODUCTION*“ handelt. Jedoch lässt es sich auch aus anderen Modulen darauf zugreifen.

Der Ablauf des Schlussfolgerungszyklus lässt sich gut anhand eines Zustandsdiagramms (Abb. A.6) wiedergeben. Die einzelnen Zustände stehen jeweils für ein Modul des Regelwerkes. Für jeden Zustand bzw. jedes Regelwerkmodul sind an den Transitionspfeilen die auslösenden Ereignisse und die zum Zustandsübergang auszuführenden Aktionen angegeben. Mit Eintritt in einen Zustand wird der Inferenz-Prozess mit den Regeln des Wissensmoduls weitergeführt. Innerhalb des Moduls läuft dieser Inferenz-Prozess bis keine aktivierte Regel mehr vorhanden ist. Da das System jedoch abgesehen von dem Zustand „*MAIN“* nicht in einem Zustand verharren, sondern automatisch zum nächsten Zustand wechseln soll, wurde eine entsprechende Konstruktion für einen Fokuswechsel implementiert. Die Regel, die zuletzt in einem Modul aktiv sein sollte, bewirkt den Fokuswechsel. Dass diese Regel als letzte gefeuert wird, wird unter Ausnutzung einer Prioritätsangabe erreicht, indem die sogenannte „*salience*“ niedriger als der Standardwert 0 definiert wird. Das Standardkonstrukt für einen regelbasierten Zustandswechsel von „*MODUL_1*“ auf „*MODUL_2*“ lautet:

```
(defrule MODUL_1::Control_flow "Control flow"
   (declare (salience -10))
    =>
    then  (focus MODUL_2) )
```

In dem „*MAIN*“-Zustand wird auf die Ereignisse eines *Button*- oder *Timer*-Ereignisses oder einer Nutzereingabe über die I/O-Schnittstelle reagiert und ein entsprechender Fokuswechsel initiert.

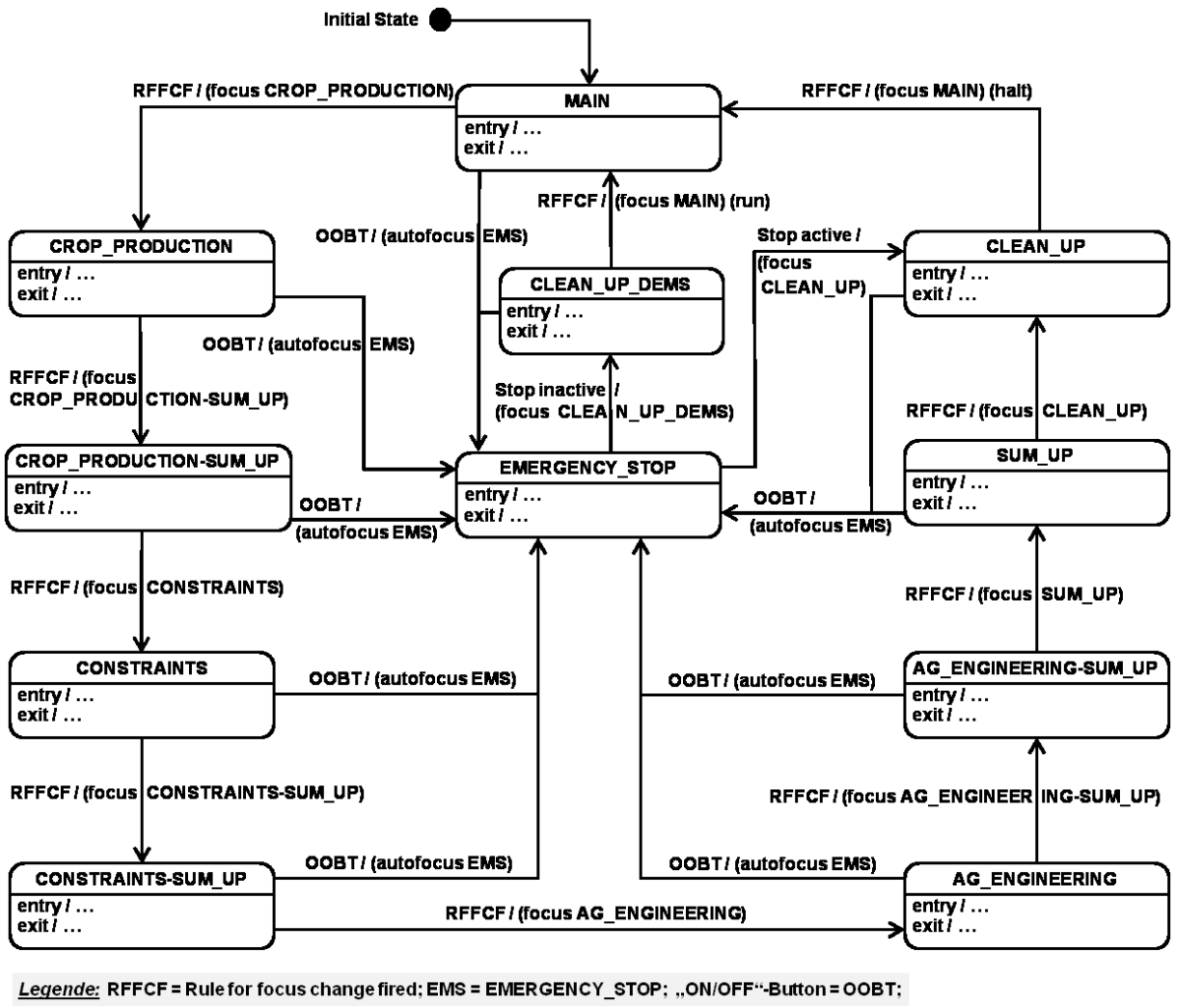


Abbildung A.6: Zustandsdiagramm für den Schlussfolgerungsprozess

Das Zustandsdiagramm zeigt einerseits einen sequentiellen Ablauf beginnend mit „*MAIN*“ über die einzelnen Module „*CROP_PRODUCTION*“, „*CROP_PRODUCTION-SUM_UP*“, „*CONSTRAINTS*“, „*CONSTRAINTS-SUM_UP*“, „*AG_ENGINEERING*“, „*AG_ENGINEERING-SUM_UP*“, „*SUM_UP*“ und „*CLEAN_UP*“ wieder zurück zum „*MAIN*“-Ausgangszustand. Andererseits wird bei Aktivierung des ON/OFF-*Buttons* in den „*EMERGENCY_STOP*“-Zustand gewechselt, der bei anhaltender Notausbedingung über „*CLEAN_UP*“ und bei „gelöschtem“ Notaussignal über „*CLEAN_UP_DEMS*“ verlassen wird.

JESS stellt die Befehle `(run)` und `(halt)` bereit, um den Inferenz-Prozess zu starten bzw. zum Stoppen zu bringen. Jedoch hebt FRIEDMAN-HILL (2003) [Fri03] hervor: *„…. Jess activates a rule only once for a given working memory state. Once the rule has fired, it will*

not fire again for the same list of facts. ...“[81] Normalerweise lässt sich die Faktenbasis sehr einfach mit den `(clear)`-Befehlen zurücksetzen und dann mit neuen Fakten bestücken. Die gewählte Struktur zur Simulation einer Prozessumgebung und die Anforderung, einen ereignisorientierten Programmablauf zu realisieren, erfordert aber einen anderen Mechanismus, der die Faktenbasis nicht stets komplett löscht bzw. darauf angewiesen ist, dass sich jeder der Sensor- und Kartenwerte sich bei jedem neuen Durchlauf ändert. Dieser Fall ist eher die Ausnahme als die Regel. Daher wurde zu der (Hilfs-)Konstruktion gegriffen, einen Fakt „*trigger*“ einzuführen, der bei den entsprechenden Regeln auf der Bedingungsseite mit abgeprüft wird und somit die damit versehenen Regeln für jeden Durchlauf aktivierbar machen kann, wenn vor Beginn des Inferenz-Prozesses ein entsprechender Fakt „*trigger*“ in die Faktenbasis mittels der `(assert)`-Funktion gestellt wird. Eine entsprechende Regel sieht daher typischerweise so aus:

```
(defrule MODUL::Regel-mit-trigger "Typ einer Regel, sensitiv hinsichtlich eines Fakts (trigger)“
   (fact_1) (fact_2) (fact_3)
   (trigger)
   =>
   (action) )
```

Am Ende eines Schlussfolgerungszyklus, in dem die Regel „Regel-mit-trigger“ gefeuert hat, wird in dem „*CLEAN_UP“*-Zustand der Fakt `(trigger)` entfernt. Für einen neuen Schlussfolgerungszyklus, ausgelöst durch ein *Timer*- oder Tastenereignis, wird dieser Fakt wieder neu in die Faktenbasis gestellt/eingespeist `(assert (trigger))` und führt dazu, dass die Regel „Regel-mit-trigger“ abermals aktiviert und bereit zum Feuern ist, falls auch `(fact_1) (fact_2) (fact_3)` wiederum zutreffend sind.

Diese Konstruktion mit einem Trigger-Fakt und dem „Löschen“ von im Laufe des Schlussfolgerungsprozesses generierten Fakten, die weiterhin nicht mehr benötigt werden, ermöglicht einen (kontrollierten) kontinuierlichen ereignisorientierten Ablauf der „*In-field Controller”*-Simulation. Das „Löschen“ bzw. Entfernen von Fakten aus der Faktenbasis im Zustand „*CLEAN_UP“* bzw. „*CLEAN_UP_DEMS“* erfolgt nach folgendem Schema, gezeigt für einen Fakt des „*CROP_PRODUCTION-SUM_UP“*-Moduls und codiert in JESS:

[81] A. a. O., S. 100

```
(defrule CLEAN_UP::CROP_PRODUCTION-SUM_UP-N_recommendation-cleanUp
   "CROP_PRODUCTION-SUM_UP::N_recommendation clean up"
   ?n_rec <- (CROP_PRODUCTION-SUM_UP::N_recommendation)
   =>
   (retract ?n_rec) )
```

Bei der Beschreibung des Schlussfolgerungsprozesses auf der Ebene des *Knowledge-base Levels* wurde bereits klar, dass Zwischenergebnisse in Form von Fakten erzeugt werden, die nur für diesen Inferenz-Prozess Gültigkeit besitzen. Werden diese Fakten vor einem neuen Situationsbewertungsdurchlauf nicht aus der Faktenbasis gelöscht, so würde das Ergebnis verfälscht. Aus diesem Grunde wurde der Zustand „*CLEAN_UP*“ implementiert, innerhalb dessen über Anwendung obig dargestellten Mechanismus die Faktenbasis von veralteten Zwischenergebnissen bereinigt wird. Herauszuheben ist, dass in diesem Zustand als letzte Aktion der Fakt „*trigger*“ ebenfalls aus der Faktenbasis gelöscht wird und anschließend der Fokus auf „*MAIN*“ gesetzt sowie mit dem `(halt)` Befehl noch ein Stopp des Inferenz-Prozesses kommandiert wird. Der „*CLEAN_UP_DEMS*“-Zustand ist mit „*CLEAN_UP*“ nahezu identisch. Jedoch ist er dem Anwendungsfall des kontinuierlichen Betriebs der speziellen Funktionalität des Notaus geschuldet, was nachfolgend erläutert wird. Folgende Regel sensiert die Betätigung des Notausschalters und aktiviert den „*EMERGENCY_STOP*“ Zustand:

```
(defrule EMERGENCY_STOP::Emergency_Stop "Emergency Stop"
   (declare (auto-focus TRUE))
   (emergency-stop-status (user_attribute ?u&: (eq ?u 1.0)))
   (trigger) ; necessary since the RHS action must be executed for each (run) cycle
   =>
   (bind ?application_setpoint 0)
   (call ?fertilizer setImplement_attribute (float ?application_setpoint))
   (?*javagui* setApplicationValue (float ?application_setpoint))
   (focus CLEAN_UP) )
```

Mittels der Autofokusfunktion ist die Regel bereit zum Feuern, sobald der Status des ON/OFF-Wechselschalters auf aktiv gestellt wurde. Die entsprechende Codezeile wird im nächsten Kapitel erläutert. Die zusätzliche Prüfung auf den „*trigger*“-Fakt ermöglicht, dass die Regel für jeden neuen Schlussfolgerungsprozess bereit zum Feuern ist, solange der Schalterstatus unverändert auf aktiv verharrt und der Fakt „*trigger*“ bei Anstoßen des neuen Zyklus wieder in die Faktenbasis eingestellt wird. Auf der Aktionsseite der Regel wird sozusagen die Ableitung der Sollwertempfehlung kurzgeschlossen und dem Stickstoffapplikationswert der Wert 0 zugewiesen. Mit der `(call)`-Funktion wird die Simulation der Prozessumgebung entsprechend aktualisiert (vgl. Kap. A.4.2) und abschließend werden die Anzeigefelder der GUI für den Sollwert aktualisiert. Schließlich erfolgt der Fokuswechsel zu „*CLEAN_UP*“. Nach dem Fokuswechsel zu „*MAIN*“ wird dort

auf den zeitbasierten neuen Start eines Bewertungszyklus gewartet, der die Eingangsinformationen einliest, den „*trigger*“-Fakt neu setzt und die Inferenz mit `(run)` neu anstößt. Abgesehen vom Beenden der Simulation oder Stopp des kontinuierlichen Betriebes läuft dieser beschriebene Ablauf endlos bis der ON/OFF Wechselschalter auf desaktiviert gestellt wird. Dieses Abschalten des Notaus wird durch die folgende Regel des „*EMERGENCY_STOP*“-Zustandes adressiert:

```
(defrule EMERGENCY_STOP::disable_Emergency_Stop "Disable Emergency Stop"
   (declare (auto-focus TRUE))
   (emergency-stop-status (user_attribute ?u&: (eq ?u -1.0)))
   =>
   (focus CLEAN_UP_DEMS) )
```

Wichtig bei dieser Regel ist die Tatsache, dass nicht auf das Vorliegen eines Fakts „*trigger*“ geprüft wird. Dies ist nicht nötig, da die Änderung des Faktes *emergency-stop-status* von aktiviert auf desaktiviert stattgefunden hat und durch die JESS zugrundeliegenden Technik die Regel bereits bereit zum Feuern geschalten hat. Der Aktionsteil der Regel setzt den Fokus auf den Zustand „*CLEAN_UP_DEMS“*. Dort wird wiederum für eine von Zwischenergebnissen bereinigte Faktenbasis gesorgt, der Fakt `(trigger)` wird jedoch nicht entfernt. Diese Faktenbereinigungsaktivität ist nötig, da zumindest theoretisch nicht davon ausgegangen werden kann, dass während des „*CLEAN_UP“*-Zustandes nach Aktivierung des Notaus dieser nicht sofort wieder desaktiviert wird. Zum Abschluss von „*CLEAN_UP_DEMS“* wird der Fokus an „*MAIN“* gegeben, jedoch mit dem `(run)`-Befehl versehen. Somit können die aktuell vorliegenden Eingangsinformationen für die Schlussfolgerung prozessiert werden und es muss nicht erst gewartet werden, bis *timer*-basiert ein neuer Zyklus angestoßen wird. Trotz des im Endergebnis nur geringfügigen Unterschieds, sollte damit gezeigt werden, wie nur der Schlussfolgerungsprozess pausiert wird, die Eingangsinformationen weiterhin eingelesen werden und nach Desaktivierung umgehend wieder automatisch eine Sollwertempfehlung abgeleitet werden kann.

Es folgt eine Bemerkung zu dem Ausgangszustand des Fakt *emergency-stop-status*, der den Notauswechselschalters repräsentiert. Initial ist er auf OFF (entspricht -1.0 für die *user_attribute* Variable*)* eingestellt, die Regel *EMERGENCY_STOP::disable_Emergency_Stop* feuert jedoch trotz ihrer Autofokusfunktionalität nach Inferenzstart nicht sofort. Dies ist in der gewählten Faktenimplementierung der Prozessumgebung (vgl. Kap. A.4.2) begründet, bei der Regeln, die dynamische Variablenwerte der Prozessumgebung in ihrem Bedingungsteil verarbeiten, erst aktiviert werden, falls eine Wertänderung der Variablen stattgefunden hat.

A.4.2 Faktenbasis und Simulation der Prozessumgebung

Für die Simulation der Datenfusionsprozesse des „*In-field Controller"* ist die Generierung und Repräsentation des kurzfristigen, mittelfristigen und langfristigen deklarativen Wissens von besonderer Bedeutung. Wie bereits mehrfach erläutert, wird dieses Wissen durch die Sensorwerte, die (Überlagerungs-)kartenwerte, Nutzereingaben und die Zwischenergebnisse der Datenfusionsprozesse bzw. Schlussfolgerungsprozesse erfüllt. Neben dem eigentlichen Fusionsprozess gilt es, auch die jeweiligen Werte zur Anzeige bzw. Eingabe über die GUI zugänglich zu machen. Die Kernkomponente der Simulation bzw. des Expertensystems in dem sich diese Wissensarten wiederfinden, ist die Faktenbasis.
Im technischen Sinne unterscheidet JESS drei verschiedene Kategorien von Fakten:

- *Unordered facts*
- *Ordered facts*
- *Shadow facts*

FRIEDMAN-HILL (2003) [Fri03] spricht von den *unordered facts* als den „Arbeitspferden" der Faktenbasis und vergleicht die zugrundeliegende Datenstruktur mit einer Reihe in einer Tabelle einer relationalen Datenbank mit individuell bezeichneten Datenfeldern, den sogenannten `slots`, die den Spalten der Tabelle entsprechen. (Wie bereits im Tool-Kapitel erwähnt, liegt jedoch eine objektorientierte Datenstruktur der Faktenbasis in JESS zugrunde). Beispielhaft für diese Faktenart könnte ein Fakt über eine Teilfläche N mit vier `slots` für zugehörigen Sensor- und Kartenwerten folgendermaßen aussehen:

(Teilfläche-N (REIP 720.0) (Ertrag 3.3) (EM38 345) (N-1-Rate 40))

Ein *ordered fact* hingegen verzichtet auf die Bezeichnung der einzelnen Felder, womit obiges Beispiel als *(Teilfläche-N 720.0 3.3 345 40)* formuliert würde.
Die für diese Arbeit wichtigste Faktenklasse sind die sogenannten *shadow facts*. *Shadow facts* sind grundsätzlich *unordered facts*, die direkt mit Java-Objekten verknüpft sind. Hierzu greift JESS auf die Technik der *JavaBeans* zurück. Mit *JavaBeans* wird die Technik von wieder verwendbaren (*reusable*) Softwarekomponenten in Java bezeichnet [Ull03].
Bevor jedoch dargestellt wird, wie mit *shadow facts* und *JavaBeans* die Prozessumgebung realisiert wurde, folgt zuerst die Beschreibung der Implementierung der Data Fusion-Prozesszwischenergebnisse (mittelfristiges Wissen) in Form von *unordered facts*.
Bevor ein *unordered fact* zum Einsatz kommen kann, ist es mittels der `deftemplate` -Funktion zu definieren. Dabei wird der dem Schlüsselwort folgende Bezeichner als Name für den Fakt genutzt und die einzelnen `slot`-Namen sind die Bezeichner für die einzelnen Werte

eines Fakts. In JESS sind die Typen von Variablen nicht zu definieren, sondern werden bei der Wertzuweisung automatisch erkannt. Beispielhaft wird das Prinzip für den nötigen Fakt des „*CROP_PRODUCTION“*-Wissensmoduls gezeigt. Mögliche Sollwertempfehlungen dieses Wissensmoduls sollten als Fakten mit dem Namen „*N_recommendation*“, dem jeweiligen numerischen Wert „*setpoint*“ und einem Erklärungstext „*explanation*“ neu in die Faktenbasis gestellt werden. Die zugehörige Definition lautet:

(deftemplate CROP_PRODUCTION::N_recommendation (slot setpoint) (slot explanation))

Ein entsprechendes Zwischenergebnis einer Sollwertempfehlung von 60 kg N/ha aufgrund der (Dünge)Regel <N60-2> wird somit folgendermaßen der Faktenbasis zugeführt:

```
(assert (CROP_PRODUCTION::N_recommendation
   (setpoint 60)
   (explanation "60 kg N/ha since rule < N60-2 >")))
```

Die Umsetzung der im *Knowledge-base Level* definierten Fakten bzw. Fusionszwischenergebnisse der einzelnen Wissensmodule ist in Tabelle A.1 zusammengestellt.

Tabelle A.1: Zusammenstellung der Fakten als Definition für *unordered facts*

Wissensmodul	Faktendefinition als unordered facts
MAIN	(deftemplate N_recommendation (slot setpoint) (slot explanation))
CROP_PRODUCTION	(deftemplate CROP_PRODUCTION::N_recommendation (slot setpoint) (slot explanation))
CROP_PRODUCTION -SUM_UP	(deftemplate CROP_PRODUCTION-SUM_UP::N_recommendation (slot setpoint) (slot explanation))
CONSTRAINTS	(deftemplate CONSTRAINTS::N_recommendation (slot setpoint) (slot explanation))
CONSTRAINTS	(deftemplate CONSTRAINTS::N_applied (slot amount) (slot explanation))
CONSTRAINTS	(deftemplate CONSTRAINTS::Adjustment_factor (slot factor) (slot explanation) (slot priority))
CONSTRAINTS-SUM_UP	(deftemplate CONSTRAINTS-SUM_UP::N_recommendation (slot setpoint) (slot explanation))
CONSTRAINTS-SUM_UP	(deftemplate CONSTRAINTS-SUM_UP::N_applied (slot amount) (slot explanation))
CONSTRAINTS-SUM_UP	(deftemplate CONSTRAINTS-SUM_UP::Adjustment_factor (slot factor) (slot explanation) (slot priority))
AG_ENGINEERING	(deftemplate AG_ENGINEERING::N_recommendation (slot setpoint) (slot explanation))
AG_ENGINEERING	(deftemplate AG_ENGINEERING::Control_Mode (slot mode) (slot explanation))
AG_ENGINEERING	(deftemplate AG_ENGINEERING::Quality_statement (slot quality) (slot explanation))
AG_ENGINEERING-SUM_UP	(deftemplate AG_ENGINEERING-SUM_UP::N_recommendation (slot setpoint) (slot explanation))
AG_ENGINEERING-SUM_UP	(deftemplate AG_ENGINEERING-SUM_UP::Control_Mode (slot mode) (slot explanation))
AG_ENGINEERING-SUM_UP	(deftemplate AG_ENGINEERING-SUM_UP::Quality_statement (slot quality) (slot explanation))
SUM_UP	(deftemplate SUM_UP::N_recommendation (slot setpoint) (slot explanation))

Bevor die Umsetzung der Ergebnisse des *Knowledge-base Level*s und die Abbildung der Prozessumgebung gezeigt wird, werden die Grundprinzipien der *shadow facts* und der JavaBeans-Integration in JESS dargelegt. Für eine tiefer gehende Betrachtung wird auf die Referenzliteratur [Fri03] und [Ull03] sowie auf die Online-Dokumentation von JESS und Java verwiesen.

(JavaBeans) Softwarekomponenten halten sich an die durch das (JavaBeans) Komponentenmodell vorgegebenen Namenskonventionen (Schnittstellendefinitionen). Dies stellt jedoch noch keinen Unterschied zu herkömmlichen Softwarebibliotheken dar. Den Unterschied macht erst die Einführung eines Interaktions-Modell zwischen den Komponenten als eine Säule dieser Softwaretechnik, die auf der Kommunikation über Ereignisse zwischen

JavaBeans basiert. ULLENBOOM (2003) [Ull03] fasst die wichtigsten Merkmale von JavaBeans und die dahinterstehenden Java-Techniken in folgender Weise zusammen:

- Selbstbeobachtung (*Introspection*): Eine Klasse lässt sich von außen analysieren; die Klasse selbst wiederum kann umgekehrt feststellen, ob sie von außen gerade ausgelesen, modelliert oder verwendet wird
- Eigenschaften (*Properties*): Attribute beschreiben den Zustand des Objektes; diese Eigenschaften können von außen durch Methoden abgefragt oder geändert werden
- Ereignisse (*Events*): Komponenten können Ereignisse auslösen, mit Hilfe derer sie andere Komponenten und Programme über eigene Zustandsänderungen informieren, Grundlage der Ereignisbehandlung ist das Modell von Java 1.1 mit Auslösern und Empfängern (, auch genannt Interessenten bzw. Zuhörer (*Listener*))
- Anpassung (*Customization*): Der Entwickler einer JavaBean kann die Eigenschaften visuell und interaktiv anpassen
- Speicherung (*Persistzenz*): Jede JavaBean kann ihren internen Zustand (, also die Eigenschaften) durch Serialisierung speichern und wiederherstellen

JavaBeans unterscheiden vier Arten von Eigenschaften, von denen hauptsächlich die gebundenen Eigenschaften (*bound properties*) für die *shadow facts* eine Rolle spielen [Ull03]. Bei den *bound properties* kann eine Komponente angemeldete Interessenten über Änderungen der Eigenschaft informieren. Die Wertabfrage- oder -änderung von Eigenschaften erfolgt über Methoden, die einer besonderen Namenskonvention folgen. Im Falle einfacher Attribute sind dies `setEIGENSCHAFT()` und `getEIGENSCHAFT()` bezeichnete Methoden, wobei `EIGENSCHAFT` stellvertretend für die jeweilige Eigenschaft/Attribut steht. Neben primitiven Datentypen können Eigenschaften jedoch auch komplexe Klassen sein.

Ein *shadow fact* ist vergleichbar einem *unordered fact*, dessen `slots` mit den Eigenschaften einer JavaBean korrespondieren. JavaBeans sind eine Art von normalen Java-Objekten, daher fungieren *shadow facts* wie eine Verbindung zwischen der Faktenbasis und einer Java-Applikation, die in JESS läuft bzw. die aus JESS herausläuft (vgl. die Möglichkeiten von JESS und Java zu verknüpfen). *Shadow facts* werden so genannt, da sie wie Schattenbilder oder Kopien von externen JavaBean-Objekten innerhalb von JESS erscheinen bzw. genutzt werden. Somit können im Prinzip reguläre Java-Objekte, vorausgesetzt sie erfüllen die Voraussetzungen der JavaBeans-Definition, als Fakten in die Faktenbasis aufgenommen und für Inferenz- und somit Fusionsprozesse genutzt werden. Im Grund basieren auch die *unordered* und *ordered facts* auf dieser objekt-orientierten Wissensrepräsentationsform und wurden auch JESS-intern derart implementiert [Fri03]. Zusammengefasst bedeutet dies, dass JESS mittels der Technik der *shadow facts* über das mächtige Potential verfügt, um flexibel die Funktionalität (Eigenschaften und Ereignisse) größerer Softwaresysteme und -komponenten innerhalb JESS für regelbasierte Aufgaben verfügbar zu machen. Um die

Faktenbasis mit *shadow facts* zu bestücken, ist anstelle der `deftemplate`-Funktion die Funktion `defclass` zu verwenden und anstatt der `assert` – Funktion wird über die JESS-Funktion `definstance` die Repräsentation einer spezifischen JavaBeans-Instanz als *shadow fact* generiert. Über Anwendung der Selbstbeobachtung *(Introspection)*-Technik erstellt JESS automatisch eine „Vorlage" (*template*) für eine JavaBeans-Klasse. Eine mit `deftemplate` definierte „Vorlage" für ein *shadow fact* besitzt für jede JavaBean-Eigenschaft einen `slot`.
Anhand eines simulierten Online-Sensors, der den Vegetationsindex REIP während der zweiten N-Gabe bestimmen soll, wird für das beschriebene Verfahren die Codierung in JESS und die minimale JavaBean-Klassendefinition mit erläuternden Kommentaren[82] gezeigt.

JESS-Code 1: Beispiel einer Definition einer JavaBean-Klasse für einen Online-Sensor-Pflanze und die Integration eines shadow facts, der ein Exemplar dieses Sensors nutzt.

```
(defclass reip2 PlantDynamic)
```
Es wird eine shadow fact-Vorlage basierend auf der JavaBeans Klasse `PlantDynamic.class` *erzeugt, um den Vegetationsindex REIP der zweiten N-Gabe als Fakt greifbar zu haben.*

```
(bind ?reip2 (new PlantDynamic))
```
In diesem Fall wird innerhalb JESS eine Instanz der JavaBeans-Klasse `PlantDynamic.class` *generiert. Als Objektname wurde* `?reip2` *gewählt, um die wesentliche (Objekt)Eigenschaft hervorzuheben.*

```
(definstance reip2 ?reip2)
```
Mit dieser Codezeile wird die (dynamische) shadow fact Instanz `reip2` *erzeugt, die direkt die JavaBeans-Instanz* `?reip2` *in der Faktenbasis abbildet.*

JavaBeans-Klasse:

```
import java.beans.*;
```
Es wird die Java-Klasse importiert, die die JavaBeans-Technik verfügbar macht bzw. integriert.

```
public class PlantDynamic {
```
Es startet die Klassendefinition für die JavaBeans-Klasse mit dem Namen `PlantDynamic`.

```
  private double plant_attribute = 0.0;
```
Es wird die wesentliche Eigenschaft als Variable `plant_attribute` *definiert, die einen Vegetationsindex in Form einer Zahl (Typ: Double) repräsentiert.*

```
  public double getPlant_attribute() {
    return plant_attribute;
  }
```
Dieser Code-Abschnitt definiert die „`get`*" Methode zum Abfragen des Zustands (=Wert) der Eigenschaft* `plant_attribute`.

```
  public void setPlant_attribute (double p) {

       double old = plant_attribute;
```

[82] Sowohl für dieses Code-Beispiel als auch folgende wird JESS- oder Java-Code linksbündig in einem Courier New-Zeichensatz aufgeführt. Rechtsbündig wird auf gleicher Höhe oder unterhalb der zugehörige Codeabschnitt kurz erläutert.

```
        plant_attribute = p;
        pcs.firePropertyChange("plant_attribute",new Double(old), new Double(p));
    }
```

Obiger Code-Abschnitt definiert die „set" -Methode zur Änderung des Zustands (=Wert) der Eigenschaft `plant_attribute`*. Zugleich wurde die für bound properties typische Fähigkeit zur Interaktion mit anderen Komponenten integriert. Dazu wird ein Methodenaufruf durchgeführt, der wiederum ein Ereignis ausgelöst (fire), das angemeldete Interessenten anderer Komponenten über die Änderung der entsprechenden Eigenschaft informiert.*

```
    private PropertyChangeSupport pcs = new
                  PropertyChangeSupport(this);

    public void addPropertyChangeListener(PropertyChangeListener pcl){
      pcs.addPropertyChangeListener(pcl);
    }

    public void removePropertyChangeListener(PropertyChangeListener pcl){
      pcs.removePropertyChangeListener(pcl);
    }
```

Obige Objekt- und Methodendefinitionen stellen die nötige Funktionalität bereit, um das Modell der Ereignisbehandlung für die vorliegende JavaBean-Implementierung zu unterstützen (support). Das Objekt und die Methoden werden genutzt (Anm.: In dieser Arbeit von der shadow facts Funktionalität in JESS), um über Zustandsänderungen von Eigenschaften informieren und die entsprechenden Interessenten (Listener) hierfür anmelden oder abmelden zu können.

```
}
```

Die Klassendefinition endet.

Mit dieser implementierten Funktionalität ist die *shadow fact* Instanz `reip2` stets über die Zustandsänderungen der Eigenschaft `plant_attribute` der JavaBeans-Instanz `?reip2` informiert und ändert stets und unmittelbar den Wert ihres eigenen `plant_attribute` – `slots` im Einklang damit. Im Gegenzug ändert auch die JavaBeans-Instanz `?reip2` automatisch ihren `plant_attribute`-Wert, wenn mit der JESS Funktion `modify` der `plant_attribute -slot` von `reip2` modifiziert wurde.

Ergänzend sei noch darauf hingewiesen, dass in dieser Arbeit nur die dynamische Definition von *shadow facts* genutzt wird. Dies bedeutet, dass die Attributwerte wie oben beschrieben stets automatisch aktualisiert werden. Zusätzlich bietet JESS aber auch eine statische Definition an, bei der Attributwertänderungen in der korrespondierenden JavaBean erst nach Anwendung der JESS-Funktion `reset` nachvollzogen werden. Einmal das `reset` Kommando am Anfang der Simulation durchzuführen, ist auch bei der dynamischen Definition von *shadow facts* aufgrund JESS-interner Strukturen nötig, um den Fakt `initial-fact` zu generieren.

Die beispielhafte JavaBeans-Klassendefinition für „PlantDynamic“ zeigt das Schema wie die Lese- und Schreibmethoden `getPlant_attribute` und `setPlant_attribute` verwendet werden können. Für diese Arbeit wurde eine Implementierungsform mit einem Hauptprogramm (Main() in JESS) in JESS gewählt, von dem aus auch die Prozessumgebungssimulation, die GUI und die Datenbankanbindung instanziert werden. Somit erfolgen Lese- und Schreibzugriffe auf die JavaBeans-Eigenschaften folgendermaßen, gezeigt am oben eingeführten Beispiel `?reip2`.

JESS-Code 2: Beispiel für Lese-und Schreibzugriffe auf JavaBeans-Eigenschaften

```
(call ?reip2 getPlant_attribute))
```

Mit dieser Code-Zeile wird die „`get`*“ -Methode von* `?reip2` *zum Abfragen des Zustands der Eigenschaft* `plant_attribute` aufgerufen *und das Ergebnis als Zahl vom Format double geliefert.*

```
(bind ?variable (call ?reip2 getPlant_attribute))
```

Demzufolge ist obige Code-Zeile die Zuweisung des Eigenschaftswert von `plant_attribute` *an die Variable* `?variable`.

```
(call ?reip2 setPlant_attribute 723.05)
(call ?reip2 setPlant_attribute ?variable)
```

Mit obigen beiden Code-Zeilen wird jeweils die „`set`*“ -Methode von* `?reip2` *zum Ändern des Zustands der Eigenschaft* `plant_attribute` aufgerufen *und zuerst auf den numerischen Wert von 723.05 geändert und dann auf den Wert der Variablen* `?variable` *gesetzt.*

In der realisierten Simulation wurden diese Art von Lese- und Schreibmethoden genutzt, um die Eigenschaftswerte der JavaBeans mit Werteinstellungen zu versehen. Je nach gewähltem Anwendungsfall (*use case*) sind die Quellen für diese Werte der Schreibzugriffe die Nutzereingaben über die GUI oder die JESS-Kommandozeile, die realen Feldtestdaten, die aus der angebundenen Datenbank ausgelesen werden, oder die implementierte Zufallszahlgenerierungsfunktion `(deffunction generate-random-inputs)`. Die Lesemöglichkeit wurde für die Dokumentationsfunktionalität und Testzwecke eingesetzt. Tabelle A.2 listet auf, wie die Prozessumgebung, abstrahiert auf ihre Eingangs- und Ausgangsgrößen für die MSDF, mit entsprechenden *shadow facts* nachgebildet wurde.

Tabelle A.2: Auflistung der implementierten *shadow facts* zur Repräsentation der Prozessumgebung

Simulierter Zustand einer Prozessumgebungsgröße auf einer Teilfläche	Shadow fact	JavaBean-Exemplar (zugrundeliegende Klasse; Eigenschaft)	[Einheit]; Datentyp;
REIP (2. N-Gabe) *Online-Sensorwert*	`reip2`	`?reip2 (PlantDynamic.class; plant_attribute)`	[nm]; Double
Ertrag (Jahr 1998) *Karteninformation*	`yield`	`?yield (SoilDynamic.class; soil_attribute)`	[Mg/ha]; Double
Zugkraft *Karteninformation*	`tforce`	`?tforce (SoilDynamic.class; soil_attribute)`	[kN]; Double
EM38-Messwert *Karteninformation*	`em38`	`?em38 (SoilDynamic.class; soil_attribute)`	[mS/m] (EC_{25}); Double
Begrenzungswert-Umweltschutzauflage *Karteninformation*	`env_limit`	`?env_limit (LimitDynamic.class; limit_attribute)`	[kg N/ha]; Double
Applikationsrate 1. N-Gabe *Karteninformation*	`n1`	`?n1 (ImplementDynamic.class; implement_attribute)`	[kg N/ha]; Double
Applikationsrate 2. N-Gabe *Sollwert für Aktorik*	`fertilizer`	`?fertilizer (ImplementDynamic.class; implement_attribute)`	[kg N/ha]; Double
Bereits ausgebrachte Applikationsrate bei 2. N-Gabe *Zwischenergebnis*	`n2_already_applied`	`?n2_already_applied (ImplementDynamic.class; implement_attribute)`	[kg N/ha]; Double
Prozessführungsmodus *Nutzereingabe*	`user-status`	`?user-status (UserDynamic.class; user_attribute)`	[ ] Double
Übersteuerungsfaktor für Applikationsrate 2. N-Gabe *Nutzereingabe*	`user-relative`	`?user-relative (UserDynamic.class; user_attribute)`	[ ] Double
Applikationsrate 2. N-Gabe Absolutwertvorgabe *Nutzereingabe*	`user-absolute`	`?user-absolute (UserDynamic.class; user_attribute)`	[kg N/ha]; Double
Status des Wechselschalters "ON/OFF" (Notaus) *Nutzereingabe*	`emergency-stop-status`	`?emergency-stop-status (UserDynamic.class; user_attribute)`	[ ] Double

Dabei wird in der linken Tabellenspalte der simulierte Zustand einer Prozessumgebungsgröße auf einer Teilfläche aufgeführt. Prozessumgebungsgrößen sind Online-Sensorwerte, Karteninformationswerte, Nutzereingaben, (Fusions-) Zwischenergebnisse und Sollwerte für die Aktorik, den Düngerstreuer. In der Spalte „Shadow fact" wird die entsprechende *shadow fact*-Instanz aufgeführt. Die nächste Spalte gibt die zugeordnete JavaBean-Instanz an. Zugleich ist die Klasse angegeben, auf der die Instanz basiert, und die genutzte JavaBean-

Eigenschaft aufgeführt, die im Grunde den Zustand der Prozessumgebungsgröße hält. In der letzten Spalte wird das Datenformat für den Eigenschaftswert sowie die entsprechende Einheit angegeben, in der der Wert zu interpretieren ist. Bis auf zwei Einheits- und Datenformatfestlegungen für Nutzereingaben ist dies geradlinig und unauffällig. Die binären Nutzereingaben für den gewählten Prozessführungsmodus und den Status des Wechselschalters „ON/OFF" (Notaus) wurden als Fließkommazahl (Typ: *Double*) realisiert. Die Umsetzung auf die binären Werte erfolgt bzw. wird berücksichtigt in den nutzenden Programmteilen, wie der GUI, dem Regelwerk oder der Dokumentation. Bei der (bidirektionalen) Konvertierung sind die Textstrings „Auto" oder „OFF" mit der Zahl „1.0" gleichbedeutend und die Textstrings „Manuell" und „OFF" mit der Zahl „-1.0". Auch für die Eingangsinformation über die Verfügbarkeit des Online-Sensors zur REIP-Bestimmung würde sich eine Realisierung über *shadow facts* anbieten, jedoch wurde der Mechanismus bereits zu einem frühen Zeitpunkt der Entwicklung über *unordered fact* funktionsfähig implementiert und abschließend nicht mehr geändert. Somit wurde diese Eingangsinformation als simple Fakten `(yesREIP)` für die Verfügbarkeit und `(noREIP)` für das Fehlen des entsprechenden Online-Sensors realisiert.

A.4.3 Regelwerk

Die Ausführungen des vorigen Kapitels haben die Umsetzung des *deklarativen* Teils des Wissens in Form der Ausgestaltung der Faktenbasis mit ihren *unordered facts* und *shadow facts* offengelegt. In diesem Kapitel gilt es die Implementierung des *langfristigen expliziten prozeduralen* Wissens, dem Regelwerk, zu beschreiben. Im Grunde wurde dieser Wissensanteil im *Knowledge-base Level* (vgl. Kap. 5.2.2.4) schon in natürlicher Sprache und „Pseudo-Code" zusammengestellt. An ausgewählten Regelbeispielen werden grundsätzliche und in der Realisierung wiederkehrende Codierungsschemata aufgezeigt. Die einzelnen Wissensmodule „*CROP_PRODUCTION"*, „*CROP_PRODUCTION-SUM_UP"*, „*CONSTRAINTS"*, „*CONSTRAINTS-SUM_UP"*, „*AG_ENGINEERING"*, „*AG_ENGINEERING-SUM_UP"*, „*SUM_UP"*, „*EMERGENCY_STOP"*, „*CLEAN_UP"* und „*CLEAN_UP_DEMS"* wurden stets als eigenständiges Regelmodul definiert. Dies lautet in JESS-Codierung:

```
(defmodule NAME_des_WISSENSMODULS)
```

In obiger Aufzählung fehlt das Modul „*MAIN"*. Dieses Modul ist in JESS das Standard-Modul, das auch existiert, selbst wenn es nicht speziell definiert wurde. In der vorliegenden Arbeit hat dieses Modul seine feste Position in der Ablaufsteuerung und verfügt nur über eine

Regel zum gezielten Fokuswechsel, dessen Implementierungsschema bereits in Kapitel A.4.1 festgehalten wurde.

Zentraler Punkt der Implementierung des Regelwerkes ist die Umsetzung der beiden Entscheidungsbäume des „*CROP_PRODUCTION“*-Modules, wie sie in Kapitel 5.2.2.4 formuliert und erörtert wurden. Anhand des Astes (Abb. A.7) des Entscheidungsbaumes mit dem Attribut des Online-Sensors REIP, der zur Sollwertempfehlung für die zweite N-Gabe von 30 N kg/ha führt, wird das Codierungsschema in JESS vorgestellt. Dieser Ast des Entscheidungsbaumes lässt sich mit einer einzigen Regel umsetzen.

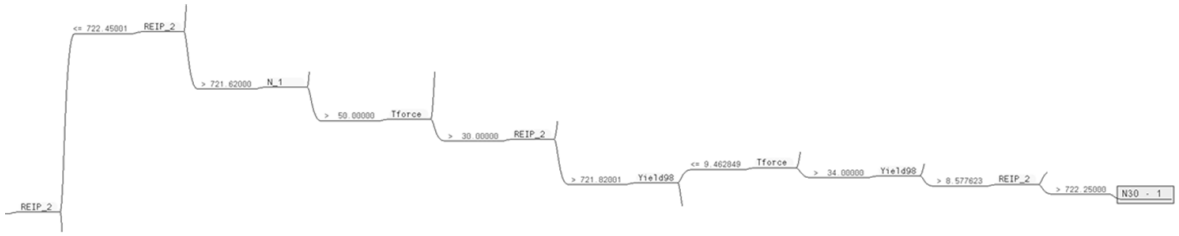


Abbildung A.7: Ast des Entscheidungsbaums (mit REIP_2) zur zweiten N-Applikation (D4-03) für die Regel N30-1

Da der Ast die Attribute *REIP_2, Yield98, Tforce* und *N_1* beinhaltet, muss auch die entsprechende Regel diese Attribute aufweisen. Über die entsprechende *shadow fact* Realisierung stehen diese Attribute als Fakten mit ihrem jeweiligen Zustand auch stets bereit. Was noch fehlt ist, die Formulierung der numerischen Größenvergleiche jedes einzelnen Entscheidungsschrittes in einer Form, die einen Mustervergleich im Bedingungsteil der Regel ermöglicht und die Zusammenhänge zwischen den Fakten wiedergibt. Dieses „Stilmittel“ ermöglicht JESS durch die *conditional elements*, sogenannten *pattern modifiers* [Fri03]. Die dementsprechend genutzte Funktion ist `test`, die zu einem Match führt, falls ein Funktionsaufruf nicht mit dem Ergebnis *FALSE* ausgewertet wird. Demgemäß lässt sich die Bedingung, ob das *REIP_2* Attribut größer als 722.25000 (nm) ist, für den Bedingungsteil (LHS) einer Regel so formulieren:

```
(reip2 (plant_attribute ?p1))
(test (> ?p1 722.25000))
```

Übersetzt bedeutet dies, dass in der Faktenbasis eine *shadow fact* Instanz von `reip2` mit der Eigenschaft `plant_attribute` existiert, die im Falle dieses positiven *match*es (positiv in Form der Existenz) der Variablen `?p1` zugewiesen wird. Das eingesetzte Stilmittel dieser Zeile wird als *variable constraints* bezeichnet. In der nächsten Zeile wird die *conditional*

element Funktion auf die Variable und den Größenvergleich angewendet. Ist der Wert von `?p1` tatsächlich größer als 722.25000, so ergibt diese Zeile das Ergebnis TRUE. Damit liegt mit den gegebenen Fakten für beide Bedingungen, die im Prinzip (implizit) über ein logisches UND verknüpft sind, ein erfolgreicher Mustervergleich/Match vor und die Regel wäre aktiviert bzw. bereit zum Feuern. Obere beide Codezeilen lassen sich auch etwas kürzer formulieren, jedoch mit identischer Funktionsweise:

```
(reip2 (plant_attribute ?p1&: (> ?p1 722.25000)))
```

Basierend auf diesem Konstrukt lässt sich der Entscheidungsbaum-Teilast für *N30-1* in die Entscheidungsregel *CROP_PRODUCTION::N30-1* umsetzen:

JESS-Code 3: Entscheidungsregel CROP_PRODUCTION::N30-1

```
(defrule CROP_PRODUCTION::N30-1 "Recommendation N30 Rule 1"
```

Beginn der Regeldefinition CROP_PRODUCTION::N30-1 mit einer Kurzbeschreibung (Textfragment zwischen Anführungszeichen).

```
(reip2 (plant_attribute ?p1&: (> ?p1 722.25000)))
(yield (soil_attribute ?s1&: (> ?s1 8.577623)))
(tforce (soil_attribute ?s2&: (> ?s2 34.00000)))
(yield (soil_attribute ?s3&: (<= ?s3 9.462849)))
(reip2 (plant_attribute ?p2&: (> ?p2 721.82001)))
(tforce (soil_attribute ?s4&: (> ?s4 30.00000)))
(n1 (implement_attribute ?i&: (> ?i 50.00000)))
(reip2 (plant_attribute ?p3&: (> ?p3 721.62000)))
(reip2 (plant_attribute ?p4&: (<= ?p4 722.45001)))
```

Bedingungsteil der Regel (LHS):

Ausgehend von dem Ende des Teilastes wird jeder einzelne Entscheidungsschritt mit der conditional element Funktion und den shadow fact Instanzen formuliert.

```
(yesREIP)
(trigger)
```

Die Prüfung auf Existenz des Faktes (yesREIP) wirkt wie ein Schalter zur Auswahl, welches Regelwerk zur Anwendung kommen soll. Entweder kommt das Set, das den Online Sensorwert berücksichtigt oder die Version für das Fehlen der REIP-Onlinesensorik, zum Einsatz.
Die Funktion der Prüfung auf den Fakt (trigger) wurde bereits im Kapitel zur Ablaufsteuerung erläutert und zeigt an dieser Stelle die konkrete Umsetzung.

```
=>

(assert (CROP_PRODUCTION::N_recommendation
        (setpoint 30)
        (explanation "30 kg/ha since rule < N30-1 >")))
```

Aktionsteil der Regel (RHS):

Der Aktionsteil der Regel führt im Falle des Feuerns dazu, dass als Zwischenergebnis ein neuer Fakt (unordered fact) in die Faktenbasis eingestellt wird vom Typ CROP_PRODUCTION::N_recommendation, dessen setpoint-`slot` der Wert 30 und dessen explanation-`slot` das Textkürzel "30 kg/ha since rule < N30-1 >" zugewiesen wird.

```
)
```

Die Regeldefinition endet.

Aus Gründen der Code-Lesbarkeit und der sehr einfachen und zielgerichteten Transformation eines Astes in eine Regel wurde diese Form gewählt. Wenngleich das mehrmalige Auftreten

einer *shadow fact* Instanz (eleganter) mit nur einer „Existenzprüfung“ und weiteren `test`-Funktionen auf die damit nur einmalig vorhandene Variable ausgeführt werden könnte.
In der gezeigten Form wurden alle 34 „Teiläste“ des Regelsets „Entscheidungsbaum zur zweiten N-Applikation mit dem Attribut REIP_2, Versuchsfeld D4 der TU München, Versuchsjahr 2003“ in 34 Regeln umgesetzt. Das alternative Regelset „Entscheidungsbaum zur zweiten N-Applikation ohne das Attribut REIP_2, Versuchsfeld D4 der TU München, Versuchsjahr 2003“ wurde in gleicher Weise in 23 Regeln transformiert. Der einzige Unterschied bestand nur darin, dass anstatt des Mustervergleichs auf *(yesREIP)* der *Matching*-Prozess auf *(noREIP)* durchgeführt wird, was eben das Fehlen der Online-Sensorik für die REIP-Bestimmung repräsentiert.
Ebenfalls im *CROP_PRODUCTION*-Regelmodul wird mit einer weiteren *conditional element* Funktion ein wichtiges Grundkonstrukt umgesetzt, das Prüfen auf Fehlen eines bzw. mehrerer Fakten mit der entsprechenden Reaktion darauf. Mit der `not`-Funktion lässt sich eine Faktenbasis hinsichtlich dieser Zielsetzung untersuchen. Der Wissensbaustein, der im Falle, dass keine Sollwertempfehlungen aufgrund der Entscheidungsregeln gefunden wurden, die Null-Applikation 0 kg N/ha als Default-Wert festgelegt, wird in JESS in folgender Weise codiert:

JESS-Code 4: Prüfen auf die Nichtexistenz eines Fakts in der Faktenbasis

```
(defrule CROP_PRODUCTION::default "default"

    (not (CROP_PRODUCTION::N_recommendation (setpoint ?s)))
    (trigger)

    =>

    (assert (CROP_PRODUCTION::N_recommendation
            (setpoint 0)
            (explanation "0 kg/ha since rule < default >")))

 )
```

Im *Knowledge-base Level*-Kapitel wurde wiederholt prozedurales Wissen zur Sammlung von Fakten und der Weiterverarbeitung und Zusammenfassung mittels Größenvergleich, Priorisierens und Aneinanderreihung von Erklärungstexten eingeführt und beschrieben. Exemplarisch für diese Wissensteile sollen entsprechende Codierungen anhand von Beispielen der Umsetzung des *CONSTRAINTS-SUM_UP*-Regelmoduls dargestellt werden. Das in „Pseudo-Code“ beschriebene Wissen zur Bestimmung der Sollwertlimitierung *„CONSTRAINTS-SUM_UP::N_recommendation“* wurde in JESS mit den drei folgenden Regeln codiert:

JESS-Code 5: Regel zur Sammlung und Formatwandlung aller Sollwertlimitierungen des Moduls „CONSTRAINTS“

```
(defrule CONSTRAINTS-SUM_UP::constraints_recommendation_all
        "All N-setpoint limitations according to constraints assessment"
```

Beginn der Definition einer Regel, die alle vorliegenden Sollwertlimitierungs-Zwischenergebnisse (des Typs CONSTRAINTS::N_recommendation) sammelt und daraus entsprechende Fakten des Moduls CONSTRAINTS-SUM_UP erzeugt.

```
  ?constr <- (CONSTRAINTS::N_recommendation (setpoint ?s1) (explanation ?e1))
```

Bedingungsteil der Regel (LHS):

Es existiert eine Limitierung des Typs CONSTRAINTS::N_recommendation (setpoint, explanation). Ihre `slot`*-Werte setpoint und explanation werden den Variablen* `?s1` *und* `?e1` *zugewiesen. Zugleich wird über das Mittel des „Pattern bindings“*[Fri03] *der pattern binding Variable* `?constr` *dieser ursprüngliche Fakt zugewiesen, so dass auf ihn im Aktionsteil der Regel die JESS-Funktionen (wie)* `retract`, `modify` *oder* `duplicate` *angewendet werden können.*

```
  =>

     (retract ?constr)
```

Aktionsteil der Regel (RHS):

Diese Funktion des Aktionsteil der Regel führt im Falle des Feuerns dazu, dass der ausgewählte Fakt des Bedingungsteil der Regel (LHS) aus der Faktenbasis entfernt wird.

```
     (assert (CONSTRAINTS-SUM_UP::N_recommendation
             (setpoint ?s1)
             (explanation ?e1)))
```

Die Werte der Variablen ?s1 und ?e1 des gelöschten Fakts werden als Werte des neu generierten Fakts (CONSTRAINTS-SUM_UP::N_recommendation für dessen setpoint-`slot` *und explanation-*`slot` *verwendet.*

```
  )
```

Die Regeldefinition endet.

Die nächste Regel bestimmt aus der Menge aller durch obige Regel erzeugten *CONSTRAINTS-SUM_UP::N_recommendation* –Fakten, denjenigen mit dem minimalen *setpoint*-Wert.

JESS-Code 6: Regel zur Bestimmung des CONSTRAINTS-SUM_UP-Fakts mit dem niedrigsten setpoint-Wert

```
(defrule CONSTRAINTS-SUM_UP::constraints_recommendation_lowest "The N-setpoint
           limitation with the lowest value wins"
```

Beginn der Regeldefinition.

```
  ?n_rec_lower <- (CONSTRAINTS-SUM_UP::N_recommendation
                                      (setpoint ?s1) (explanation ?e1))

  ?n_rec_higher <- (CONSTRAINTS-SUM_UP::N_recommendation
                                      (setpoint ?s2) (explanation ?e2))
```

```
(test (< ?s1 ?s2))
```

Bedingungsteil der Regel (LHS):

Es existieren zwei Fakten CONSTRAINTS-SUM_UP::N_recommendation. Ihre `slot`*-Werte setpoint und explanation werden jeweils den Variablen* `?s1`*,* `?e1` *sowie* `?s2` *und* `?e2` *zugewiesen. Wiederum werden die beiden Fakten den beiden pattern binding Variablen* `?n_rec_lower und ?n_rec_higher` *zugewiesen. Die Codezeile mit der conditional element Funktion* `test` *resultiert in TRUE, was einem positiven Mustervergleich (Match) entspricht, falls* `?s2` *größer als* `?s1` *ist. Damit erklärt sich auch die gewählte Namensgebung für die beiden pattern binding Variablen*

```
=>

  (retract ?n_rec_higher)
```

Aktionsteil der Regel (RHS):

Feuert diese Regel, wird der `?n_rec_higher` *zugeordnete Fakt des Bedingungsteil der Regel (LHS) aus der Faktenbasis entfernt wird.*

```
 (assert (CONSTRAINTS-SUM_UP::N_recommendation
         (setpoint ?s1)
         (explanation ?e1)))
```

Ein Einstellen eines Fakts (CONSTRAINTS-SUM_UP::N_recommendation mit den `slot` *-Werten* `?s1`*,* `?e1` *wirkt wie ein Überschreiben des Fakts der* `?n_rec_lower`*. (Damit wird sichergestellt, dass dieser Fakt stets für neue Mustervergleichsprozesse aktiv gehalten wird.)*

```
)
```

Die Regeldefinition endet.

Die oben definierte Regel ist in der Lage aus einer Menge von *CONSTRAINTS-SUM_UP::N_recommendation*-Fakten, den mit dem kleinsten *setpoint*-Wert zu ermitteln und dabei alle anderen aus der Faktenbasis zu entfernen. Mit der abschließenden dritten Regel wird der noch offene Punkt adressiert, dass entsprechende Fakten mit gleich großen *setpoint*-Werten vorliegen, aber einen anderen *explanation*-Text („String“) besitzen. Der Fall, dass auch gleiche *explanation*-Texte vorliegen, spielt keine Rolle, da die gewählte Fakten-Implementierung dies nicht als zwei unterschiedliche Fakten führt. Die Implementierung in JESS lautet:

JESS-Code 7: Regel für das Vorgehen bei Vorliegen von CONSTRAINTS-SUM_UP-Fakten mit dem gleichen setpoint-Wert

```
(defrule CONSTRAINTS-SUM_UP::constraints_recommendation_same "N-setpoint
           limitations with same values result in explanation concatenation"
```

Beginn der Regeldefinition.

```
  ?n_rec_same1 <- (CONSTRAINTS-SUM_UP::N_recommendation
                                    (setpoint ?s1) (explanation ?e1))
  ?n_rec_same2 <- (CONSTRAINTS-SUM_UP::N_recommendation
                                    (setpoint ?s2) (explanation ?e2))
```

```
(test (eq ?s1 ?s2))
(test (neq ?e1 ?e2))
```

Bedingungsteil der Regel (LHS):

Es existieren zwei Fakten CONSTRAINTS-SUM_UP::N_recommendation. Ihre `slot`*-Werte setpoint und explanation werden jeweils den Variablen* `?s1`*,* `?e1` *sowie* `?s2` *und* `?e2` *zugewiesen. Beide Fakten werden zugleich den pattern binding Variablen* `?n_rec_lower und ?n_rec_higher` *zugewiesen. Die* `test`*-Funktionen, ob* `?s1` *und* `?s2` *gleich und* `?e1` *und* `?e2` *ungleich sind, resultieren beide in einem TRUE-Wert.*

```
=>
 (retract ?n_rec_same1)
 (retract ?n_rec_same2)
```

Aktionsteil der Regel (RHS):

Feuert diese Regel, werden beide den Variablen `?n_rec_same1 und ?n_rec_same2` *zugeordnete Fakten des Bedingungsteil der Regel (LHS) aus der Faktenbasis entfernt.*

```
 (assert (CONSTRAINTS-SUM_UP::N_recommendation
         (setpoint ?s1)
         (explanation (str-cat ?e1 " / " ?e2))))
```

Es wird ein neuer Fakt CONSTRAINTS-SUM_UP::N_recommendation generiert, dessen setpoint-`slot` *mit dem gleichgroßen* `?s1` *Wert belegt wird und dessen explanation* `-slot` *durch eine Aneinanderreihung der beiden Texte von* `?e1` *und* `?e2` *sowie zweier Leerzeichen mit einem Schrägstrich gebildet wird. Die JESS-Funktion* `str-cat` *ermöglicht diese Aneinanderreihung (string concatenation).*

```
)
```

Die Regeldefinition endet.

Das gezeigte Prinzip einer regelbasierten Suche nach dem Minimum lässt sich natürlich auch auf die Suche nach einem Maximum übertragen, indem jeweils von den beiden Fakten, derjenige mit dem kleineren *setpoint*-Wert entfernt (`retract`) wird. Von diesem Implementierungsschema wird beispielsweise auch in dem Regelmodul „*CROP_PRODUCTION-SUM_UP“* Gebrauch gemacht. Auch im „*CONSTRAINTS-SUM_UP“*-Modul findet es in dem prozeduralem Wissensabschnitt Anwendung, bei dem nach dem Übersteuerungswunsch mit der höchsten Priorität gesucht wird. In diesem Wissensbereich wird die Ermittlung eines Maximums bzw. Minimums einer Größe um die Ermittlung einer Optimierung bezüglich zweier Größen, der höchsten Priorität *(priority)* und niedrigstem Übersteuerungswunsch *(factor)* bei gleich höchster Priorität, erweitert. Die bereits in „Pseudo-Code“ (vgl. Kap. 5.2.2.4) vorgestellte Regel wurde in JESS folgendermaßen formuliert:

JESS-Code 8: Regel zur Auswahl des CONSTRAINTS-SUM_UP-Fakts mit dem niedrigsten Übersteuerungsfaktor bei Vorliegen von Fakten mit gleich hoher Priorität

```
(defrule CONSTRAINTS-SUM_UP::adjustment_factor_same_priority_lower_factor
            "In case of same priority the lowest adjustment factor value wins"
```

Beginn der Regeldefinition.

```
?adf_samePriority_lowerFactor <- (CONSTRAINTS-SUM_UP::Adjustment_factor
                                    (factor ?f1) (explanation ?e1) (priority ?p1))
?adf_samePriority_higherFactor <- (CONSTRAINTS-SUM_UP::Adjustment_factor
                                    (factor ?f2) (explanation ?e2) (priority ?p2))
(test (eq ?p1 ?p2))
(test (neq ?e1 ?e2))
(test (< ?f1 ?f2))
```

Bedingungsteil der Regel (LHS):

Es existieren zwei Fakten CONSTRAINTS-SUM_UP::N_recommendation. Ihre `slot`*-Werte factor, explanation und priority werden jeweils den Variablen* `?f1`*,* `?e1`*,* `?p1` *sowie* `?f2`*,* `?e2` *und* `?p2` *zugewiesen. Die beiden Fakten werden den beiden pattern binding Variablen* `?adf_samePriority_lowerFactor` *und* `?adf_samePriority_higherFactor` *zugewiesen. Die beiden ersten* `test`*-Funktionen resultieren in einem TRUE-Wert, wenn die Prioritäten* `?p1` *und* `?p2` *gleich und die Erklärungstexte* `?e1` *und* `?e2` *ungleich sind. Die folgende* `test`*-Funktion resultiert in TRUE, was einem positiven Mustervergleich (Match) entspricht, falls* `?f2` *größer als* `?f1` *ist. Damit erklärt sich auch die gewählte Namensgebung für die beiden pattern binding Variablen.*

```
  =>
   (retract ?adf_samePriority_higherFactor)
```

Aktionsteil der Regel (RHS):

Diese Funktion des Aktionsteil der Regel führt im Falle des Feuerns dazu, dass der Fakt des Bedingungsteil der Regel (LHS) mit dem höheren Übersteuerungsfaktor aus der Faktenbasis entfernt wird.

```
   (assert (CONSTRAINTS-SUM_UP::Adjustment_factor
                              (factor ?f1)
                              (explanation ?e1)
                              (priority ?p1)))
```

Das Bestücken der Faktenbasis mit dem Fakt (CONSTRAINTS-SUM_UP::Adjustment_factor mit den `slot` *-Werten* `?f1`*,* `?e1` *und* `?p1` *wirkt wie ein Überschreiben des Fakts, der* `?adf_samePriority_lowerFactor` *zugeordnet ist. Damit wird sichergestellt, dass dieser Fakt stets für neue Mustervergleichsprozesse aktiv gehalten wird.*

```
)
```

Die Regeldefinition endet.

Hinsichtlich der Definition der Prioritätswerte mit *very_high, high, neutral, low* und *very low* wurde ein Hilfsmittel angewandt, das aus obiger Regeldefinition nicht eindeutig ersichtlich ist. Um die bisher vorgestellten Konstruktionen auch für die getroffene Wertezuordnung wiederverwenden zu können, wurden die Prioritätswerte mit Zahlen hinterlegt. Fälle, in denen

Fakten mit gleicher Priorität und gleichem Übersteuerungsfaktor vorliegen, werden mit Aneinanderreihung der Erklärungstexte gelöst. Die entsprechende Codierung folgt dem Schema der bereits beschriebenen Regelimplementierung für *CONSTRAINTS-SUM_UP::constraints_recommendation_same*.

Nachdem wichtige Grundprinzipien und JESS-Funktionen zur Implementierung des Regelwerks dargelegt wurden, schließt dieses Kapitel mit einem für das „*SUM_UP*“-Modul typischen Implementierungsbeispiel dessen Wissen bereits im Kapitel zur Konzeption und Formalisierung schriftlich festgehalten wurde. Dabei sollte die Situation, dass das Produkt aus *CROP_PRODUCTION-SUM_UP::N_recommendation (setpoint)* und *CONSTRAINTS-SUM_UP::Adjustment_factor (factor)* kleiner oder gleich dem Wert der Variable *setpoint* von *CONSTRAINTS-SUM_UP::N_recommendation* ist, dazu führen, dass ein neuer Fakt *SUM_UP::N_recommendation* generiert wird, dessen *setpoint*-`slot` den Ergebniswert der Multiplikation aus der Bedingung annimmt. Dem *explanation*-`slot` werden alle Erklärungstexte der an der Entscheidung beteiligten Fakten zugewiesen.

JESS-Code 9: Eine Regel des SUM_UP-Moduls für den Automatikmodus

```
(defrule SUM_UP::control_mode_automatically_1
    "Control mode automatically and limit is higher than adjusted N-recommendation"
```

Beginn der Regeldefinition.

```
   ?n_rec <- (CROP_PRODUCTION-SUM_UP::N_recommendation
                                  (setpoint ?s1) (explanation ?e1))
   ?ctrl_m <- (AG_ENGINEERING-SUM_UP::Control_Mode
                          (mode ?m1&:(eq ?m1 "automatically")) (explanation ?e2))
   ?n_constr <- (CONSTRAINTS-SUM_UP::N_recommendation
                                    (setpoint ?s2) (explanation ?e3))
   ?n_adf <- (CONSTRAINTS-SUM_UP::Adjustment_factor
                                (factor ?f1) (explanation ?e4))
   (test (<= (* ?s1 ?f1) ?s2))
```

<u>Bedingungsteil der Regel (LHS):</u>

Es existieren aus den vorhergehenden SUM_UP-Regelmodulen jeweils ein Fakt CROP_PRODUCTION-SUM_UP::N_recommendation, AG_ENGINEERING-SUM_UP::Control_Mode, CONSTRAINTS-SUM_UP::N_recommendation und CONSTRAINTS-SUM_UP::Adjustment_factor . Ihre `slot`*-Werte werden entsprechenden Variablen zugewiesen. Alle vier Fakten werden über die Zuweisung zu pattern binding Variablen im Aktionsteil der Regel zugänglich gemacht. Die* `test`*-Funktion resultiert in einem TRUE-Wert, wenn der Sollwert* `?s1` *multipliziert mit dem Übersteuerungsfaktor* `?f1` *größer oder gleich dem limitierenden Sollwert* `?s2` *ist.*

```
   =>
   (retract ?n_rec)
   (retract ?ctrl_m)
   (retract ?n_constr)
```

```
(retract ?n_adf)
(bind ?N_setpoint (* ?s1 ?f1))
```

Aktionsteil der Regel (RHS):

Feuert diese Regel, werden alle vier Fakten des Bedingungsteils der Regel (LHS) aus der Faktenbasis entfernt. Zugleich wird einer Variable `?N_setpoint` *das Produkt aus Sollwert* `?s1` *und dem Übersteuerungsfaktor* `?f1` *zugewiesen.*

```
(assert (SUM_UP::N_recommendation
            (setpoint ?N_setpoint)
            (explanation (str-cat " > " ?e1 ?*crlf* " > " ?e3 ?*crlf*
                                  " > " ?e2 ?*crlf* " > " ?e4 ?*crlf* ))))
```

Es wird ein neuer Fakt SUM_UP::N_recommendation generiert, dessen setpoint-`slot` *mit dem Wert von* `?N_setpoint` *belegt wird und dessen explanation -* `slot` *durch eine Aneinanderführung der (Erklärungs-) Texte von* `?e1`, `?e2`, `?e3` *und* `?e4` *sowie jeweils einem einleitenden Textfragment „ > “ und abschließenden Zeilenumbruch gebildet wird.*

```
)
```

Die Regeldefinition endet.

Bei dieser Regel fällt noch auf, dass der `slot`-Wert von *AG_ENGINEERING-SUM_UP::Control_Mode* mittels einer *predicate function* [Fri03] auf eine Übereinstimmung mit dem Automatik-Modus „*automatically*“ eingeschränkt wird. Auch eine weitere Funktionalität wurde noch nicht (explizit) angesprochen. Mustervergleiche von *unordered facts* auf der Bedingungsseite der Regel müssen nicht alle im `deftemplate` definierten `slots` enthalten, falls deren Wert keine einschränkende (*constraining*) Wirkung hat, d.h. für den Match nicht von Interesse ist. Dies wurde bei *CONSTRAINTS-SUM_UP::Adjustment_factor* angewandt, wo der *priority* –`slot` bei dieser Regel keine Rolle spielt.

Das Schema zur Entfernung von nicht mehr benötigten Fakten (Zwischenergebnissen) in den Modulen „*CLEAN_UP“* und „*CLEAN_UP_DEMS“* wurde bereits in dem Kapitel A.4.1 dargelegt. Somit bleibt zum Abschluss dieses Kapitels nur noch zu erwähnen, dass das implementierte Regelwerk 136 einzelne Regeln umfasst.

A.4.4 Datenbankanbindung

Zielsetzung der Datenbankanbindung war es einerseits, das Grundprinzip einer Datenbank-Integration in den „*In-field Controller*“ zu zeigen. Anderseits sollten die teilflächenspezifischen Messwerte für das TU München-Versuchsfeld D4 zur Applikation der zweiten N-Gabe bei Winterweizen für das Versuchsjahr 2003 für die Simulation zugänglich gemacht werden. Weiterhin sollte den Testaktivitäten eine Diagnosefunktionalität zur Seite gestellt werden.

Wie bereits im Kapitel 5.2.1 eingeführt, liegen die Versuchsdaten in einer rasterbasierten Form vor, was in einer relationalen Datenbank darstellbar ist. Aufbauend auf der definierten Systemarchitektur ist diese Datenbank lokal auf dem „*In-field Controller*“ angesiedelt. Abbildung A.8 zeigt das zugrundeliegende relationale Datenmodell in Form von flachen Tabellen. Durch die Fokussierung auf nur ein Testfeld mit einer Versuchsapplikation wurde das Datenmodell sehr einfach gehalten.

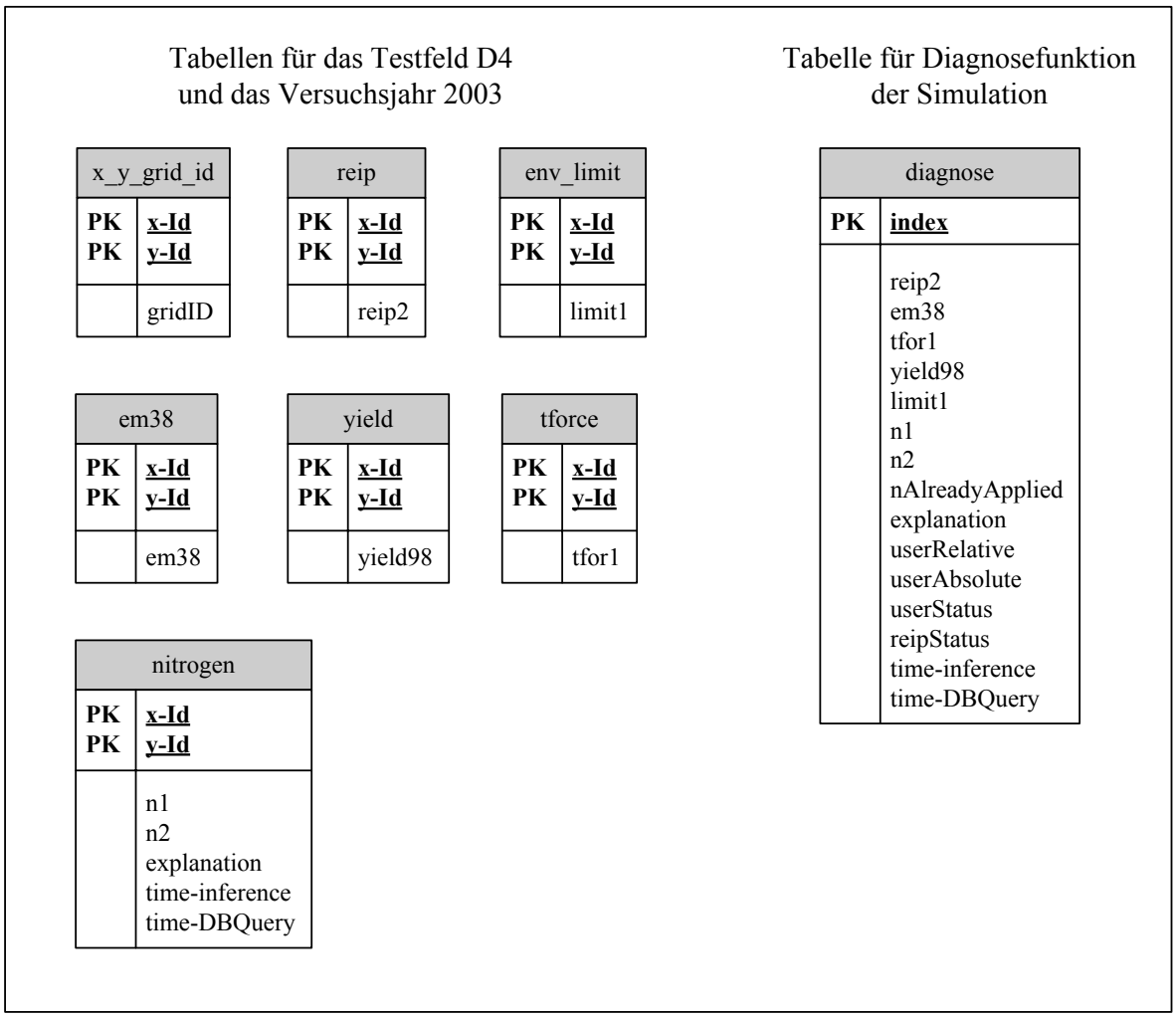


Abbildung A.8: Tabellen des relationalen Datenmodells der Simulation

Die Tabellen im linken Teil der Abbildung repräsentieren das Versuchsfeld mit den teilflächenspezifischen Attributen und die Tabelle im rechten Teil der Abbildung dient der Diagnose- und Testfunktionalität. Der Name der Relation, gleichbedeutend der jeweiligen Tabellenüberschrift, ist in den grauen Feldern aufgeführt. Die Primärschlüssel sind jeweils in dem anschließenden Tabellenabschnitt als unterstrichener Schriftzug angeführt. Im abschließenden Tabellenteil folgen jeweils die Attribute.

Das Versuchsfeld D4 wurde in ein x,y-Raster aufgeteilt, wobei jedem Rasterelement ein eindeutiger Rasterelement-Bezeichner (*gridID*) zugeordnet wurde. Diese Zuordnung wird in der Tabelle *x_y_grid_id* abgebildet, deren Primärschlüssel aus der Kombination von *x-Id* für den x-Wert und *y-Id* für den y-Wert besteht. Dabei erstreckt sich der Wertebereich von *x-Id* auf die ganzen Zahlen von 1 bis 11 und für *y-Id* von 1 bis 20.
In der Tabelle *reip* werden die einzelnen REIP-Vegetationsindexwerte zur 2. N-Gabe für jedes Rasterelement vorgehalten. Der Primärschlüssel ist wiederum die Kombination von *x-Id* und *y-Id*. Die teilflächenspezifische Ertragsinformation aus dem Jahre 1998 wird in der Tabelle *yield*, die bodenbezogenen Kartendaten zur EM38- und zur Zugkraftmessung werden in den Tabellen *em38* und *tforce* gespeichert. Die kartierte Information über einen hypothetischen Begrenzungswert zugunsten des Umweltschutzes wurde in der Tabelle *env_limit* hinterlegt. Die Tabelle *nitrogen* speichert in der bereits mehrfach beschriebenen Form die teilflächenspezifischen Stickstoffgaben der 1. N-Gabe, d.h. das Attribut *n1*, und die Sollwertempfehlung für den Applikationswert der 2. N-Gabe. d.h. das Attribut *n2*. Je nach Zeitpunkt des Lese- bzw. Schreibzugriffes kann *n2* jedoch auch die auf der Teilfläche bereits ausgebrachte Düngergabe *nAlreadyApplied* repräsentieren. Nicht ganz konsistent mit dem bisher angewandten Schema werden in der *nitrogen*-Tabelle auch die zur Sollwertempfehlung gehörenden Dokumentations- und Diagnoseattribute *explanation*, *time-inference* und *time-DBQuery* abgespeichert. Dabei ist *explanation* der Erklärungstext aus dem `explanation-slot` des Fakts für die Applikationssollwertempfehlung, der alle Erklärungstexte der an der Entscheidung beteiligten Fakten zusammenfasst. Wurde bei Simulationsstart die Diagnosefunktion aktiviert, so werden die Gesamtdauer der Schlussfolgerungsprozesse (*time-inference*) aller beteiligten Wissensmodule und die Dauer der lesenden sowie schreibenden Datenbankzugriffe (*time-DBQuery*) für einen Zyklus gemessen und in der Datenbank hinterlegt. Das Attribut *explanation* ist als Dokumentation für den Nutzer noch stimmig in der Tabelle *nitrogen*, die beiden letzten Attribute sind jedoch eindeutig der Diagnosefunktion zuzuordnen und prinzipiell fehl am Platz[83].
Diese beiden Zeitmessungen sind essentielle Bestandteile der Tabelle für die Diagnose- und Testfunktionalität. Unter der fortlaufenden Nummerierung *index* in der Funktion eines Primärschlüssels werden alle an einem Durchlauf des Entscheidungsfindungsprozesses beteiligten Eingangsinformationen, Zwischenergebnisse und der resultierende Erklärungstext zusammengefasst. Dabei handelt es sich um die Attribute, die für die Tabellen des linken

[83] Diese Funktionalität wurde bereits in einer frühen Phase der Implementierung eingebaut und nach Integration der Diagnosefunktion nicht mehr entfernt.

Teils der Abbildung bereits aufgeführt wurden, einschließlich der Zeitmessungen. Zusätzlich wird ein Teil der Nutzereingaben, die über die GUI eingestellt wurden, dokumentiert. Dies sind der Prozessführungsmodus *userStatus*, der Übersteuerungsfaktor *userRelative* für die Applikationsrate, die Absolutwert-Vorgabe *userAbsolute* für die Applikationsrate und die Angabe *reipStatus*, die widergibt, ob Online-Sensorik mitberücksichtigt wurde. Weiterhin wird das Zwischenergebnis einer auf der Teilfläche bereits ausgebrachten Düngergabe als *nAlreadyApplied* abgelegt.

Aus Sicht der „*In-field Controller*"-Simulation wird auf den Datenbankteil „Versuchsfeld" vor allem lesend zugegriffen, mit Ausnahme der Tabelle *nitrogen*, bei der einige Attribute auch geschrieben bzw. aktualisiert werden. Der Diagnoseteil der Datenbank wird nur über Schreibzugriffe angesprochen. Das Bestücken der Datenbank mit den Werten des Versuchsfeldes erfolgte in einem absätzigen Verfahren außerhalb der Simulation über Standardwerkzeuge für die angewandte Datenbank-Implementierung. Gleiches gilt auch für die nachträgliche Auswertung der Attributwerte des Diagnoseteils.

Bei der Auswahl eines relationalen Datenbanksystems zur Umsetzung des beschriebenen Schemas fiel die Wahl auf das Open Source Produkt MySQL (Version 4.1) mit dem in einem anderen Projekt am Fachgebiet für Technik im Pflanzenbau [Rot06] bereits gute Erfahrungen gemacht wurden. Eine ausführliche Dokumentation des Produktes ist über den zugehörigen Internetauftritt zugänglich [Mys11]. Für die vorliegende Arbeit ist jedoch nur von Bedeutung, dass die Datenbank grundlegende Funktionen der internationalen Norm [ISOIEC9075] zum Stand 1999 erfüllt, d.h. SQL:1999. Einen Überblick über die Datenbanksprache-Normen der Reihe ISO/IEC 9075 gibt [OP04].

Zur Integration der Datenbankanbindung in die „*In-field Controller*"-Simulation wurde auf den nach WHITE ET AL. (1999) [WFCHH99] De-facto-Standard JDBC™ für den Zugriff auf SQL-Datenbanken zurückgegriffen. Die JDBC™ Schnittstelle ist ein sogenanntes *call-level interface* und ist Teil des Java Core API. MySQL unterstützt diese Schnittstelle und lässt sich auf diesem Weg aus Java aus ansprechen. Einen Überblick über das zugehörige Zusammenspiel von SQL-Norm und Java findet sich beispielsweise bei [Deß01], ein tiefgehende Betrachtung und umfangreiche Dokumentation von JDBC™ findet sich bei [WFCHH99]. Für die Simulationsimplementierung wurde somit der Weg gewählt, Datenbankzugriffe in Java zu realisieren. Über entsprechende Objektdefinitionen und Exemplar-Erzeugungen innerhalb JESS wurde diese Funktionalität auf dieser Ebene zugänglich gemacht. Einerseits wurde die minimale Funktionalität zum allgemeinen Zugriff auf eine MySQL-Datenbank implementiert, anderseits wurden wenige „*In-field Controller*"-

spezifische Lese- und Schreibzugriffe umgesetzt. Der allgemeine Teil setzte das Grundprinzip des Datenbankzugriffs aus Verbindungsaufbau, Erzeugung eines „*SQL Statement*“, Ausführung des „*SQL Statement*“ und der Verarbeitung des Resultats um. Die spezifischen Funktionen wurden mittels dreier JESS-Funktionen umgesetzt, deren Funktionalität für die Simulation in Java implementiert wurde. Die JESS-Funktionen sind in Tabelle A.3 mit den entsprechenden Erläuterungen aufgelistet.

Tabelle A.3: Spezifische Lese- und Schreibzugriffe auf die Datenbank

JESS-Funktion	Beschreibung
sql-read-data (?xId ?yId)	Diese Funktion ermöglicht den Lesezugriff auf die Attributwerte bzw. Attributausprägungen der Tabellen (Relationen) des Versuchsfeldes, die zu einem ausgewählten Rasterelement gehören. Das Rasterelement wird durch die Variablen ?xId und ?yId bestimmt. Diese JESS-Funktion greift wiederum auf Funktionalität der implementierten Java-Klasse `SqlDatabase.java` zurück, womit über „SELECT“ -„*SQL Statements*“ Daten aus einer Relation, zu einem bestimmten Attribut und für den Primärschlüssel *x-Id* und *y-Id* gelesen werden können.
sql-docu-n2-and-explantion (?xId ?yId)	Diese Funktion ermöglicht den Schreibzugriff auf die Attributwerte bzw. Attributausprägungen der Tabelle (Relation) `nitrogen` des Versuchsfeldes, die zu einem ausgewählten Rasterelement gehören. Das Rasterelement wird durch die Variablen ?xId und ?yId bestimmt. Diese JESS-Funktion greift wiederum auf Funktionalität der implementierten Java-Klasse `SqlDatabase.java` zurück, womit über „UPDATE –„*SQL Statements*“, die Werte für die Attribute *n2*, *explanation*, *time-inference* sowie *time-DBQuery* in der Relation `nitrogen` für den Primärschlüssel *x-Id* und *y-Id* aktualisiert bzw. geschrieben werden können.
sql-docu-insert-diagnose (?index)	Diese Funktion ermöglicht das Einfügen von Attributwerten bzw. Attributausprägungen der Tabelle (Relation) `diagnose`, die über den Primärschlüssel *index*, entsprechend ?index, identifiziert werden. Diese JESS-Funktion greift wiederum auf Funktionalität der implementierten Java-Klasse `SqlDatabase.java` zurück, womit über „INSERT“-„*SQL Statements*“, die Werte für die spezifizierten Attribute eingefügt werden können.

In der JESS-Funktion `sql-read-data` wurde zusätzlich zum reinen Datenbankzugriff auch die Wertezuweisung an die zugeordneten JavaBeans-Objekte und GUI-Elemente integriert.

A.4.5 Mensch-Maschine-Schnittstelle

Die realisierte Mensch-Maschine-Schnittstelle wurde bereits am Beginn des Implementierungs-Kapitels (vgl. Kap. 5.2.3) eingeführt. Sie besteht zum einen aus der für den Endnutzer gedachten interaktiven GUI und der Ergänzung einer Kommandozeilenumgebung

auf JESS-Ebene für den Experten bzw. Wissensingenieur zum anderen. Für die Entwicklung der Mensch-Maschine-Schnittstelle wurden neben der Softwaredokumentation von JESS und des Java SDK die entsprechenden Kapitel bei [Bro03], [Fri03], [Ull03] zu Rate gezogen.

Abbildung 5.7 zeigt die GUI mit ihren gruppierten Ein- und Ausgabeelementen, deren Bedeutung und Verwendung bereits im entsprechenden Ergebnisteil dieser Arbeit beschrieben wurde.

Die grundsätzliche statische Struktur der GUI, d.h. die logische Platzierung der einzelnen GUI-Elemente und deren (hierarchische) Beziehungen untereinander, gibt die sogenannte *containment hierarchy* wider. Abbildung A.9 zeigt diese Hierarchie für den Teil, der hauptsächlich in JESS implementiert wurde.

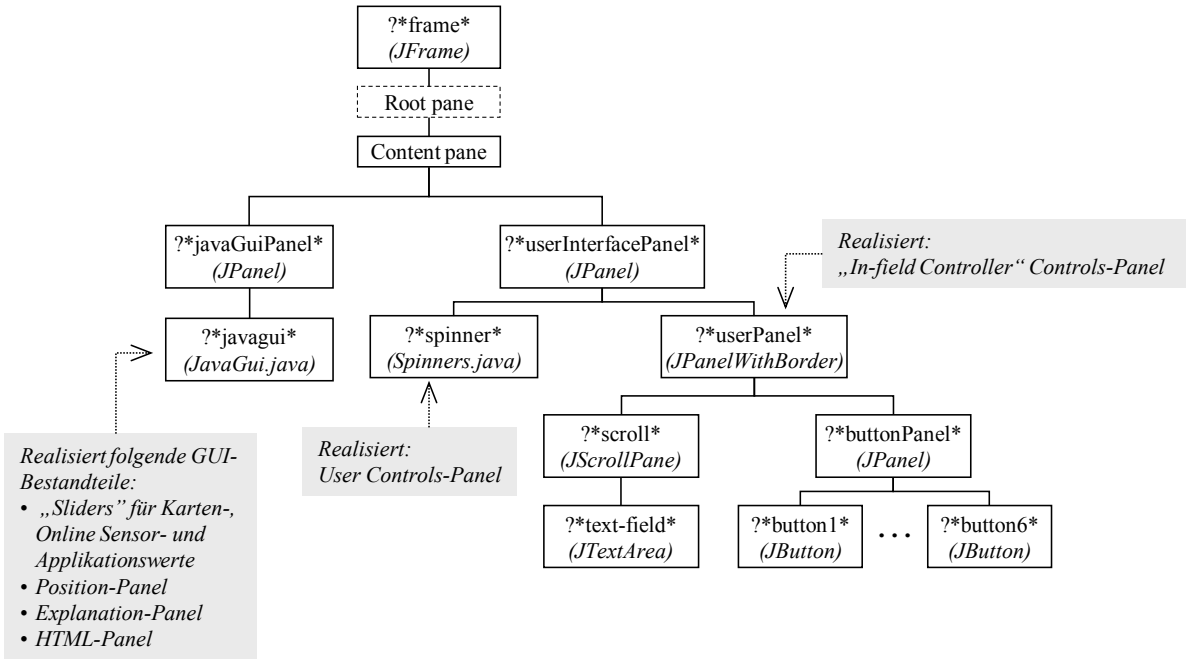


Abbildung A.9: „*Containment*“-Hierarchie der GUI - Überblick auf JESS-Ebene

Die einzelnen Rechtecke repräsentieren einzelne GUI-Elemente aus der Menge der Swing-Komponententypen *Top-Level Container* (z.B. *JFrame*), *Intermediate Container* (z.B. *JPanel*) oder *Atomic Components* (z.B. *JButtons* oder *JTextArea*). In der ersten Zeile der Rechtecke wird dabei in JESS-Notation die Variable, die ein entsprechendes GUI-Objekt referenziert, aufgeführt. In der zweiten Zeile wird in Klammern die Klasse dargelegt, auf der das GUI-Objekt basiert. Die mit *„J“* beginnenden Klassenbezeichner sind Standard-Swing-Klassen. Die mit *„.java“* endenden Bezeichner weisen auf eigene Klassendefinitionen hin, die für die vorliegende Simulation entworfen und direkt in Java umgesetzt wurden.

Die Hierarchie startet mit dem Hauptfenster `?*frame*`, dem *Top-Level Container*, der (automatisch) eine *root pane* mitbringt, die wiederum die *content pane* beinhaltet, auf der alle

sichtbaren Elemente platziert werden müssen. Auf der nächstniedrigeren Ebene werden die beiden *Intermediate Container* `?*javaGuiPanel*` und `?*userInterfacePanel*` angeordnet. In `?*userInterfacePanel*` werden wiederum zwei weitere *Intermediate Container*, `?*spinner*` und `?*userPanel*`, logisch platziert, die einerseits das *„User Controls-Panel“* und anderseits das *„Infield Controller Controls-Panel“* realisieren. `?*spinner*` referenziert hierzu ein Objekt der Klasse *Spinners*, die wiederum die Klassendefinition *JPanel* erweitert. Bei einem *JPanel* handelt es sich um den Inhalt einer Zeichenfläche bzw. ein Panel-Widget. `?*userPanel*` basiert auf einem *JPanelWithBorder*-Objekt, in dem neben einem Container (`?*buttonPanel*`) für sechs Schaltflächen ein rollbares/(scrollbares) Textausgabefenster mittels *JScrollPane*- und *JTextArea*-Objekten angeordnet wird. Mit der Anordnung der sechs Variablen `?*button1*` bis `?*button6*`, die *JButton*-Objekte referenzieren, wird im JESS-Teil der GUI die tiefste Ebene erreicht.

Das *„User Controls-Panel“* besteht einerseits aus zwei Optionsfeldern, auch Kontrollfeldgruppen genannt, für die Auswahl zwischen automatischem und manuellem Modus sowie der Auswahl hinsichtlich der Verfügbarkeit des Online Sensors für den REIP-Index. Anderseits sind zwei Eingabefelder (Editoren) vorhanden, deren Wert über zwei kleine Pfeile verändert werden kann. Abgebildet in der *Spinners*-Klasse wird diese Funktionalität in den *Intermediate Container allPanel, spinnerRelativePanel und spinnerAbsolutePanel* logisch angeordnet. Die Optionsfelder werden in *allPanel* und dort nochmals in Containern der Klasse *JPanel* platziert, wo auf tiefster Ebene die Funktion über *JRadioButton* und *ButtonGroup*-Objekte realisiert wird. Innerhalb *spinnerRelativePanel* und *spinnerAbsolutePanel* sind jeweils *JSpinner und JLabel*-Objekte angeordnet. Die Wertebereiche und die Schrittweite der *JSpinner*-Objekte werden mit einem Zahlenmodell (*SpinnerNumberModel*) vorgegeben.

Für die Realisierung der GUI-Teile *„Sliders for Maps“, „Position Panel”,* *„Explanation (SUM_UP & DETAILS) Panel”, „Slider Online Sensor REIP“, „HTML Panel”* und *„Slider Application N-setpoint“*, ist ein Objekt der Klasse *JavaGui,* abermals eine Erweiterung von *JPanel,* der oberste *Intermediate Container*. Auf der nächsten Ebene sind weitere *JPanel*-Objekte angeordnet, die die Gruppierung auf die vorher angeführten GUI-Teile vornehmen und auch für die Positionierung innerhalb des Simulationsfensters entscheidend sind. In einem dieser Container wurden die einzelnen Objekte der Klasse *Sliders* für die Ein- und Ausgabe der Kartenwerte angesiedelt. In dem „*Sliders*-Container“ wird auf der Stufe der *Atomic Components* die Funktion über *JLabel*-, *JFormattedTextField*- und einem *JSlider*-Objekte realisiert. Mittels *JLabel* lässt sich ein Informationstext, mittels *JFormattedTextField* ein

validiertes Eingabefeld erzeugen. Auch die Umsetzung der GUI-Teile *„Slider Online Sensor REIP“* und *„Slider Application N-setpoint“* beruht auf der Nutzung der *Sliders*-Klasse. Teile der bereits beschriebenen *Spinners*-Klasse wären hilfreich bei der Realisierung des *„Position-Panel”*. Jedoch werden dort keine Optionsfelder benötigt und es wurde daher eine eigene Klasse *Position* definiert. In der zugehörigen „*Containment hierarchy*” kommen auf der untersten Ebene Objekte der Klassen *JLabel*, *JTextArea* und *JSpinner* zum Einsatz. Für das „*Explanation (SUM_UP & DETAILS) Panel”* wurde mit *ExplanationPanel* eine weitere neue Unter- bzw. Subklasse von *JPanel* gebildet. Ein davon erzeugtes Objekt dient als *Intermediate Container* für zwei rollbare mehrzeilige Textfelder. Die entsprechenden Exemplare basieren auf den Swing-Klassen *JScrollPane* und *JTextArea*. Die Funktionalität eines minimalen Browsers für das *„HTML-Panel”* wurde über eine Klasse *HtmlPanel* erzeugt. *HtmlPanel* war wiederum als Subklasse von *JPanel* angelegt. Jeweils ein Exemplar der *JEditorPane*-Klasse und ein Exemplar der *JScrollPane*-Klasse sind die Hauptbestandteile des *HtmlPanel*-Exemplars. Über *JScrollPane* wird der (vertikal und horizontal) rollbare Fensterausschnitt erzeugt und über *JEditorPane* wird eine leistungsfähige Textkomponente integriert, die (u.a.) in HTML codierte Inhalte unterstützt.

Mit der Beschreibung des logischen Aufbaus der GUI wurde bereits auch ein Blick auf die grobe Ausgestaltung der für den Nutzer sichtbaren Benutzerschnittstelle gegeben. Die konkrete Ausgestaltung hinsichtlich Farbgebung, Fonts, Formatierungen, teilweise Größendefinitionen und Voreinstellungen wurden sowohl in Java als auch in JESS über die entsprechenden Methoden der aufgelisteten GUI-Komponenten bestimmt. Mittels der Layoutmanager-Funktionalität wurden die Positionierungen gesetzt. Von Javas Standardlayouts kamen hierfür *BorderLayout*, *GridLayout* und *BoxLayout* zum Einsatz. Für das prinzipielle Aussehen der Komponenten wurde das standardmäßig eingestellte Java - „*Look & Feel*“ vom Typ „*Metal*“ beibehalten.

Die bisherige GUI-Beschreibung gibt nur Aufschluss über den statischen Aufbau der Benutzerschnittstelle, die Realisierung des dynamischen Verhaltens soll daher nachfolgend noch kurz beleuchtet werden. Den Interaktionsmöglichkeiten des Nutzers mit der GUI liegt neben grafischen Interaktionskomponenten, den sogenannten Widgets (*Window Elements*) ein Modell zur Behandlung von Ereignissen, wie z.B. dem Betätigen eines Tasters, zugrunde [Ull03]. Java stellt für die Ereignisbehandlung das Abstract-Window-Toolkit (AWT) zur Verfügung.

Nach BRONSON (2003) [Bro03] funktioniert die Ereignisbehandlung in Java nach dem *event delegation model* und wurde in dieser Arbeit bereits bei der Behandlung der Themen

Faktenbasis mit *shadow facts* und der Prozessumgebungssimulation über JavaBeans vorgestellt. Für die Anwendung bei der GUI lässt sich das Verfahren in Anlehnung an [Bro03] knapp zusammenfassen:

1. Erzeugung einer (sichtbaren) GUI-Komponente, die als Ereignisauslöser bzw. Ereignisquelle dient.
2. Bereitstellung einer Funktionalität zur Reaktion auf das Ereignis (*Event Handler*), indem:
    a. Definition einer *Event Handler*-Klasse, d.h. des Interessenten (*Listener*) an dem Ereignis, der die (Re)aktion ausführt.
    b. Erzeugung eines Exemplar dieser *Listener*-Klasse, nachfolgend *Listener-Objekt* genannt.
    c. Registrierung des *Listener*-Objekts bei der Ereignisquelle.

Nach diesem Verfahren wurden auf JESS-Ebene Reaktionen auf die Ereignisse Schließen des Hauptfensters und Betätigung eines der sechs Schaltflächen des *„In-field Controller Controls Panel“* implementiert. Hierzu wurden die aus Java bekannten *WindowListener*- und *ActionListener*-Schnittstellen (*interfaces*) umgesetzt. Für die entsprechende JESS-Implementierung stellt sich dies für die Schaltfläche „*infer*“ folgendermaßen dar:

JESS-Code 10: GUI - Ereignisbehandlung für den Taster „infer“

```
(defglobal ?*button1* = (new JButton "infer "))
```

Es wird ein Exemplar der JButton-Klasse mit der Beschriftung „infer “ erzeugt und in der globalen JESS-Variable `?*button1*` *gehalten.*

= Schritt 1 (Erzeugung der Ereignisquelle)

```
(deffunction button1 (?evt)
  (read-inputs-from-gui)
  (assert (trigger))
  (run-system)
   ... )
```

Der „Funktionscode“ des Event Handler `button1` *wird in Form einer JESS- Funktion definiert. Dabei werden weitere Funktionen zum Lesen der GUI- Eingabewerte, dem Einbringen des* `(trigger)` *- Fakts, dem Start des Schlussfolgerungsprozesses sowie weiterer nicht aufgeführter Funktionen zur bedingten Debug- und Dokumentationsfunktionalität eingeschlossen.*

= Teil des Schritt 2a, (Definition eines Event Handlers)

```
(call ?*button1* addActionListener
  (new jess.awt.ActionListener button1 (engine)))
```

Definition einer Listener-Klasse ist in jess.awt.ActionListener erfolgt. Der „Funktionscode“ wurde bereits über `button1` *definiert.*

= Teil des Schritt 2a, (Definition eines Event Handlers)

In der inneren Klammer der Codezeile wird ein entsprechendes Listener-Objekt mit ActionListener-Schnittstelle und der Funktionalität von `button1` *erzeugt.*

= Schritt 2b (Erzeugung eines Listener-Objektes)

Mit dem `call`*-Aufruf wird dieses Listener-Objekt bei der Ereignisquelle* `?*button1*` *registriert.*
= Schritt 2c, (Registrierung des Listener-Objektes)

Nach dem vorgestellten Muster wurde das ereignisbasierte Verhalten der weiteren Schaltflächen und des Hauptfensters realisiert. Ergänzend sei angemerkt, dass die Ereignisbehandlung für den Fall des Schließens des Hauptfensters darin besteht, dass neben Schließen der GUI, eine geöffnete Datenbankverbindung geschlossen, die „*In-field Controller*"-Simulation beendet und auch die JESS-Umgebung verlassen wird. Bereits am Anfang des Kapitels 5.2.3 wurde die Funktionalität der *Event Handler* jedes einzelnen Tasters aus Nutzersicht beschrieben. Auf einige dafür definierte JESS-Funktionen wird im weiteren Verlauf dieses Kapitels noch eingegangen.

Neben der Ereignisbehandlung im JESS-Teil der GUI war auch für den Teil, der in Java entwickelt wurde, eine Implementierung von ereignisbasierten Verhalten nötig. Exemplare von *JSlider*, *JSpinner,* Optionsfeldern und einem Html-fähigen Editor sind dadurch erst von Wert. Hierzu wurden für folgende Klassen die angeführten Schnittstellen (*interfaces*) umgesetzt:

- *Spinners.java* implementiert *ChangeListener* und *ActionListener,*
- *Position.java* implementiert *ActionListener,*
- *Sliders.java* implementiert *ChangeListener* und *PropertyChangeListener,*
- *HtmlPanel.java* implementiert *HyperlinkListener.*

Ein weiterer Aspekt von dynamischen Verhalten der GUI wurde bisher noch nicht behandelt. Über das *„HTML Panel",* realisiert über ein Exemplar der *HtmlPanel*-Klasse, ist es möglich, in dem zentralen Fensterausschnitt der Simulation einen wechselbaren Inhalt darzustellen. Die entsprechenden Inhalte wurden in Html formuliert. Die resultierenden Html-Dateien und einzubindenden Grafiken wurden in einem lokalen Verzeichnis des jeweiligen Testrechners gespeichert.

Bei der Ereignisbehandlung wurde bereits darauf hingewiesen, dass im Zusammenhang mit der Mensch-Maschine-Schnittstelle verschiedene JESS-Funktionen realisiert wurden. Ein Teil davon fasst Schreib- und Lesezugriffe auf die GUI-Elemente zusammen, ein anderer Teil definiert Funktionen, die in den *Event Handler* der Tasten verwendet werden. Tabelle A.4 listet die wichtigsten Funktionen mit ihrer Kurzbeschreibung auf.

Tabelle A.4: GUI-relevante definierte JESS-Funktionen

JESS-Funktion	Beschreibung
read-controls-of-user-panel	Diese Funktion ermöglicht die Einstellungen des *„User Controls Panels"* auszulesen und die entsprechenden Variablen der Prozessumgebungssimulation zu aktualisieren. Aufgrund der anderen Realisierungsform für den Status der Verfügbarkeit eines Online-Sensors wird in diesem Fall direkt ein „`(yesREIP)`"- oder „`(noREIP)`"-Fakt in die Faktenbasis eingestellt.
read-inputs-from-gui	Diese Funktion ermöglicht die Einstellungen der GUI-Teile *„Sliders for Maps"* und *„Slider Online Sensor REIP"* auszulesen und die entsprechenden Variablen der Prozessumgebungssimulation zu aktualisieren.
generate-random-inputs	Diese Funktion erzeugt für alle Kartenwerte und den Online-Sensor Zufallswerte, aktualisiert die entsprechenden Variablen der Prozessumgebungssimulation und stellt diese Werte auch in den GUI-Teilen *„Sliders for Maps"* und *„Slider Online Sensor REIP"* ein.
get-values-for-docu-diagnose	Diese Funktion frägt alle für die Tabelle „Diagnose" der Datenbank nötigen Attributwerte von der Prozessumgebungssimulation ab und legt sie in JESS-Variablen ab, die für die Dokumentation und Diagnose genutzt werden (vgl. Kap. A.4.4). Ausgenommen sind die Werte für die Zeitmessung und der Status der Verfügbarkeit eines Online-Sensors, der direkt von dem entsprechenden GUI-Optionsfeld gelesen wird. Auch wird der Inhalt für die Variable des Erklärungstextes „explanation" des Schlussfolgerungsprozesses direkt in den Aktionsteilen des *„SUM_UP"*-Regelwerk-Modules belegt.

Die Schreibzugriffe von JESS aus auf die verschiedenen Textausgabefelder der GUI sowie der Schreibzugriff auf *„Slider Application N-setpoint"* wird direkt über die implementierten Methodenzugriffe(, wie in vorhergehenden Kapiteln prinzipiell gezeigt,) durchgeführt. Dies gilt für das *„Position Panel"* neben den Schreib-, auch für die Lesezugriffe.

Der Teil des Kapitels, der sich mit der GUI beschäftigt soll mit einer knappen Zusammenstellung des „Funktions-Code" der *Event Handler* für die Schaltflächen des *„In-field Controller Controls Panel"* abgeschlossen werden. Auch diese Zuordnung wird in Tabellenform (vgl. siehe Tab. A.5) durchgeführt:

Tabelle A.5: Event-Handler des „In-field Controller Controls Panel“

Schaltfläche (GUI-Komponente)	Zugeordnete Aktionen bzw. Funktionen	Bemerkung
„infer“ `(?*button1*)`	• Textausgabe in `?*userPanel*` • (read-controls-of-user-panel) • (read-inputs-from-gui) • (assert (trigger)) • (run-system) • Zeitmessung für Inferenz-Prozess • Optional: • (get-values-for-docu-diagnose) • (sql-docu-insert-diagnose) • Zeitmessung für Datenbankzugriff	Die einzelnen Funktionen wurde bereits in vorherigen Abschnitten des Kapitels 5.2.3 vorgestellt bzw. eingeführt. Optionale Teile werden über entsprechende „Schaltervariablen“ und IF … THEN… Konstruktionen aktiviert bzw. desaktiviert. Zeitmessung auf JESS-Ebenen unter Nutzung der Systemzeitabfrage auf Millisekundenbasis: `(call System currentTimeMillis)`
„DB-Query" `(?*button2*)`	• Abfrage/Entscheidung nach aktivierter Datenbankverbindung • Textausgabe in `?*userPanel*` • Lesen der x, y-Werte des im *„Positon Panel”* eingestellten Rasterelements • (read-controls-of-user-panel) • (sql-read-data ?x ?y) • Update des GridID-Feldes im *„Position Panel”* • (assert (trigger)) • (run-system) • (get-values-for-docu-diagnose) • (sql-docu-n2-and-explanation ?x ?y) • Zeitmessung für Inferenz-Prozess • Zeitmessung für Datenbankzugriff	Siehe Eintrag in Spalte „Beschreibung“ für die „infer“-Schaltfläche.
„random“ `(?*button3*)`	• Textausgabe in `?*userPanel*` • (read-controls-of-user-panel) • (generate-random-inputs)	Siehe Eintrag in Spalte „Beschreibung“ für die „infer“-Schaltfläche.
„start“ `(?*button4*)`	• Textausgabe in `?*userPanel*` • (`?*timer1*` start)	Starten des zyklischen Prozesses, realisiert durch das *Timer*-Objekt `?*timer1` mit dessen integrierter Ereignisbehandlung.
„stop“ `(?*button5*)`	• Textausgabe in `?*userPanel*` • (`?*timer1*` stop)	Stoppen des zyklischen Prozesses, realisiert durch das *Timer*-Objekt `?*timer1` und dessen integrierter Ereignisbehandlung.
„ON/OFF" `(?*button6*)`	• Textausgabe in `?*userPanel*` • Bedingte Wertzuweisung für den Status des „Notausschalters“ in der Simulation der Prozessumgebung	Realisierung der Funktionalität eines Wechselschalters, dessen Zustand in der zugehörigen JavaBean `?emergency-stop-status` abgebildet wird. Die weitere Funktionsweise/Wirkung wurde in Kapitel A.4.1 bereits ausgeführt.

Bereits bei der Beschreibung der Simulation aus Anwendersicht wurde darauf hingewiesen, dass unter Nutzung der JESS-Kommandozeilenumgebung auch eine (rudimentäre) Konsolenschnittstelle für eine minimale Möglichkeit zum Dialog mit dem Nutzer umgesetzt wurde. Abbildung 5.8 gibt einen Eindruck über die Ausgestaltung der Konsolenschnittstelle. Zugleich zeigt es auch wie die Simulation gestartet werden kann, vorausgesetzt die Java und JESS-Umgebungen mit den geeigneten Umgebungsvariablen sind installiert und der MySQL-Server-Dienst wurde auf dem System ebenfalls gestartet. Für den nach dem Start der JESS-Umgebung und dem anschließenden Beginn des Simulationsprogramms öffnenden Nutzerdialog bezüglich unterschiedlicher Programmoptionen bzw. -konfigurationen wurde eine JESS-Funktion `(ask-user)` implementiert, die einen Fragetext auf der Kommandoschnittstelle ausgibt, eine Nutzereingabe ermöglicht, bei dieser Antwort eine Typprüfung durchführt und eine gültige Antwort in entsprechenden Konfigurationsvariablen ablegt. In der Abbildung erscheint der Nutzerdialog, ob auch die Tabelle *„Diagnose“* in der Datenbank abgelegt werden soll, nicht, da dies in Abhängigkeit von einer positiven Auswahl der Datenbankverbindungsoption ist. Weiterhin wird aufgrund der aktivierten Ausgabe von Debug-Nachrichten (*„Commandline Printout“*) der Beginn des Schlussfolgerungs-Zyklus dokumentiert. Der Kontrollfluss bzw. Fokuswechsel zwischen den Regelwerkmodulen wie auch die wichtigsten Ergebnisse, die den jeweiligen Erklärungstexten in den Aktionsteilen der „*SUM_UP*“-Regeln entsprechen, werden ausgegeben. Auch die Ergebnisse der Zeitmessungen werden angezeigt. Wäre die Option „*Debug“*-Nachrichten (*„Debug messages for JavaBeans active“*) der Simulation der Prozessumgebung aktiviert worden, so würden auch die Attributwerte der JavaBeans-Exemplare ausgegeben werden. Die Grundstrukturen zur Realisierung der Ausgaben auf der JESS-Konsole sind die `printout` Funktion von JESS und innerhalb der JavaBeans-Exemplare der Methodenaufruf *System.out.println()*. Die entsprechenden Konfigurationsvariablen kontrollieren über Bedingungsabfrage die Ausführung der Konsolen-Textausgabefunktionen.