Technische Universität München
Lehrstuhl für Raumfahrttechnik

Eine objektorientierte Modellierungsmethode für die simultane Systementwicklung

Michael Markus Schiffner

Vollständiger Abdruck der von der Fakultät für Maschinenwesen
der Technischen Universität München zur Erlangung
des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. H. Baier
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. rer. nat. U. Walter
	2.	Univ.-Prof. Dr.-Ing. K. Bender

Die Dissertation wurde am 27. Juni 2007 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 22. November 2007 angenommen.

Meinen Eltern in tiefer Zuneigung und Dankbarkeit.

Danksagung

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Raumfahrttechnik an der Technischen Universität München vom November 1996 bis Sommer 2007. Daher möchte ich Prof. U. Walter für die Betreuung meiner Arbeit danken genauso wie Prof. E. Igenbergs für die jahrelange Unterstützung meiner Arbeit dankbar bin. Ich möchte mich bei beiden Professoren für ihr Vertrauen in meine Arbeit und die inspirierende Atmosphäre am Lehrstuhl bedanken. Durch diese Atmosphäre war es mir möglich meine Ideen mit meinen Kollegen zu diskutieren, zu strukturieren und neuen Ideen nachzugehen.

Prof. K. Bender danke ich für die Mitberichterstattung und Prof. H. Baier für die Übernahme des Prüfungsvorsitzes.

Allen meinen Kollegen danke ich für ihre vielfältigen Anregungen, ohne die diese Arbeit sicher nicht zustande gekommen wäre.

Zuletzt möchte ich mich bei meiner Familie und meinen Freunden bedanken, die mir auf meinem Weg stets Unterstützung und Förderung gewährt haben und auch, dass sie während der heißen Phase Geduld und Verständnis gezeigt haben.

München, im Juni 2007 Michael M. Schiffner

Zusammenfassung

Während der Konzeptphase von neuen Raumfahrtsystemen wird heutzutage multidisziplinär und modellbasiert gearbeitet. Die integrierte Arbeitsweise wird bereits durch das Concurrent Engineering erfolgreich umgesetzt. Für die modellbasierten Konzeptstudien werden fachdisziplinübergreifende Systemmodelle benötigt. Im Rahmen dieser Arbeit werden die derzeitigen Ansätze für diese Systemmodelle bei der ESA und NASA sowie alternative Systemmodellierungsmöglichkeiten wie UML und SysML gezeigt. Aufbauend auf den derzeitigen Ansätzen und den Erfahrungen des Autors stellt die Arbeit eine neue Modellierungsmethode bestehend aus einer objektorientierten Systemmodellierung, einer Softwarearchitektur und einem Datenmodell vor. Die neue Methode wurde in einer Software implementiert, die es dem Autor an dem Beispiel eines Raumfahrtsystems ermöglicht die vielfältigen Möglichkeiten bei der Systemmodellierung und Analyse durch die neue Modellierungsumgebung aufzuzeigen.

Synopsis

During the concept phase of new space systems integrated and model based engineering is state of the art. The integrated way of working is already successfully implemented by using the Concurrent Engineering approach. For the model based concept studies technical discipline independent system models are needed. In the context of this work the present implementation of these system models from ESA and NASA as well as alternative system modeling possibilities such as UML and SysML are shown. Based on the present approaches and the experiences of the author the work presents a new modeling concept consisting out of an object-oriented way of system modeling, software architecture, and a data model. The new concept was implemented in a software tool, so that the author is able to point out the various possibilities during space system modeling and analysis by the new modeling environment.

Inhaltsangabe

Abkürzungen

A

ACE	Airbus Concurrent Engineering
AR	Acceptance Review; Abnahme Review

B

BaiCES	Bavarian Center of Excellence in Satellite Constellations
BFS	Bayerische Forschungsstiftung

C

CAD	Computer Aided Design
CAE	Computer Aided Engineering
CDC	Concurrent Design Center
CDC	Customer Demand Center bei Fairchild Dornier Company
CDF	Concurrent Design Facility
CDR	Critical Design Review; Kritischer Design Review
CE	Concurrent Engineering
CEW06	2nd Concurrent Engineering for Space Applications Workshop
CNESS	Französische Raumfahrtagentur

D

DC	Design Center
DDE	Dynamic Data Exchange
DE-ICD	Design Environment for Integrated Concurrent Engineering
DJF	Design Justification File; Designbegründungsakte
DLL	Dynamic Link Library

E

ECSS	European Cooperation for Space Standardization

EDRC	Engineering Design Research Center
EP	Entscheidungspunkt
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre

F

FDC	Fairchild Dornier Company

I

ICD	Interface Control Document; Schnittstellenkontrolldokument; (ESA)
ICD	Integrated Concurrent Design (NASA)
ICE	Integrated Concurrent Engineering
IDEA	Integrated Data Exchange Architecture
IDM	Integrated Data Model
IMSE	integrierte, modellbasierte Satellitenentwicklung
INCOSE	International Council of Systems Engineering
IPO	Input-Prozess-Output
IPPD	Integrated product and process development
IPT	Integrated Product Teams
IRF	Swedish Institute of Space Physics, Uppsala
ISM	Integrierte Systemmodellierung
IV	Informationsverarbeitung

J

JPL	Jet Propulsion Laboratory

L

LAN	Local Area Network; Computernetzwerk
LRT	Lehrstuhl für Raumfahrttechnik
LSMD	Laboratory for Spacecraft and Mission Design

M

MAU	Million Accounting Units

MDR	Mission Definition Review
MIT	Massachusetts Institute of Technology
MO	Mars Observer
MOF	Meta-Object Facility Meta Model
MuSSys	Modellierung und Simulation von Systemen
MuSSat	Modellierung und Simulation von Satellitensystemen

N

NASA	National Aeronautics and Space Administration

O

OCL	Object Constrain Language
ODBC	Open Database Connectivity
OMG	Object Management Group
ORR	Operational Readiness Review

P

PACT	Product Attribute Conversation Tool
PAD	Product Attribute Database
PDC	Project Design Center
PDR	Preliminary Design Review
PRR	Preliminary Requirements Review
PSP	Projektstrukturplan

Q

QR	Qualification Review

R

R-PET	Relational Parameter Exchange Tool
RFX	Request for Change
RKMVA	Rosetta Knowledge Management Video Approach

S

SDO	Satellite Design Office
SE	Systems Engineering
SEDT	System Engineering Design Tool
SGM	Stage-Gate-Methode
SGP	Stage-Gate-Prozess
SRR	System Requirements Review
S_2C_2	Space Systems Concept Center
SSDL	Space and Systems Development Laboratory
SSTL	Surrey Satellite Technology Limited in England
STK	Satellite Toolkit
SysML	Systems Modeling Language

T

TS	Technische Spezifikation
TUM	Technischen Universität München

U

(U)CML-ed	Unified Compatibility Modeling Language - Editor
UML	Unified Modeling Language

V

VBA	Visual Basic for Applications
VU	Virtuelle Unternehmen
(v)Sys-ed	virtuelles System – Editor

W

WYSIWYG	What you see is what you get

X

XML	Extensible Markup Language
XMI	XML Metadata Interchange

Z

ZOPH	Zielsystem, Objektsystem, Prozesssystem, Handlungssystem

Formelzeichen und Einheiten

D	Dämpfungskoeffizient	$\left[N\frac{s}{m}\right]$
g	Oberflächenbeschleunigung	$\left[\frac{m}{s^2}\right]$
f	Anzahl Fachdisziplinen	[-]
i	Laufvariabel (natürliche Zahl einschließlich 0)	[-]
l	Länge	$[m]$
m	Masse	$[kg]$
μ	Gravitationskonstante	$\left[\frac{m^3}{kg\ s^2}\right]$
n	Anzahl hierarchisch gleicher Objekte (Anzahl Kinder)	[-]
R	Radius zwischen Mittelpunkt und Oberfläche	$[m]$
r	Abstand zwischen Mittelpunkt und Objekt	$[m]$
v	Geschwindigkeit	$[\frac{m}{s}]$
w	Wert eines Attributes eines Objektes	[-]
N	Normalenvektor	$[n]$
t	Zeit	$[s]$
x	beliebige Hilfsvariable	[-]

Winkel

Winkel werden in Grad angegeben und sind durch griechische Buchstaben definiert.

α	Angriffswinkel zwischen landendes Objekt und Boden	[°]
β	Aufsetzwinkel zwischen landendes Objekt und Aufsetzgeschwindigkeit	[°]
γ	Aufsetzwinkel zwischen Boden und Aufsetzgeschwindigkeit	[°]

Indizes

I	Während des Aufsetzvorgangs (abgeleitet von engl. Impact)
L	Lander
K	Komet
v	hierarchisch übergeordnetes Objekt (Vater)
k	hierarchisch gleiche Objekte (Kinder)
D	Während des Abstiegs (abgeleitet von engl. Decent)

Ein Punkt über einer Variablen bedeutet eine Ableitung nach der Zeit ($\dot{v} = \frac{dv}{dt}$)

Ein Pfeil über einer Variable kennzeichnet einen Vektor: z.B. $\vec{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

1 Einleitung

1.1 Ausgangssituation

Die Entwicklung von Raumfahrtsystemen wie Satelliten oder Trägerraketen war schon immer eine besondere Herausforderung an die Entwickler und Ingenieure. Die Herausforderung hierbei liegt insbesondere in der Gewährleitung einer hohen Zuverlässigkeit, die diese Systeme ab dem Zeitpunkt der Inbetriebnahme erreichen müssen. Fehler und Fehlfunktionen während des Betriebs können zu einem Verlust des Systems und sogar Menschenleben führen. Nachträgliche Änderungen können nur in seltenen Fällen vorgenommen werden. Das bekannteste Beispiel für einen Komplettverlust in der Vergangenheit der europäischen Raumfahrt war der erste Start der Ariane-5 (Flug 501) am 4. Juni 1996. Bei diesem Flug geriet die Rakete 36,7 Sekunden nach dem Start in einen instabilen Zustand und musste durch eine Notsprengung zerstört werden. Dabei wurde auch ihre Nutzlast, die vier Satelliten der Cluster Mission, zerstört. Die Ursache für den instabilen Zustand war ein Werteüberlauf in der Steuerungssoftware, die von der Ariane-4 auf die Ariane-5 übernommen aber nicht ausreichend getestet wurde. (Ariane-5: Learning from Flight 501 15.05.1997). Die Trageweite dieses Fehlers kann man am finanziellen Verlust aufzeigen. So hat alleine die Cluster Mission ca. 480

MAU[1] gekostet (ESA 27.06.1996 – The Cluster Mission). Die Entwicklung der Ariane-5 hat 10 Jahre gedauert und die Kosten haben sich auf mehrere Milliarden Euro summiert.

Neben der benötigten hohen Zuverlässigkeit von Raumfahrtsystemen sind bei der Entwicklung dieser Systeme auch die extremen Umweltbedingungen zu berücksichtigen. Hierzu zählen:

- Extreme Hitze-Kälte-Schwankungen
- Weltraumstrahlung, die die elektronischen Komponenten schädigt
- Mechanischen Belastungen durch Schall und Vibrationen

So wirkt bei einem Start der Ariane-5 ein durchschnittlicher Schalldruck von 140,5 dB und eine Beschleunigung in lateral Richtung von bis zu 4,55g (Perez 2004 – Ariane 5, S. 3-4, 3-2).

Um diese vielfältigen Aspekte bei der Systementwicklung zu berücksichtigen, bestehen die Entwicklungsteams der Raumfahrtsysteme aus Experten der verschiedensten Fachdisziplinen. Dadurch entsteht ein neues Problem: Die Experten müssen darauf achten, eine Gesamtsystem-optimale Lösung und keine Fachdisziplin- oder Teilsystem-optimale Lösung zu entwickeln. Bei einer Gesamtsystem-optimalen Lösung müssen alle beteiligten Fachdisziplinen ihre Teillösungen aufeinander abstimmen. Dieses Abgleichen der Teillösungen bedarf Schnittstellen zwischen allen betroffenen Fachdisziplinen. So sind maximal $f \bullet (f-1)$ unidirektionale[2] bzw. $\frac{1}{2}[f \bullet (f-1)]$ bidirektionale[3] Schnittstellen bei f beteiligten Fachdisziplinen zu berücksichtigen. So existieren bei einer in der Raumfahrt üblichen Teamgröße in der Vorentwicklung von durchschnittlich zehn Experten 45 bidirektionale Schnittstellen. Bei einer Teamgröße von 16 Experten sind es dann 120 bidirektionale Schnittstellen. Bereits bei 45 Schnittstellen wird offensichtlich, wie schwierig es ist sicherzustellen, dass alle Daten fehlerfrei zwischen den Teammitgliedern ausgetauscht werden. Die Folgen ungenau definierter Schnittstellen wurden bei der Mars Observer (MO) Mission der National Aeronautics and

[1] MAU: Million Accounting Units; Accounting Units sind Euros in Bezug auf ein bestimmtes Finanzjahr und berücksichtigen somit die Inflation und Deflation. In der Quelle fehlt eine Aussage zu dem Bezugsjahr.

[2] unidirektionale Schnittstelle: Die Information wird nur von dem Teilnehmer A an Teilnehmer B übertragen

[3] bidirektional Schnittstelle: Informationen werden von Teilnehmer A an Teilnehmer B und auch von B an A übertragen.

Space Administration (NASA) deutlich, bei der der Satellit beim Eintritt in die Marsumlaufbahn im August 1993 verloren ging. Die Ursache lag neben Managementfehlern auch in der gleichzeitigen Verwendung von englischen und metrischen Einheiten. So hatten der Hersteller und der Betreiber des Satelliten jeweils unterschiedliche Maßeinheiten verwendet und die notwendige Umrechnung nicht richtig berücksichtigt, was zum Verlust des Systems führte (Oberg Dez. 1999 – Why the Mars probe [accident).

Die hohe Anzahl von Schnittstellen bei der Entwicklung von Raumfahrtsystemen und der großen Bedeutung für eine klare Definition dieser Schnittstellen, verdeutlich wie wichtig es ist, die Experten bei der Kommunikation zu unterstützen. Eine aktuell immer beliebter werdende Lösung in der frühen Entwicklungsphase von neuen Raumfahrtmissionen ist die integrierte, modellbasierte Satellitenentwicklung (IMSE). Bei dieser wird ein spezieller Prozess, ein ausgewähltes Team, eine spezielle Infrastruktur und ein zentrales Modell zusammengefasst (Wilke 2002 – Integrierter modellbasierter Satellitenentwurf). Das zentrale Modell dient bei der IMSE für den Informationsaustausch zwischen den Teammitgliedern.

1.2 Zielsetzung der Arbeit

Der Schwerpunkt dieser Arbeit liegt bei der Analyse derzeitiger Ansätze für dieses zentrale Modell und stellt eine neue Methode für die Erzeugung dieses zentralen Modells vor. Dazu wird eine **neue Modellierungsmethode** vorgestellt, wobei folgende Anforderungen mit berücksichtigt werden:

- **Multi-Disziplinären Modellierung**

  Beteiligung verschiedener Fachdisziplinen.

- **Multi-User Modellierung**

  Beteiligung verschiedener Benutzer, die gleichzeitig mit dem gleichen Modell arbeiten.

- **Gleichzeitiges paralleles Arbeiten**

  Mehrere Entwickler aus den verschiedenen Fachdisziplinen arbeiten gleichzeitig am gleichen Modell.

Dazu wurde eine **fachdisziplin-unabhängige Beschreibung** eines Systems definiert. Zusätzlich wurde eine **objektorientierte Modellierung** von Systemen basierend auf Klassen und Objekten implementiert. Ein weiteres wichtiges Ergebnis dieser Arbeit ist die **Definition** des **Datenmodells** zur Speicherung des Modells. Für die **Verifizierung** wurde ein Raumfahrtsystem mit der neuen Methodik modelliert und analysiert. Zu diesem Zweck wurde diese neue Methodik in einer S**oftware** vom Autor umgesetzt.

1.3 Gliederung der Arbeit

Die Gliederung orientiert sich an der in Abbildung 1-1 skizzierten Systematik und ist in die zwei Hauptbereiche Analyse sowie Umsetzung und Verifikation gegliedert. Die Analyse beginnt in Kapitel 2 mit einer Beschreibung des Umfeldes der IMSE anhand der vier entscheidenden Elemente:

- Prozess,
- Team,
- Infrastruktur sowie
- Werkzeuge und den notwendigen Systemmodellen

Nach der Beschreibung dieser Elemente wird in Kapitel 3 verstärkt auf die aktuellen Werkzeuge und Modelle in der Satellitenentwicklung eingegangen. Dieser Übersicht der aktuellen Werkzeuge folgen die Erfahrungen des Autors im Bereich der integrierten Satellitenentwicklung in Kapitel 4. Nach der Analyse des aktuellen Stands der Wissenschaft (Kapitel 2 bis 4) folgt in Kapitel 5 die Vorstellung der neuen Modellierungsmethode. Die Anwendung der neuen Methodik wird anhand eines Beispiels aus der Raumfahrt in Kapitel 6 gezeigt.

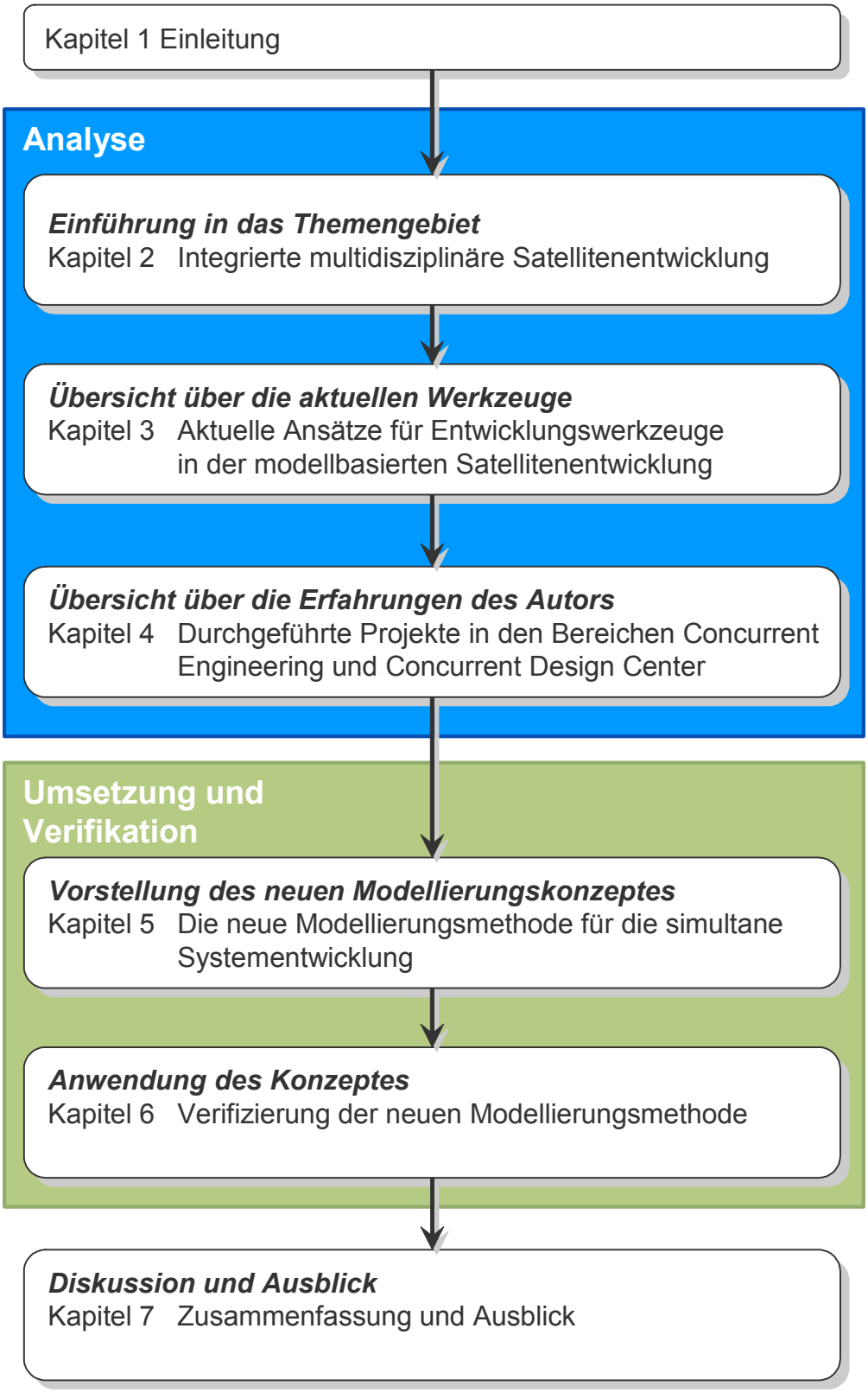


Abbildung 1-1 Gliederung der Arbeit

2 Integrierte multidisziplinäre Satellitenentwicklung

In today's aerospace industry companies have to be more efficient and faster in doing feasibility studies and preparing firm fix price proposals in order to be competitive on the market. But also agencies must work more efficient as the budgets become more and more tighten. Thus one idea was to implement the idea of concurrent engineering in combination with concurrent design environments so called concurrent design centers in agencies and companies. (Aguilar, Dawdy et al. 1998 – The Aerospace Corporation's Concept Design, S. 1)

Im Rahmen dieser Arbeit wird eine Methodik für die Modellierung von Satellitensystemen im Rahmen der IMSE vorgestellt. Für ein besseres Verständnis über den Einsatz und die Anforderungen an diese Methode wird zu Beginn das Umfeld für die Satellitenentwicklung näher beschrieben.

Die Gliederung dieses Kapitels orientiert sich dabei an den charakteristischen Elementen für die IMSE. Diese Elemente werden von verschiedenen Autoren unterschiedlich eingeteilt. So gliedert Wilke (Wilke 2002 – Integrierter modellbasierter Satellitenentwurf, S. 73–98) die IMSE in die fünf wichtigen Elemente Team, Prozess, Infrastruktur, Werkzeuge und Modelle.

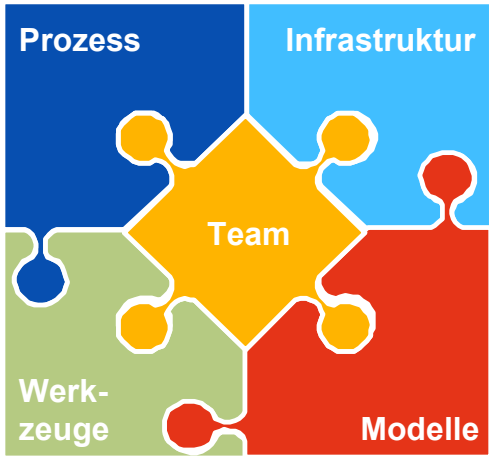


Abbildung 2-1 Die fünf Elemente für die integrierte, modellbasierte Satellitenentwicklung (Wilke 2002 – Integrierter modellbasierter Satellitenentwurf, S. 73)

Mager (Mager, Hartmann 2000 – The Satellite Design Office, S. 294–296) wiederum definiert im Rahmen der Implementierung des Satellite Design Office (SDO) bei EADS-Astrium Friedrichshafen die vier Elemente der IMSE als Team, Prozess, Einrichtung und Werkzeuge.

Bei dem Aufbau der Concurrent Design Facility (CDF) bei der European Space Agency (ESA) wurden nach Bandecchi (Bandecchi, Melton et al. 2000 – The ESA/ESEC Concurrent Design Facility, S. 330) die fünf Elemente der IMSE als Prozess, multidisziplinäres Team, integriertes Design-Modell, Einrichtung und Infrastruktur definiert.

Dieses Kapitel lehnt sich an die Konzeption der genannten Autoren an und ist in die folgenden vier Abschnitte gegliedert:

- **Merkmale des Entwicklungsprozesses**

  Einleitend wird sowohl ein allgemeines Vorgehensmodel, als auch die Beschreibung des von Seiten der ESA vorgegebenen Standardprozesses für die Satellitenentwicklung, dargelegt. Abschließend werden hier die interdisziplinären Aspekte bei der Satellitenentwicklung diskutiert und der Lösungsansatz des Concurrent Engineering (CE) beschrieben.

- **Team**

  Hier werden die für die Entwicklung essentiell wichtigen integrierten und interdisziplinären Entwicklungsteams für die IMSE beschrieben.

- **Infrastruktur**

Anhand von Beispielen aus ESA, NASA und Industrie wird die notwendige Infrastruktur, das sogenannte Design Center, für die IMSE skizziert.

- **Werkzeuge und Modelle**

  Überleitend auf das Kapitel 3, in dem die derzeit vorhanden Werkzeuge im Detail erläutert werden, wird hier die allgemeine Rolle der Werkzeuge und Modelle für die IMSE behandelt.

2.1 Merkmale des Entwicklungsprozesses

Der richtige Entwicklungsprozess ist ein entscheidender Faktor für die erfolgreiche Satellitenentwicklung. Dieser Prozess leitet das Entwicklungsteam über die gesamte Lebensdauer eines Produktes an und ist daher ein Schlüsselfaktor für die erfolgreiche Produktentwicklung. Bevor der spezielle Entwicklungsprozess der ESA beschrieben wird, wird eine allgemeine Prozessstruktur für eine erfolgreiche und schnelle Produktentwicklung außerhalb der Raumfahrt vorgestellt.

2.1.1 Allgemeine Struktur

Einer der führenden Autoren zu dem Thema Produktentwicklung ist Prof. R. G. Cooper. Die von ihm entwickelte Stage-Gate-Methode wird heute schon in mehr als der Hälfte aller US-Unternehmen (u. A. General Motors, IBM, Intel, Proctor & Gamble, DuPont und Nortel Networks) als auch andere internationalen Konzerne wie Siemens, Nokia oder Sony eingesetzt (Cooper 2002 – Top oder Flop, S. 1).

Basierend auf einer retrospektiven Analyse von über 3000 neuen Produkten unter dem Aspekt der Produktentwicklung, fasst Cooper (Cooper 2002 – Top oder Flop, S. 85–124) mehrere Erfolgsfaktoren für eine erfolgreiche Produktentwicklung zusammen. Aufbauend auf diesen Erfolgsfaktoren schlägt Cooper die Stage-Gate-Modell (SGM) für den Entwicklungsprozess neuer Produkte vor. Ein nach der SGM definierter Prozess wird als Stage-Gate-Prozess (SGP) bezeichnet. Das allgemein gehaltene Flussdiagramm eines typischen SGM zeigt Abbildung 2-2.

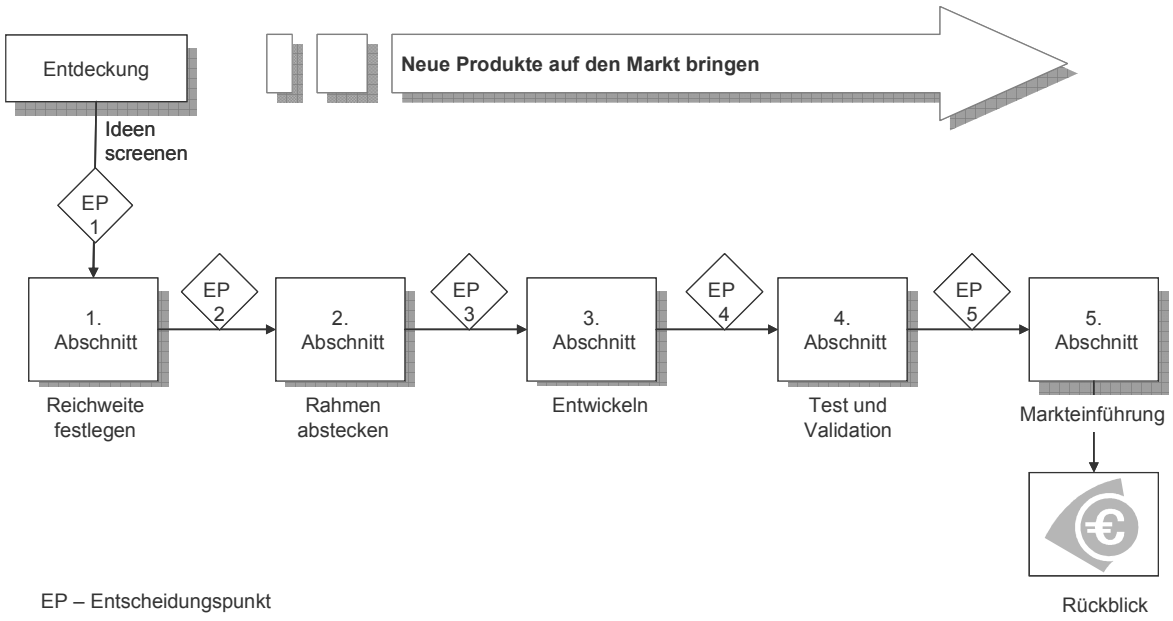


Abbildung 2-2 Das typische Stage-Gate-Modell - von der Entdeckung bis zur Markteinführung bestehend aus fünf Abschnitten und fünf Entscheidungspunkten, zusammen mit einer Entdeckungs- und einer Rückblickphase (Cooper 2002 – Top oder Flop, S. 146)

Die Abschnitte dienen der Unterteilung aller notwendigen Aktivitäten für eine erfolgreiche Produktentwicklung und werden in der Folge zusammen mit der Entwicklungsphase kurz beschrieben:

- **Entdeckung**

  In diesem Abschnitt werden günstige Marktchancen gesucht, Ideen und neue Technologien erforscht. Um eine zielgerichtete Ausrichtung der Forschung in diesem Abschnitt zu schaffen, kann dieser Abschnitt selbst in ein SGP untergliedert werden. Der Entscheidungspunkt (EP) vor dem nächsten Abschnitt dient dazu die Ideen zu sortieren und zu prüfen, welche Idee oder Technologie es wert ist, dass mehr in sie Zeit und Geld investiert wird.

- **Reichweite festlegen**

  Hier wird eine rasche Projektanalyse durchgeführt, um eine mögliche Umsetzung der Idee und das Marktpotential im Vorfeld zu identifizieren. Zu den Tätigkeiten gehört eine vorläufige Beurteilung der Marktattraktivität, des Marktpotenzials, Produktakzeptanz, Wettbewerbssituation und der Erstellung eines

vorläufigen Produktdesigns. Auch der EP am Ende dieser Phase dient einer genauen Prüfung der Projektidee bevor es in die nächste Phase vorrückt.

- **Rahmen abstecken**

  In dieser Phase werden die vorläufigen Studien weiter detailliert und in ein Geschäftskonzept überführt, das unter Anderem einen Markt- und Entwicklungsplan beinhaltet. Die Ergebnisse dieses Abschnittes sind die Grundlage für die Rechtfertigung im EP 3, das Projekt in die Entwicklungsphase überzuführen.

- **Entwicklung**

  Das Design wird im Detail ausgearbeitet, das Produkt entwickelt, spätere Durchführungs- und Herstellungsprozesse werden ausgearbeitet.

- **Test und Validierung / Erprobung und Bestätigung**

  Das Produkt wird in dieser Phase auf Herz und Nieren getestet. Dieses schließt Tests und Erprobungen im Labor, in der Fabrik und auf dem Zielmarkt ein. Zusätzlich werden das Marketing, die Herstellung und die weitere Durchführung des Projektes weiter detailliert.

- **Markteinführung**

  Hier beginnt wird die Serienproduktion, die Preis- und Gewinnspannen sowie das Marketing festgelegt.

Die oben aufgeführten EP sind ein weiterer wichtiger Baustein in dem SGM, da hier anhand der vorzuweisenden Resultate und vorgegebenen Kriterien über die Fortsetzung oder Abbruch des Projektes entschieden wird (Abbildung 2-3).

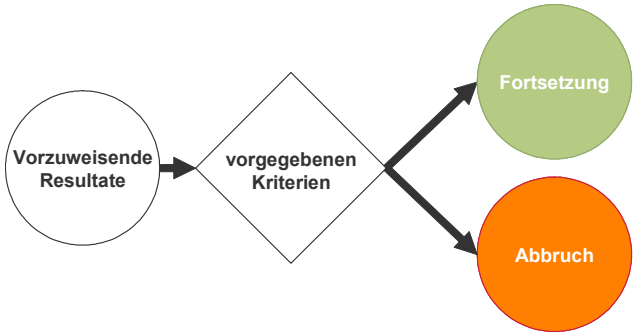


Abbildung 2-3 Gemeinsamer Aufbau der Entscheidungspunkte bestehend aus den Resultaten, Kriterien und den Entscheidungsergebnissen Fortsetzung oder Abbruch (Cooper 2002 – Top oder Flop, S. 148)

Cooper hebt den Geschwindigkeitsvorteil des SGM gegenüber den „phased-review“ Prozessen, wie sie unter anderem die NASA und ESA bis in die neunziger Jahre angewendet hat, hervor (Cooper 2002 – Top oder Flop, S. 162). Der Vorteil soll sich durch die multifunktional angelegten Abschnitte und die parallel durchführbaren Aktivitäten in diesen Abschnitten ergeben. Dazu zählt auch, dass nur ein einziges Team unter einer dedizierten Leitung das Projekt bearbeitet. Das Team enthält Mitglieder aus allen betroffenen Bereichen.

Die ESA sowie die NASA haben eigene Entwicklungsprozesse für ihre Raumfahrtprojekte definiert. Vergleicht man diese Entwicklungsprozesse mit dem SGM, sieht man Übereinstimmungen zwischen den Phasen in der Satellitenentwicklung gleich den Abschnitten im SGM und den von Cooper vorgeschlagenen Entscheidungspunkten, die bei der ESA und NASA als Reviews bezeichnet werden. Daher können die Begriffe Abschnitte und Phasen sowie Entscheidungspunkte und Reviews synonym verwendet werden. Parallel zur gängigen Praxis in der Raumfahrt werden im weiteren Verlauf nur noch die Begriffe Phasen und Reviews benutzt. Es wird nun gezeigt wie so ein „phased-review“ Prozess am Beispiel des ESA-Entwicklungsprozesses aussieht und wie man in der Raumfahrt versucht, diesen Prozess ähnlich zu dem SGM zu beschleunigen.

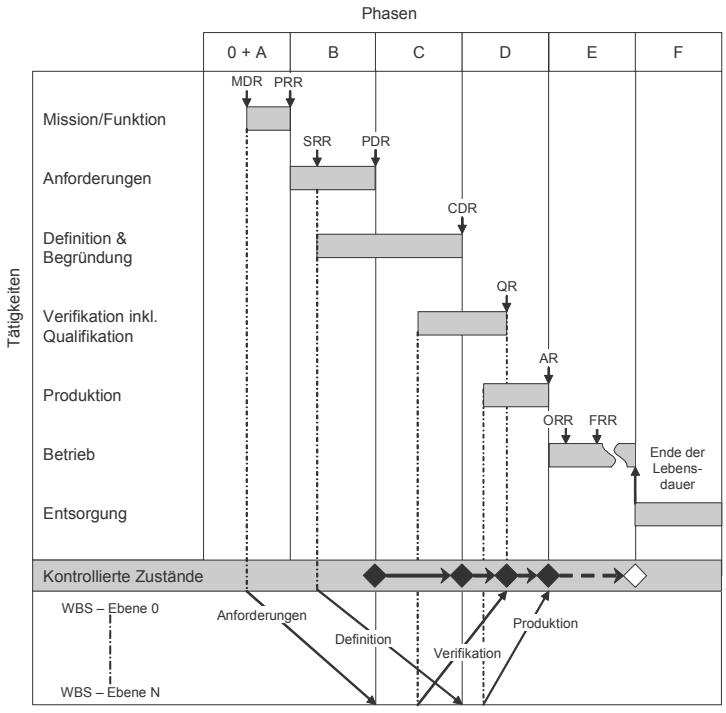
Bevor nun der ESA Prozess im Detail beschrieben wird, sollte hier zur Vollständigkeit noch auf die eigenen Entwicklungsprozesse der NASA, des International Council of Systems Engineering (INCOSE) und das V-Modell XT der Bundesrepublik Deutschland hingewiesen werden. Eine detaillierte Beschreibung dieser Prozesse erscheint im Rahmen dieser Arbeit nicht zweckmäßig, da der Schwerpunkt auf der Modellierung von Satellitensystemen beruht. Eine genaue Beschreibung der Entwicklungsprozesse der NASA findet man im „NASA Systems

Engineering Handbook“ (Shishko 1995 – NASA Systems Engineering Handbook), der INCOSE Prozess ist im „SE Handbook“ (International Council of Systems Engineering (Hg.) – INCOSE) definiert und für das V-Modell XT findet man die Dokumentation unter „http://v-modell.iabg.de“. Nach dieser kurzen Übersicht über die vielfältigen internationalen Entwicklungsprozesse wird nun auf den in Europa für die Entwicklung von Raumfahrtsystemen wichtigen Entwicklungsprozess der ESA näher eingegangen.

2.1.2 Der ESA-Entwicklungsprozess

Das European Cooperation for Space Standardization (ECSS) ist eine Initiative mit dem Ziel, eine kohärente Reihe von benutzerfreundlichen Standards für alle Raumfahrtprojekte in Europa zu definieren. Der ESA Produktentwicklungsprozess ist im ECSS-M-30A „Projektphaseneinteilung und –planung“ beschrieben (ECSS M30-A). Der Standard beschreibt detailliert die notwendigen Vorbedingungen, Aktivitäten und Ergebnisse aller Phasen und Reviews.

Die Projektphasen und Reviews des Entwicklungsprozesses



Reviews

AR	Acceptance Review
CDR	Critical Design Review
FRR	Flight Readiness Review
MDR	Mission Definition Review
ORR	Operational Readiness Review
PDR	Preliminary Design Review
PRR	Preliminary Requirements Review
QR	Qualification Review
SRR	System Requirements Review
WBS	Work Breakdown Structure

Abbildung 2-4 Typischer Lebenszyklus eines Projektes beschrieben durch die Phasen und Reviews nach ECSS Standard M-30A (ECSS M30-A, S. 11)

Die Planung eines Projekts wird bei der ESA in aufeinanderfolgende Phasen gegliedert. Der Beginn einer Phase ist im Allgemeinen abhängig vom erfolgreichen Abschluss eines bestimmten Reviews. Ein Projekt wird üblicherweise in sieben Hauptphasen (Phase 0, A-F) unterteilt. Abbildung 2-1 zeigt diese verschiedenen Phasen mit der wichtigsten Tätigkeit während dieser Phasen. Die Tätigkeiten wiederum sind in der Abbildung mit den Reviews und den kontrollierten Zuständen (Baselines) verknüpft. Die Tätigkeiten während der verschiedenen Phasen lassen sich wie folgt beschreiben:

- **Phase 0: Missionsanalyse/Identifizierung der Erfordernisse**

  In dieser Phase werden die Anforderungen an die Mission und eine erste Missionsanalyse durchgeführt und die geplante Mission charakterisiert. Zusätzlich werden die Anforderungen an die zu erwartende Leistung und Zuverlässigkeit des Systems definiert. Eine weitere wichtige Tätigkeit in dieser Phase ist die Identifikation von möglichen Systemkonzepten unter besonderer Beachtung des Innovationsgrads und aller kritischen Aspekte.
  Diese Phase wird mit einer vorläufige Bewertung von Projektmanagementdaten (Organisation, Kosten, Zeitpläne) und einer entsprechenden Dokumentation der Phase 0 (z. B. die Missionsspezifikation) abgeschlossen. Am Ende der Phase 0 kann ein Missions-Definitions-Review (MDR) stattfinden.

- **Phase A: Durchführbarkeit**

  In der Durchführbarkeitsphase (Phase A) werden die Ergebnisse aus der Phase 0 weiter präzisiert und der Prozess für die Umsetzung dieser Erfordernisse wird eingeleitet. Dadurch werden die in der Phase 0 formulierten Erfordernisse und die daraus resultierenden Lösungsvorschläge vervollständigt. Die Umsetzung erfolgt dabei unter anderem durch

  - die Erstellung eines Funktionsbaums,
  - Quantifizierung und Charakterisierung der kritischen Elemente (hinsichtlich technischer und wirtschaftlicher Eignung) und
  - die Untersuchung der verschiedenen möglichen **Systemkonzepte** durch **Modellierung** und **Vergleich** dieser Konzepte.

Diese Tätigkeiten sind **iterativ auszuführen**, wobei die enge Verknüpfung mit den Tätigkeiten der Phase 0 beachtet werden muss. Die Wiederholung dieser Tätigkeiten (Iteration) kann eine Definitionsänderung der geplanten Mission zur Folge haben.
Die Dokumentation der Phase A beinhaltet alle Projektmanagementdaten. Es kann sich dabei um eine Aktualisierung der Dokumentation und Ergebnisse aus Phase 0 handeln. Am Ende von Phase A wird das vorläufige Anforderungs-Review (PRR) abgehalten, das zu einer Systemkonzeptauswahl und zur Einführung des kontrollierten Basisstandes der zugehörigen Funktionsspezifikation führt.

- **Phase B: Vordefinition (Projekt und Produkt)**

  In dieser Phase wird eine Auswahl der technischen Lösung für das in Phase A gewählte Systemkonzept getroffen. Dadurch kann eine genaue und abgestimmte Definition (Leistungsniveaus, Kosten, Zeitpläne) des Lösungskonzeptes bestimmt werden. Diese Tätigkeiten bedingen ein besseres Verständnis der Durchführbarkeitsfaktoren, die bereits in der vorangegangen Phase erarbeitet wurden. Von diesen Durchführbarkeitsfaktoren werden die Kriterien für das Systemanforderungs-Review (SRR) auf Teilsystemebene, das während der Phase B abgehalten wird, bestimmt.
  Die Dokumentation der Phase B enthält Entscheidungskriterien zur Projektbestätigung, insbesondere die umfassende Bedarfsformulierung in einer Technischen Spezifikationen (TS). Am Ende von Phase B wird das vorläufige Design-Review (PDR) abgehalten, das die in der Technischen Spezifikation (TS) und der Design-Begründungsakte (DJF) beschriebene Entwicklungs-Bezugskonfiguration liefert. Sie führt zu einer vorläufigen Definition der Schnittstellen und dem vorläufigen Schnittstellenkontrolldokumente. Die von den Lieferanten zu liefernden Implementierungsdokumente (einschließlich Entwicklungsplan) werden zu diesem Zeitpunkt vereinbart. Diese Vordefinitionsphase hat somit die Auswahl, Begründung und Bestätigung der für die Entwicklung gewählten Lösung zum Ziel.

- **Phase C: Detaildefinition (Produkt)**

In dieser Phase wird die Lösung aus der vorangegangenen Phase weiter untersucht und detailliert, so dass das Produkt in allen Einzelheiten definiert ist und bereits erste Bauteile für Tests hergestellt werden können. Die Aufbau-, Test- und Qualifikationsbedingungen und Initialisierung der Produktions- und Verifikationsmethoden und –mittel werden freigegeben. Sollte noch nicht in der früheren Phase mit der Beschaffung der notwendigen Komponenten begonnen worden sein, so wird dies in dieser Phase gestartet. Falls erforderlich wird dabei eine endgültige Entscheidung über "Eigenfertigung oder Kauf" der Komponenten getroffen.
Es wird die Produktions-Stammakte auf den endgültigen Stand aktualisiert, damit die ersten Modelle des Systems hergestellt werden können. Dabei müssen die Schnittstellen in der Entwicklungsbezugskonfiguration festgelegt und die entsprechenden Schnittstellenkontrolldokumente (ICDs) in die Konfigurationsüberwachung aufgenommen werden. Am Ende dieser Phase wird das kritische Design-Review (CDR) zur Abnahme der Produktions-Stammakte abgehalten.

- **Phase D: Produktion/Bodenqualifikationstest**

  Die Phase D stellt das Ende der Systementwicklung dar und die Produktions-Stammakte und Betriebsdokumentation werden freigegeben. Dabei werden zuerst die Qualifikationsmodelle für das Produkt gefertigt, integriert und abschließend getestet. Dieses Tests dienen zur Bestätigung und Qualifikation von Methoden, Verfahren sowie Produktions- und Verifikationsmitteln für Herstellungs-, Montage-, Integrations- und Verifikationsaufgaben. Die Ergebnisse bilden die Entscheidungsgrundlage für das Qualifikation-Reviews (QR) in dem die Herstellung des Systems freigegeben wird. Die Bodenqualifikationstests sind somit ein Weg zur Verifizierung der technischen Konformität der Komponenten gegenüber den Anforderungen (Designqualifikation) sowie deren Gebrauchstauglichkeit für den Betrieb (Betriebsqualifikation) mit Identifizierung der Funktions- und Betriebsreserven.
  Der zweite wichtige Teil in dieser Phase ist die Produktion des Systems. Sollte es sich um eine Serienfertigung handeln, so wird in dieser Phase nur ein Proto-

typ gebaut und die Serienproduktion findet in der folgenden Phase E statt. Diese Phase endet mit dem Abnahme-Review (AR) und ermöglicht neben den oben genannten Tätigkeiten eine Vorbereitung der Betriebsphase und der eventuellen Serienfertigung. Der Abnahmeprozess beinhaltet einen vereinbarten technischen Prozess mit Kontrollen und Tests, in dessen Verlauf der Kunde die vollständige Übereinstimmung des gelieferten Produkts mit der jeweils gültigen Bezugskonfiguration bestätigt. Am Ende dieses Prozesses muss der Kunde über die Annahme der Lieferung entscheiden.
Aufgrund des integrierten Charakters ihrer Tätigkeiten sind die Phasen C und D normalerweise nicht voneinander trennbar.

- **Phase E: Betrieb**

  Diese Phase umfasst die Startkampagne, bestehend aus Start- und Flugabnahme (ORR und FRR), dem Start, dem Betrieb und der Instandhaltung von Raumfahrtsystemen. Sowohl der Operational Readiness Review (ORR) als auch der Flight Readiness Review (FRR) werden vor dem Start von Satelliten durchgeführt, um die Einsatzbereitschaft zu gewährleisten. Nach dem Start findet noch eine sogenannte Check-Out Phase oder Commissioning Phase zur Überprüfung der Funktionsbereitschaft im Orbit statt.

- **Phase F: Entsorgung**

  Die Entsorgungsphase umfasst alle Vorgänge vom Ende der Lebensdauer bis zur endgültigen Entsorgung des Produkts. Diese Phase wird in der Regel während der Betriebsphase des Systems vorbereitet. Sie ist für das jeweilige Produkt spezifisch und wird vom Betreiber initiiert. Die Entsorgung kann eine Bewertung der Umweltauswirkungen erforderlich machen.

Die beiden ersten beschrieben Phasen 0 und A werden auch als Konzeptphase bezeichnet. Diese Konzeptphase ist **stark iterativ**[4] und es soll nach dem ESA Standard **modellbasiert gearbeitet** werden. Daher können die Durchlaufzeiten dieser Phasen durch paralleles abarbeiten der Tätigkeiten ähnlich dem SGM deutlich reduziert werden. Um diese integrative, paral-

[4] Bei der Iteration werden die gleichen Tätigkeiten mehrmals wiederholt bis ein gewünschtes Ergebnis erreicht wird.

lele, modellbasierte Systementwickelung zu unterstützten, wurde im Rahmen dieser Arbeit eine neue Modellierungsmethode entwickelt. Für ein besseres Verständnis der Konzeptphase wird diese an Hand ihrer Charakteristika genauer beschrieben.

Charakteristika der Konzeptphase in der Satellitenentwicklung

Wilke (Wilke 2002 – Integrierter modellbasierter Satellitenentwurf, S. 39–41) definiert für die Konzeptphase folgende sechs wichtige Charakteristika:

1. **Bedeutung der Konzeptphase**

   In der Konzeptphase von neuen Raumfahrtsystemen werden ca. 75% der später anfallenden Systemkosten und ungefähr 80% der funktionalen Produkteigenschaften festgelegt (Ehrlenspiel, Kiewert et al. 2003 – Kostengünstig entwickeln und konstruieren, S. 13). Da in dieser Phase das System noch nicht im Detail definiert ist, können Änderungen schnell und günstig am Produktkonzept vorgenommen werden. Werden Änderungen erst in einer späteren Projektphase vorgenommen, multiplizieren sich die Kosten für diese Änderung pro Phase um den Faktor 10 (Rule of Ten, vgl. Fricke 1998 – Der Änderungsprozess als Grundlage, S. 113). Dennoch wird in der Regel dieser Produktentwicklungsphase wesentlich weniger Aufmerksamkeit geschenkt als späteren Phasen. Andersson (Andersson 1994 – Early Design Phases and Their) kommt in einer Untersuchung von unterschiedlichen Unternehmen zu dem Schluss, dass der zeitliche Aufwand für die Konzeptphase bei 40% der untersuchten Unternehmen weniger als 10% und bei weiteren 40% der Unternehmen nur zwischen 10% und 20% der gesamten Entwicklungsdauer liegt. Während dieser Phase werden bei 70% der Unternehmen nur ein bis zwei Konzepte auf Systemebene untersucht, wodurch eine breite Untersuchung des Lösungsraumes somit nicht stattfindet. Durch eine modellbasierte Entwicklung könnten die Anzahl der untersuchten Konzepte deutlich erhöht werden.

2. **Interdisziplinarität**

   Da die Produkte immer vielfältigere Eigenschaften besitzen, sind in den Entwicklungsteams für diese Produkte die verschiedenstes Fachdisziplinen vertre-

ten. Daher ist diese Konzeptphase stark interdisziplinär. Zur erfolgreichen Planung eines qualitativ hochwertigen Produkts müssen von Anfang an Vertreter aller relevanten Fachbereiche, evtl. auch über Unternehmensgrenzen hinweg, zusammenarbeiten.

3. **Hoher Vernetzungsgrad und hoher Kommunikationsbedarf**

   Da bei der Planung und Definition eines Produktes alle benötigten Spezialdisziplinen eingebunden werden müssen, bedarf es eines hohen Vernetzungsgrades zwischen den Disziplinen. Diese Vernetzung muss mit einem starken Informationsaustausch zwischen den Experten einhergehen. Der Kommunikationsbedarf wird neben den interdisziplinären Charakter durch die Kommunikation von unscharfen oder sich ändernden Anforderungen und notwendigen Iterationsschleifen gesteigert.

4. **Iterativität**

   Wie in der Beschreibung der Phase A und D gezeigt, ist die Produktentwicklung kein linear lösbarer Prozess. Daher sind mehrere Iterationsschleifen während der Entwicklung eines Lösungskonzeptes notwendig. Sollten zu dem mehrere verschiedene Lösungskonzepte berücksichtigt werden, steigert dies die Anzahl der notwendigen Iterationen.

5. **Hohe Dynamik im Systemdesign**

   Da das Produkt in der frühen Phase nur als Konzept existiert und die meisten Designentscheidungen auf Annahmen beruht, ändert sich das Design eines Systems auf Grund von Änderungen der Annahmen extrem schnell. Auch stellen die vom Kunden oft geänderten Randbedingungen einen Motor der Systemdynamik dar.

6. **Beschränkte Genauigkeitsanforderungen**

   Im Gegensatz zum Detaildesign und zur Verifikation eines Konzepts, wie es in der Phase B, C und D verlangt wird, sind in der Konzeptphase nicht das dynamische Verhalten und die exakte Auslegung eines Konzepts von Bedeutung. Vielmehr steht die ganzheitliche Betrachtung des Problemfeldes, die Identifi-

kation unterschiedlicher Lösungskonzepte und deren Analyse auf Budgetlevel im Vordergrund. Daher ist die Anforderung an die Genauigkeit der Aussagen in dieser Phase geringer als in späteren Phasen. Die dadurch ergebenden Unsicherheiten werden durch Risikoabschätzungen abgedeckt.

In der Konzeptphase sind zwei Charakteristika besonders hervorzuheben. Zum Einen ist dies die Interdisziplinarität und zum Anderen die daraus resultierenden Iterationen in der Satellitenentwicklung. Bei diesen Iterationen werden Tätigkeiten mehrfach innerhalb einer Phase durchgeführt. Dies ist Notwendig, um zu einem Gesamtsystem-optimalen und nicht einer Teilsystem-optimalen Lösung zu kommen. Diese Iterationen sind aus der Phasen Darstellung in Abbildung 2-4 (Seite 12) nicht ersichtlich. Der folgende Abschnitt befasst sich daher mit diesen Iterationen und der Interdisziplinarität in der Konzeptphase.

2.1.3 Iterationen und Interdisziplinarität in der Konzeptphase

Obwohl der Entwicklungsprozess der ESA ein sequentielles Vorgehen widerspiegelt, so sind doch die Aktivitäten innerhalb einer Phase stark iterativ. Dies ist insbesondere in der frühen Phase eines Satelliten (Phase 0 und Phase A) der Fall, in der die verschiedensten Lösungsmöglichkeiten identifiziert und analysiert werden müssen.

Diese iterativen Abhängigkeiten wurden von dem Autor während der Betreuung mehrer Workshops in den Jahren 2001-2004 am Fachgebiet für Raumfahrttechnik (LRT) selbst beobachtet. Während dieser einwöchigen „Concept Design Center“–Workshops haben zehn Studenten eine Satellitenmission, vergleichbar einer Phase 0 Studie, ausgelegt. Dabei hat jeder Student eine Fachdisziplin (Teilsystem) des Satelliten bzw. der Mission übernommen. Eine detaillierte Beschreibung des Prozesses, Vorgehensmodells und der Ergebnisse der Workshops ist in Abschnitt 4.4 zu finden. Hier wird nur auf die enge Verzahnung der einzelnen Fachdisziplinen untereinander eingegangen.

Da das Modell zur Auslegung des Satelliten, das den Studenten während des Workshops zur Verfügung stand, nicht vollständig mit allen Berechnungen abgebildet werden kann, wird an dieser Stelle nur eine Matrix Darstellung der Abhängigkeiten zwischen den Fachdisziplinen gezeigt. Die in Abbildung 2-5 dargestellten Abhängigkeiten zeigen die Ergebnisse liefernden

Elemente (Outputs) in Zeilen und die empfangenden Elemente (Inputs) in Spalten. Eine „1“ kennzeichnet eine Abhängigkeit des Spaltenelementes von dem Reihenelement.

Output Elemente \ Input Elemente	Lage- und Orbitregelung	Kommunikation	Konstanten	Kosten	Dokumentation	Mission und Betrieb	Nutzlast	Energieversorgung	Antrieb	Anforderungen	Struktur und Konfiguration	Test	Thermalsystem
Lage- und Orbitregelung				1		1		1			1		1
Kommunikation				1		1		1			1		1
Konstanten	1					1	1		1				1
Kosten													
Dokumentation													
Mission und Betrieb	1	1		1			1	1	1		1		
Nutzlast		1		1				1			1		1
Energieversorgung	1			1							1		1
Antrieb	1			1				1			1		1
Anforderungen	1			1		1	1	1					
Struktur und Konfiguration	1			1		1		1	1				1
Test				1									
Thermalsystem	1			1							1		

Abbildung 2-5 Abhängigkeitsmatrix des Satellitenmodells bei einem der „Concept Design Center“–Workshops

Während dieser Workshops haben sich ausgeprägte Iterationen zwischen den Fachdisziplinen für das Antriebssystem, Lageregelungssystem und der Struktur besonders aufgezeigt. Wie aus Abbildung 2-5 ersichtlich wird, haben einzelnen Fachdisziplinen eine hohe Anzahl von Schnittstellen und somit eine hohe Abhängigkeit von anderen Disziplinen. Diese starke Verknüpfung der verschiedenen Fachdisziplinen macht die **Notwendigkeit** einer **modellbasierten Satellitenentwicklung** deutlich. Denn bei jeder Nahtstelle zwischen zwei verschiedenen Fachdisziplinen gibt es Informationsverluste zum Beispiel auf Grund von mangelhaftem Informationstransfers oder durch ein unterschiedliches Verständnis der Informationen zwischen Sender und Empfänger. Der modellbasierte Ansatz verringert diese Informationsverluste, da alle relevanten Informationen in einem einheitlichen fachdisziplin-unabhängigen Format abgespeichert werden.

Modelle dienen aber nur zur Beschreibung und Auslegungsrechnung von Systemen und können daher die Entwickler nur unterstützen. Die Entwickler müssen somit parallel und sequen-

tiell mit diesen Modellen arbeiten. Um die starken Abhängigkeiten in der Systemauslegung besser koordinieren und die damit verbundenen iterativen Auslegungsschritte schneller abarbeiten zu können, wurde als Lösungsansatz das Concurrent Engineering (CE) in der Raumfahrt eingeführt (Stagne 2003 – The Integrated Concurrent Enterprise). Die Idee hinter CE wird im folgenden Abschnitt genauer beschrieben.

2.1.4 Concurrent Engineering

Ende der 80er, Anfang der 90er Jahre interessierten sich Firmen und Organisationen aus der Raumfahrt vermehrt für das CE. Das CE ist charakterisiert durch physisch zusammengeführte Entwicklungsteams bestehend aus den Experten, die an Produktentwicklungs-, Produktentstehungs- und Produktnutzungsprozessen beteiligt sind. Diese Experten arbeiten gemeinsam an den Anforderungen, Produkt- und Prozesskonzepten, sowie Produkt- und Prozessplänen (Grady 1993 – System requirements analysis). Trotz dieser recht guten Beschreibung für das CE kann man keine einheitliche Definition in der Literatur finden. Fricke bietet eine gute Übersicht über die Definition von CE (Fricke 1998 – Der Änderungsprozess als Grundlage, S. 14–15):

- Winner (Winner 1988 – The role of concurrent engineering) definieren CE als einen systematischen Ansatz für ein integriertes und gleichzeitiges Design von Produkten und ihren zugehörigen Prozessen, einschließlich Fertigung und Vertrieb. Dieser Ansatz fördert seiner Meinung nach die Berücksichtigung aller Elemente des Produktlebenszyklus von der Konzeption bis zur Außerdienststellung und berücksichtigt Qualität, Kosten, Zeitplan und Nutzeranforderungen bereits in der Angebotsphase.
- Dean (dean92elements) versteht unter CE, dass die richtigen Personen zur richtigen Zeit zusammen kommen, um Designprobleme zu lösen. Dabei werden oft Computer Aided Engineering (CAE) Tools benutzt, um Entwicklung und Fertigung innerhalb der CE Umgebung zu unterstützen. Diese Tools basieren auf einer integrierten Datenhaltung der notwendigen Daten wie zum Beispiel Systemanforderungen, Designinformationen oder Zeichnungen, um die Entwicklung der Produkte zu unterstützen.

- CE wird oft als Simultaneous Engineering (Krottmaier 1995 – Leitfaden Simultaneous Engineering), Concurrent Design oder Integrierte Prozess- und Produktentwicklung (Integrated product and process development – IPPD) bezeichnet (IEEE-SA Standard Board (Hg.) 8. Dezember 1998 – IEEE Standard for Application). Grady (Grady 1993 – System requirements analysis) nennt es „eine Wiedergeburt des Systems Engineering Ansatzes". Als Grundlage des CE Gedankens können die Ausführungen von Hall (Hall 1962 – A Methodology for Systems Engineering) betrachtet werden, bei dem es letztendlich ein Teil des Systems Engineering ist.

Das bekannteste Beispiel für die Implementierung von CE in der Raumfahrt ist der Integrated Concurrent Design (ICD) Prozess des Jet Propulsion Laboratory (JPL) der NASA. Das JPL hat als Reaktion auf die NASA Direktive „faster, better, cheaper" des NASA Headquaters aus dem Jahre 1990 den ICD Ansatz zusammen mit „The Aerospace Corporation" entwickelt (McInnes A. 2003 – Integrated Concurrent Design). Der ICD – Prozess unterstützt das Integrated Concurrent Engineering (ICE), das derzeit in der Raumfahrtsystementwicklung beim JPL zum Einsatz kommt.

Zusammenfassend werden beim CE die einzelnen Tätigkeiten bei einer Systementwicklung von den zeitlichen Phasen entkoppelt. Somit können Tätigkeiten parallelisiert und so früh wie möglich gestartet werden, um ein möglichst frühes Feedback von den einzelnen Tätigkeiten zu erhalten. Somit findet eine frühzeitige Verknüpfung und Einbindung aller am Projekt Beteiligten statt. Diese Parallelisierung von Tätigkeiten fordert bereits Cooper bei seinem SGM. Somit konnte gezeigt werden, wie die NASA ihren alten „phased-review" Prozess verbessert hat. Die ESA hat diese Idee auch aufgegriffen und implementiert (Bandecchi, Melton et al. 2000 – The ESA/ESEC Concurrent Design Facility).

Dieses Kapitel zeigte eine Übersicht über allgemeine Entwicklungsprozesse und den „phased-review" Prozess der ESA. Dabei wurde die Bedeutung der Konzeptphase sowie der Interdisziplinarität und Iterationen und der modellunterstützten Entwicklung hervorgehoben. Zusätzlich wurde CE als ein Lösungsansatz für die Beschleunigung der Systementwicklung vorgestellt. Für die Umsetzung dieses Lösungsansatzes wird ein Entwicklungsteam benötigt. Auf dieses Entwicklungsteam wird in dem nun folgenden Abschnitt näher eingegangen.

2.2 Teams

Das Entwicklungsteam stellt bei jeder Satellitenentwicklung den zweiten wichtigen Erfolgsfaktor dar. Ohne das Wissen und die Fähigkeiten der Experten in dem Entwicklungsteam kann trotz eines perfekten Prozesses, den Werkzeugen und Simulationstools kein komplexes System entwickelt werden. Zwar ist in den Prozessen, Werkzeugen und Simulationen bereits Fachwissen hinterlegt, dieses Fachwissen muss aber der jeweilige Entwickler nutzen können. Daher müssen Entwicklungsteams aus Experten aller beteiligten Fachdisziplinen zusammengestellt werden.

Für die Entwicklung von komplexen Systemen wie Satelliten, die sich durch eine starke Verknüpfung verschiedenster Disziplinen auszeichnen, werden in den letzten Jahren verstärkt integrierte Entwicklungsteams eingesetzt. Das Verteidigungsministerium der Vereinigten Staaten von Amerika bezeichnet diese Teams als Integrated Product Teams (IPT). IPTs bestehen aus Vertretern aller betroffenen Fachdisziplinen die gemeinsam erfolgreiche Programme erstellen, Probleme erkennen und lösen, sowie rechtzeitig die richtigen Vorschläge für Entscheidungen unterbreiten (Department of Defence (Hg.) 23.02.2001 – DoD 5000).

Eine allgemeine Übersicht über Engineering Teams bietet Ferris (Ferris 2003 – Review of Papers Concerning Engineering). Er hat dazu alle Veröffentlichungen auf den jährlichen Symposien des International Council of Systems Engineering zwischen 1991 und 2002 analysiert. Er verweist unter Anderem auf eine Veröffentlichung aus dem Jahre 1992 von Graves (Graves, Morgan et al. 1992 – Project Planing and proposal development), in der er hervorhebt, dass **Entwicklungsteams von** den **Vorgesetzten** mit **klaren Zielen** und der **notwendigen Entscheidungsbefugnis ausgestattet** werden müssen. Diese Forderungen sind die Grundlage für das CE wie es derzeit von ESA, NASA und einigen Firmen propagiert wird.

Obwohl das Entwicklungsteam eine bedeutende Rolle im Rahmen der IMSE spielt, wird im Rahmen dieser Arbeit auf die wichtigsten allgemein gültigen Aspekte bei der Teamzusammensetzung eingegangen. Neben diesen Aspekten spielt die spezielle Aufgabenstellungen und die menschliche Individualität bei der Teamzusammenstellung eine wichtige Rolle.

- **Moderator**

  Die eigenen Erfahrungen haben gezeigt wie wichtig es ist, einen Moderator im Team zu haben. Ein Moderator muss bestimmte Qualifikationen besitzen, um das Team zielorientiert durch die Teamsitzungen zu führen. Dieser Moderator kann daher entweder der Projektleiter oder ein anderes Teammitglied sein. In manchen Fällen ist der Moderator keiner Fachdisziplin zugeordnet.

- **Kunde**

  Der Kunde, sei es ein firmeninterner oder externer Kunde, sollte so weit wie möglich bei den Teamsitzungen anwesend sein. Dies ist besonders in der Anfangsphase der Systementwicklung wichtig, wenn das Team die Anforderungen und Wünsche des Kunden aufnimmt. In dieser Phase bildet sich bei dem Team ein Verständnis der Problemstellung. Im späteren Verlauf können Designentscheidungen schneller im Einverständnis mit dem Kunden getroffen werden, wenn dieser bei den Teamsitzungen anwesend ist.

- **Modell**

  Das Modell ist kein offensichtliches Merkmal eines Entwicklungsteams, trotzdem ist es ein sehr wichtiges Merkmal. Jeder Mensch abstrahiert eine Problemstellung und bildet somit intuitiv ein Modell. Diese Modelle sind von Mensch zu Mensch unterschiedlich, da ein Teammitglied zum Beispiel für die Kosten, ein anderer für das Risiko und ein dritter für die technische Umsetzung verantwortlich ist. Alle diese verschiedenen Teilmodelle, sei es in einem Softwarewerkzeug oder nur im Kopf des Entwicklers, müssen bei der integrierten Entwicklung aufeinander abgestimmt werden. Diese Abstimmung der Teilmodelle erfolgt bei der IMSE durch die Verwendung eines zentralen Systemmodells. **Die Entwicklung eines solchen fachdisziplinunabhängigen Modells ist die Zielsetzung dieser Arbeit**.

Im Rahmen dieser Arbeit wurde auch eine Studie über den Einsatz von Concurrent Engineering (Schiffner, Kuss 2006 – Results of an international survey) durchgeführt. Die Ergebnisse zeigen eine Teamgröße zwischen sechs und 20 Mitgliedern bei 80% der befragten Unternehmen (vgl. Abbildung 2-6). Dabei haben alle Teams einen dedizierten Moderator. Ein weiteres

wichtiges Ergebnis aus dieser Studie ist die Schulung des Entwicklungsteams für den Entwicklungsprozess und die Werkzeuge.

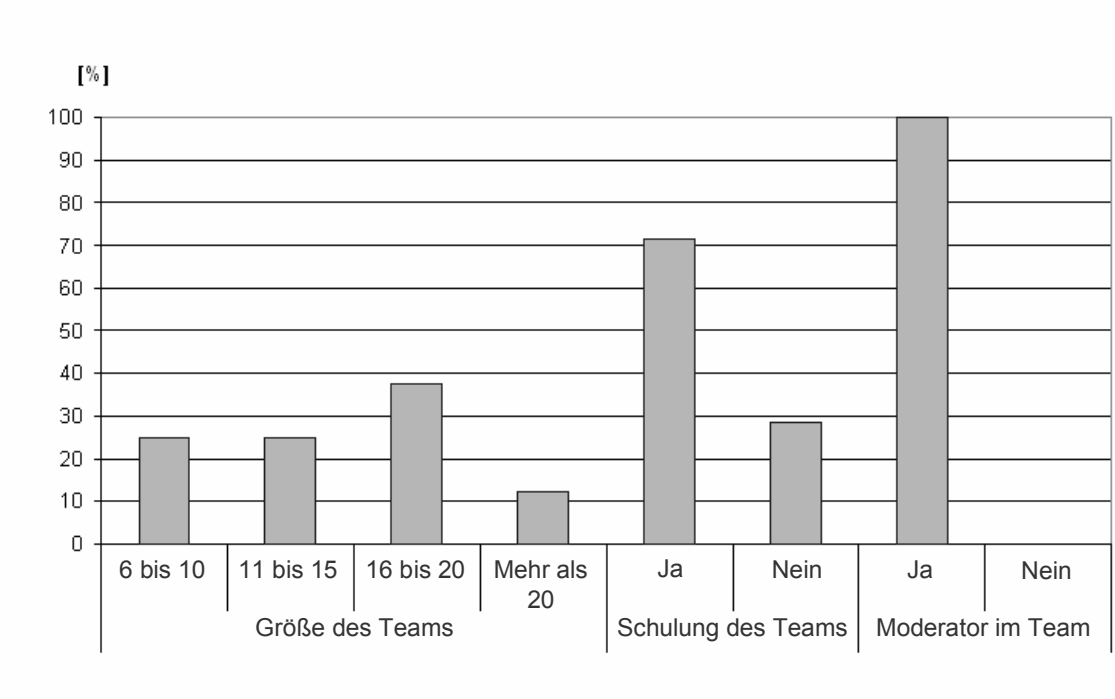


Abbildung 2-6 Ergebnisse (in Prozent) einer Umfrage über die Teamgröße und -schulung bei CE in der Raumfahrt (Schiffner, Kuss 2006 – Results of an international survey)

Da die Entwicklungsteams normalerweise eine Matrixabbildung der Fachdisziplinen oder Organisationseinheiten einer Firma darstellen (Abbildung 2-7), benötigen sie eine entsprechende Infrastruktur in der sie nach dem CE Ansatzes arbeiten können. Dafür werden in den Firmen sogenannte Design Center (DC) bzw. Concurrent Design Center (CDC) eingerichtet. Im Rahmen dieser Arbeit wird der Begriff CDC verwendet, um die Vereinigung des DC mit dem CE mehr hervorzuheben. Der nun folgende Abschnitt erklärt CDC anhand von mehrerer Implementierungen.

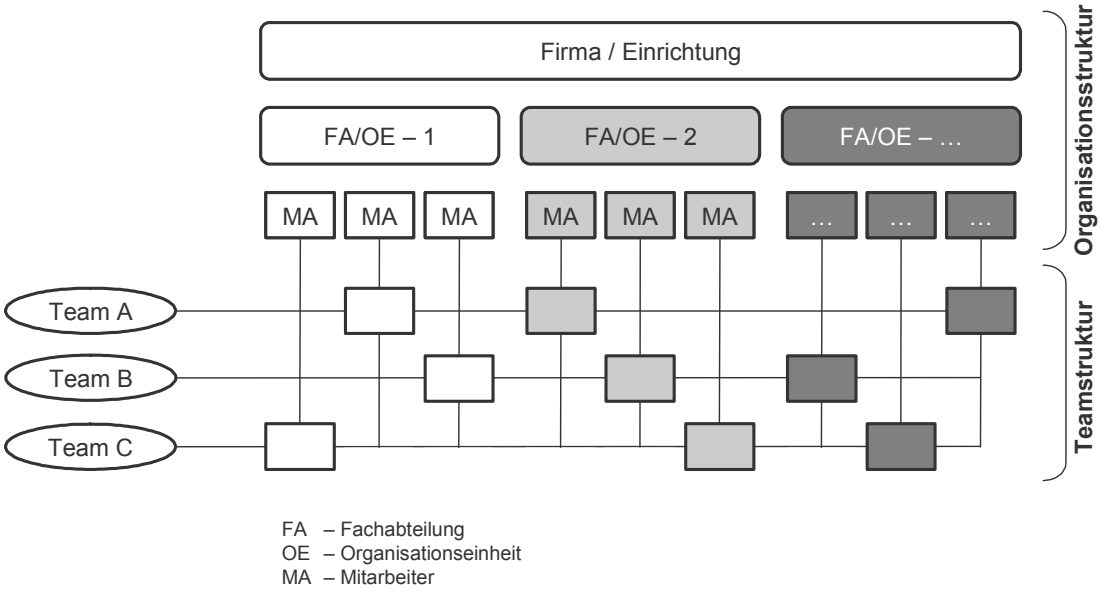


Abbildung 2-7 Matrixorganisation bei den Entwicklungsteams

2.3 Concurrent Design Center

Concurrent Design Center sind dedizierte Räume, die durch ihre Ausstattung eine besondere Infrastruktur für das CE zur Verfügung stellen. Diese Räume werden meistens speziell für das CE eingerichtet. Die Infrastruktur bietet für einen schnellen Zugang zu den benötigten Informationen jedem Teammitglied einen eigenen Arbeitsplatz. Auf diesen Arbeitsplätzen sind alle notwendigen Werkzeuge, die das Team benötigt installiert. Daneben haben die CDC ein oder mehrere Projektionsflächen, über die die einzelnen Teammitglieder wichtige Informationen allen anderen Beteiligten gleichzeitig präsentieren können. Fast alle CDC besitzen ein Videokonferenzsystem um während einer Teamsitzung andere Entwickler, den Kunden oder Projektpartner einbinden zu können. Eine genaue Beschreibung wie ein CDC auszusehen hat, ist nicht möglich, da die Ausstattung eines CDC von dem Prozess, dem Team, dem Produkt und den finanziellen Mitteln der Firma oder Einrichtung abhängen. Sollte eine Firma mehrere CDC haben, so können diese auch je nach Standort unterschiedlich ausgestattet sein.

Die eigene Studie (Schiffner, Kuss 2006 – Results of an international survey) hat gezeigt, dass 70% der CDC Implementierungen dem Design Team neben dem zentralen Raum noch Möglichkeiten für Nebenbesprechungen zur Verfügung stellen. Der Raum enthält fast immer eine

zentrale Projektionsfläche für einen Projektor. Die meisten (ca. 88%) Implementierungen haben die Infrastruktur mit einem Videokonferenzsystem erweitert (vgl. Abbildung 2-8).

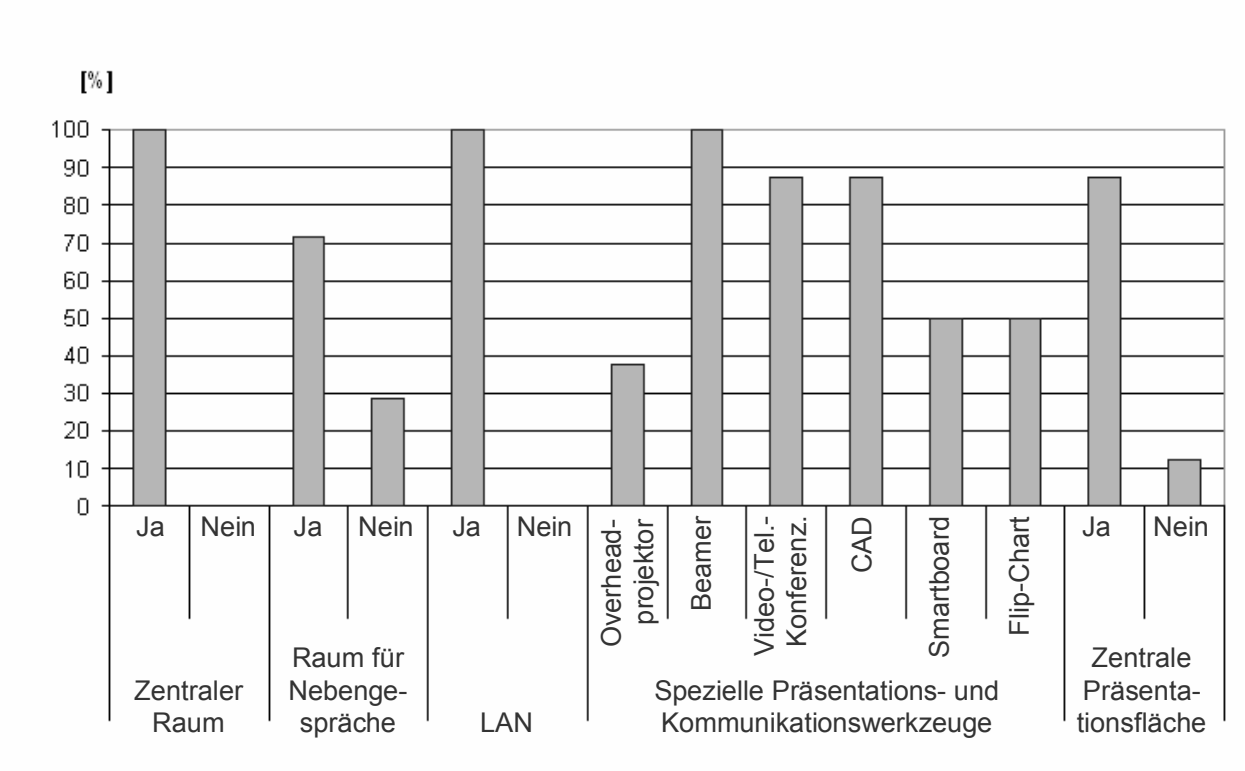


Abbildung 2-8 Ergebnisse (in Prozent) einer Umfrage über die Infrastruktur eines CDC in der Raumfahrt (Schiffner, Kuss 2006 – Results of an international survey)

Eine kleine Übersicht über die Verbreitung des CDC in der Raumfahrt soll an dieser Stelle die Geschichte in der Raumfahrt aufzeigen.

NASA

Das Jet Propulsion Laboratory (JPL) der NASA hat zusammen mit der „The Aerospace Corporation“ das Project Design Center (PDC) mit dem entsprechenden Prozess (ICD) und Werkzeugen entwickelt und eingerichtet (Aguilar, Dawdy et al. 1998 – The Aerospace Corporation’s Concept Design). Das PDC war das erste CDC in der Raumfahrt. Das PDC besteht aus zwei Räumen, einem großer Raum in dem das Team für das Missionsdesign arbeitet und einem kleineren Raum, in dem das Team für die Nutzlasten arbeitet (Abbildung 2-9).

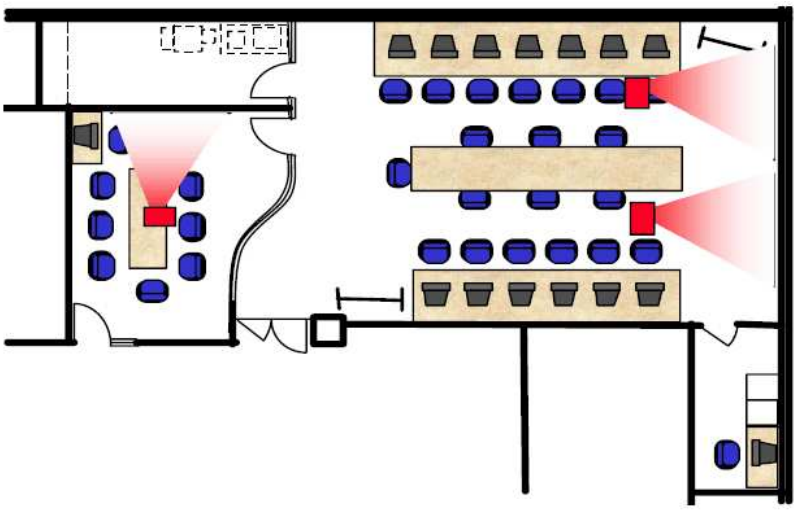


Abbildung 2-9 Raumaufteilung des Hauptraumes (großer Raum) und Nebenraumes (kleiner Raum) des Project Design Centers am Jet Propulsion Laboratory der NASA (McInnes A. 2003 – Integrated Concurrent Design, S. 14)

Die NASA hat sich für eine Raumaufteilung mit einem zentralen Mitteltisch und den Arbeitstationen zur Wandseite hin entschieden. Zwei Projektoren dienen zur Anzeige der relevanten Informationen für das gesamte Entwicklungsteam.

ESA

Die ESA hat die Idee CE in Verbindung mit CDC für die Phase 0 / Phase A-Studien in ihrem Concurrent Design Facility (CDF) umgesetzt, das im European Space Research and Technology Centre (ESTEC) in Noordwijk steht. Das CDF wird dabei nicht nur für Studien eingesetzt sondern auch für Reviews von Industriestudien. Im Frühjahr dieses Jahres (2007) wurde im CDF die 70te Studie bzw. Review durchgeführt.

Die Raumaufteilung der ESA unterscheidet sich deutlich von der Aufteilung wie sie die NASA in ihrem PDC hat. Die ESA hat die Tische in einer U-Form um eine zentrale Multimediawand bestehend aus drei Rückprojektionsflächen angeordnet (Abbildung 2-10). Dadurch haben alle Teilnehmer die Möglichkeit gleichzeitig auf die Multimediawand und auf ihren eigenen Arbeitsplatz zu schauen.

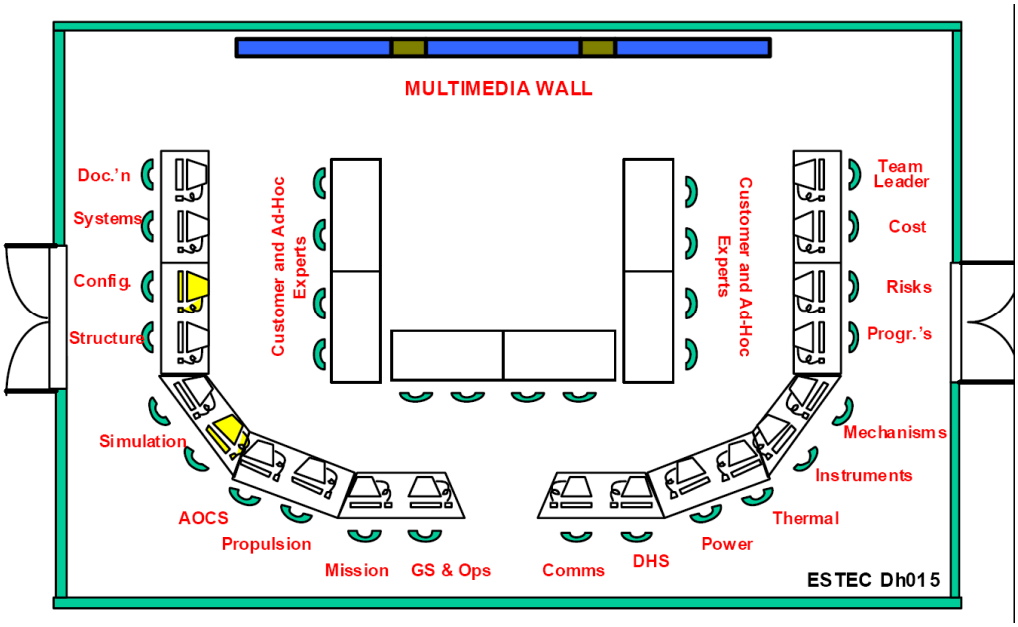


Abbildung 2-10 Raumaufteilung der Concurrent Design Facility der ESA in ESTEC (Bandecchi 2001 – The ESA Concurrent Design Facility @ Finnland, S. 18)

Industrie

Neben den beiden bekanntesten CDCs in der Raumfahrt, das PDC der NASA und das CDF der ESA, gibt es bereits mehrere CDC in der Industrie. So hat EADS-Astrium mit dem SDO das erste CDC in Europa eingerichtet (Mager, Hartmann 2000 – The Satellite Design Office). Dabei wurden beide Raumaufteilungen sowohl der ESA als auch der NASA verwendet. Inzwischen findet man in ganz Europa verteilt die verschiedensten Implementierungen des CE mit CDC (European Space Agency 2006 – Proceedings of the 2nd Concurrent). Daneben wurde das Customer Demand Center (CDC) bei Fairchild Dornier Company (Finkel S. 2002 – Design Centers –Transferring Experiences) in der Luftfahrtbranche implementiert. Bei diesem wurde die Raumaufteilung der NASA wiederverwendet.

Universitäten

Inzwischen haben auch mehrere Universitäten für die Ausbildung ihrer Studenten und für Forschungszwecke ein CDC eingerichtet. So findet man in Amerika bei dem California Institute of Technology das „Laboratory for Spacecraft and Mission Design (LSMD)“, das Massachusetts Institute of Technology (MIT) hat das „Design Environment for Integrated Concur-

rent Engineering (DE-ICE)“, die Stanford University das „Space and Systems Development Laboratory (SSDL)“, die Utah State University das “Space System Analysis Laboratory” und die Carnegie Mellon University das „Engineering Design Research Center (EDRC)“ eingerichtet. In Europa beschäftigten sich derzeit die Università di Roma La Sapienza in Italien oder auch die Cranfield University in England mit dem CE und CDC. Die Universität Stuttgart hat für den „Space Station Design Workshop“ ein CDC eingerichtet. Der Autor selber war maßgeblich beim Aufbau des „Space Systems Concept Center (S_2C_2)“ (Abbildung 2-11) am Lehrstuhl für Raumfahrttechnik der Technischen Universität München (TUM) beteiligt.

Abbildung 2-11 Das Space System Concept Center am Lehrstuhl für Raumfahrttechnik der Technischen Universität München

Dieser Abschnitt beginnt mit einer allgemeinen Definition eines CDC und den wichtigen Elementen in einem solchen Raum. Anschließend wird eine Übersicht über die bereits existierenden CDC gezeigt. Da die Umstellung der Entwicklungsprozesse auf das CE zusammen mit CDC erhebliche Kosten verursacht (Schiffner, Kuss 2006 – Results of an international survey), ist dieser Ansatz derzeit nur bei großen Firmen wie EADS-Astrium und Forschungseinrichtungen wie ESA, NASA und Universitäten zu finden. Trotzdem besteht auch bei klein- und mittelständischen Unternehmen Interesse, diesen Ansatz in der Konzeptphase zu implementieren.

2.4 Werkzeuge für den Entwicklungsprozess

Um ein effektives Arbeiten im Rahmen von CE in einem CDC zu ermöglichen, benötigt das Entwicklungsteam unterstützende Werkzeuge und Modelle. Diese Werkzeuge dienen zum Modellieren, Simulieren, Analysieren und Dokumentieren des Systemdesigns.

Betrachtet man die Ergebnisse (vgl. Abbildung 2-12) aus der Studie über CDC in der Raumfahrt, dann sieht man, dass alle Befragten sowohl Standard-Softwarewerkzeuge als auch selbstentwickelte Softwarewerkzeuge einsetzen. Als Standard-Softwarewerkzeuge kommen hauptsächlich Microsoft Excel-Spreadsheets zum Einsatz (McInnes A. 2003 – Integrated Concurrent Design, S. 11). Obwohl über 80% der Befragten ihre Modelle für Raumfahrtprojekte bei neuen Projekten wiederverwenden, liegt der Anteil von objektorientierten Modellen nur bei 20% (Abbildung 2-12).

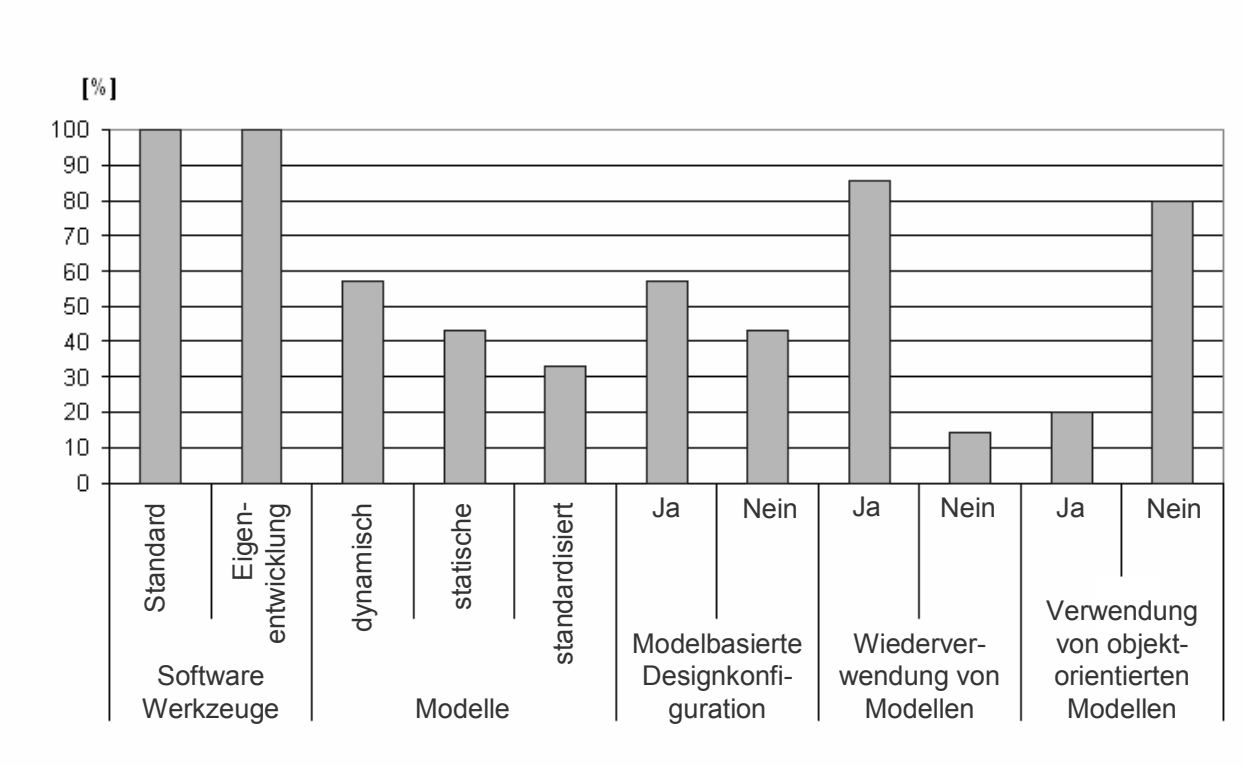


Abbildung 2-12 Ergebnisse (in Prozent) einer Umfrage über die verwendeten Werkzeuge für CE in einem CDC (Schiffner, Kuss 2006 – Results of an international survey)

In der Softwareentwicklung hat sich der Vorteil durch objektorientierte Modellierung und Entwicklung bereits erfolgreich bewährt, so dass dieser Ansatz auch für die Satellitenmodellierung in Betracht gezogen werden sollte. Früh (Früh – Software Entwicklung SwE im IT-Studiengang, S. 3–4) sieht die Vorteile der objektorientierten Entwicklung generell in den folgenden vier Punkten bei der Software-Entwicklung:

- **Bessere Verständlichkeit**

  Objektorientiert entwickelte Systeme führen zu einer Software-Struktur, die ein Abbild der realen Welt darstellt. Dies führt zum Beispiel in einem Textverarbeitungssystem zu Modulen (Klassen) wie Wort, Zeile, Abschnitt, Fußnote, Seite, Dokument, Wörterbuch usw. Eine solche Struktur ist von Natur aus leichter verständlich.
  Im Gegensatz dazu führen die klassischen funktionsorientierten Methoden auf eine Software-Struktur, bei der eine oberste Systemsfunktion immer weiter zerlegt wird. Die reale Welt ist aus Objekten aufgebaut. Die Vorstellung, dass ein System durch eine einzige, oberste Funktion charakterisiert werden kann, ist dagegen häufig fragwürdig. So kann man die oberste Funktion eines Betriebssystems nur schwer definieren.

- **Modifizierbarkeit**

  Jede brauchbare Software wird während ihrer Lebensdauer verändert. Es hat sich gezeigt, dass die Softwarekosten während der gesamten Lebensdauer eines Systems häufig zu weit über 50 Prozent im Unterhalt liegen.
  Die Software-Architektur sollte deshalb so aufgebaut sein, dass sie dem Veränderungsprozess widerstehen kann. Bei einer funktionalen Modularisierung (klassischer Top-Down Design) ist das aber häufig nicht der Fall. Insbesondere führt dies häufig zu einer Software, bei der die Daten über das ganze System verstreut sind (man kümmert sich bei der Modularisierung um die Funktionen und vernachlässigt die Daten).
  So sieht zum Beispiel der Ablauf eines Textverarbeitungsprogramms im Batch-Betrieb (ASCII-Files mit Kontrollinformationen als Input) grundsätzlich anders aus als bei einem modernen 'WYSIWYG[5]' Menü gesteuertes Textverarbeitungssystem. Die grundsätzlichen Objekte (Zeilen, Abschnitte usw.), sind jedoch geblieben.

[5] WYSIWYG steht für die englische Abkürzung “What you see is what you get” auf Deutsch “Was Sie sehen ist was sie bekommen”. Damit werden Programme wie Microsoft Word oder Macromedia Dreamweaver bezeichnet, bei denen der Benutzer grafisch das Resultat seiner Arbeit auf dem Bildschirm angezeigt bekommt.

- **Entwicklungsmethodologie**

  Zu Beginn eines Projekts ist es häufig sehr schwierig, exakte Spezifikationen aufzustellen. Bei einem Top-Down Design müssen aber ausgerechnet dann die folgenschwersten Entscheide gefällt werden, wenn die Unsicherheit am Größten ist. Objektorientierte Entwicklung kommt hierbei den natürlichen Denkprozessen näher (design a little, implement a little, test a little)

- **Wiederverwendung**

  Objektorientierte Entwicklung führt auf natürliche Weise zu wieder verwendbaren Software-Komponenten.

  (Früh – Software Entwicklung SwE im IT-Studiengang, S. 3–4)

Diese vier Punkte sind auch bei der Satellitenentwicklung von entscheidender Bedeutung. **Daher ist die objektorientierte Modellierung ein weiterer wichtiger Aspekt bei der im Rahmen dieser Arbeit entwickelten neuen Methodik.**

2.5 Zusammenfassung

Dieses Kapitel hat eine Übersicht über die Rahmenbedingungen bei der integrierten, modellbasierten Satellitenentwicklung gegeben. Dazu wurde der Stage-Gate-Prozess für die erfolgreiche Produktentwicklung dem Entwicklungsprozess der ESA gegenüber gestellt. Insbesondere wurde der multidisziplinäre und iterative Charakter der Konzeptphase beschrieben und daraus die **Notwendigkeit** der **modellbasierten Satellitenentwicklung**, der Ansatz des CE, des CDC und die integrierten Entwicklungsteams abgeleitet.

Abschließend wurde basierend auf einer Umfrage, die im Rahmen dieser Arbeit durchgeführt wurde, die derzeit verwendeten Werkzeuge und Modelle charakterisiert. Diese Werkzeuge sind das Handwerkszeug der Entwickler und somit ein essentieller Bestandteil der IMSE. Die Charakterisierung der Werkzeuge und Modelle hat die starke Verbreitung der klassischen funktional strukturierten Modellierung gezeigt. Die aus der Softwareentwicklung bekannten Vorteile durch eine objekt-orientierte Modellierung werden derzeit bei der IMSE noch nicht umgesetzt. Das folgende Kapitel, in dem die Werkzeuge bei der ESA und NASA beschrieben werden, wird dies noch stärker zeigen.

3 Aktuelle Ansätze für Entwicklungswerkzeuge in der modellbasierten Satellitenentwicklung

Im vorherigen Kapitel wurde ein Überblick über die integrierte, modellbasierte Satellitenentwicklung gegeben. Dabei wurde die Prozess- und der Arbeitsumgebung (Team und CDC) beschrieben. Als weiterer wichtiger Bestandteil wird ein zentrales Werkzeug für alle beteiligten Disziplinen benötigt, ohne den die modellbasierte Satellitenentwicklung nur schwer möglich ist. Dieses Werkzeug besitzt ein zentrales Datenmodell des Raumfahrtprojektes und erlaubt allen Teammitgliedern gleichzeitig auf die Daten zuzugreifen. Erst dadurch kann das CE erfolgreich umgesetzt werden. Es hat sich gezeigt, dass die meisten Anwender von CE und CDC dafür ihre eigenen Tools entwickeln und verwenden. Dabei kann man die folgenden drei großen Ansätze in der Raumfahrt unterscheiden:

- **Der NASA-Ansatz**

  Bei dem NASA-Ansatz handelt es sich um eine Keimzelle bestehend aus JPL und „The Aerospace Corporation“, aus der verschiedene Ansätze weiterentwickelt wurden.

- **Der ESA-Ansatz**

  Die ESA hat, inspiriert durch den NASA-Ansatz, ein eigenes Modell für den modellbasierten Satellitenentwurf entwickelt. Dieses Modell wird derzeit an

interessierte Forschungseinrichtungen und Firmen in Europa verteilt. Diese passen die Grundstruktur nach ihren Bedürfnissen an.

- **Der Münchner-Ansatz**

  Der Lehrstuhl für Raumfahrttechnik (LRT) an der Technischen Universität München (TUM) beschäftigt sich seit ca. 1990 mit der Systemmodellierung. Im Rahmen dieser Forschungstätigkeiten wurden verschiedene Modellierungslösungen für Systeme, Prozesse und Satellitensystemen entwickelt und in Werkzeugen implementiert.

Neben diesen drei raumfahrtspezifischen Ansätzen werden noch Lösungsansätze aus der Informatik betrachtet:

- **Lösungsansätze außerhalb der Raumfahrt**

  Nicht nur die Raumfahrt, sondern auch andere Branchen beschäftigen sich mit dem Problem der Modellierung von komplexen Systemen. Dazu haben Firmen wie z.B. Airbus Industries eine eigene interne Lösung für das Concurrent Engineering (Airbus Concurrent Engineering - ACE) entwickelt (ACE – Konstruktion ohne Grenzen). Darüber hinaus gibt es aus dem Bereich Softwareentwicklung erfolgreiche Ansätze wie die Unified Modeling Language (UML) und deren Ableitung für die Modellierung von komplexen Systemen (SysML). In diesem vierten Abschnitt wird auf UML und SysML näher eingegangen.

- **Softwarestruktur**

  Die im Rahmen dieser Arbeit vorgeschlagene Modellierungsmethode unterstützt nicht nur den Entwickler bei der Modellerstellung sondern hilft auch verschiedenen Fachdisziplinen gemeinsam an einem Modell zu arbeiten. Dazu wird in dem letzten Abschnitt die mögliche Softwarearchitektur für ein derartiges Werkzeug untersucht.

Neben diesen drei Lösungsansätzen für das CE im Bereich der Satellitenentwicklung wird an dieser Stelle noch auf Satellitenentwicklungstools verwiesen, die für spezielle Zwecke entwickelt wurden. So gibt es das Satellite Tool Kit (STK) der Firma AGI (http://www.agi.com/) für orbitmechanische Untersuchungen oder das System Engineering Design Tool (SEDT) für

das Konzeptdesign von Kleinsatelliten (Young-Keun, Ki-Lyong et al. 2005 – Development of System Engineering Design). Bei diesen Tools besteht das Problem, dass das dahinterliegende Daten- und Auslegungsmodell durch den Benutzer nur bedingt erweitert und angepasst werden kann. Diese Softwarelösungen sind nicht für das CE ausgelegt. Sie können zwar beim CE eingesetzt aber nicht aber als zentrales Modell zwischen allen Fachdisziplinen verwendet werden.

3.1 Der NASA-Ansatz

Der NASA-Ansatz ist die erste Implementierung von einem Design Center in Verbindung mit CE und modellbasierter Systementwicklung im Bereich der Raumfahrt. Bei dem Prozess handelt es sich um den beim CE erwähnten ICD-Prozess (Aguilar, Dawdy et al. 1998 – The Aerospace Corporation's Concept Design).

Der ICD-Prozess

Beim ICD Prozess gibt es sogenannte Designsitzungen und dazwischen Phasen, sogenannte „Hausaufgaben" Phasen, in denen Informationen für die nächste Designsitzung vorbereitet werden. In den Designsitzungen werden Lösungsansätze untersucht, diskutiert und Entscheidungen über das Design des Systems getroffen.

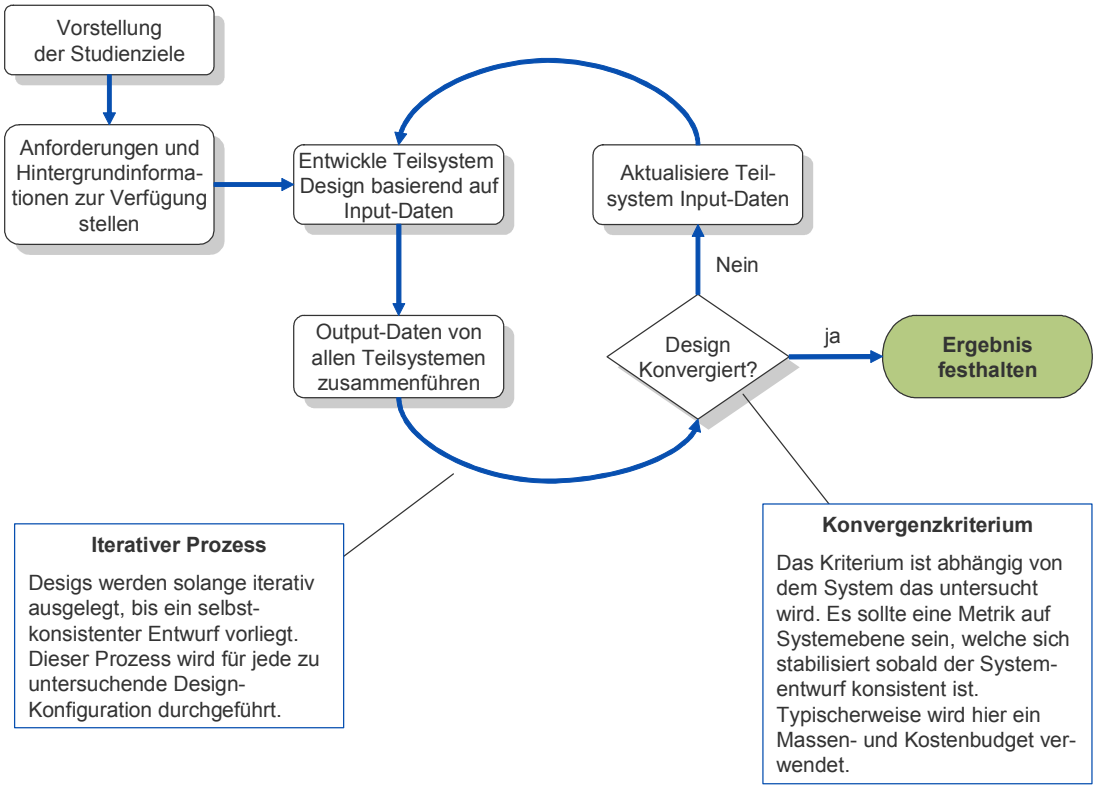


Abbildung 3-1 Die grundlegende Struktur des ICD Prozesses (McInnes A. 2003 – Integrated Concurrent Design, S. 8)

Abbildung 3-1 zeigt die grundlegende iterative Struktur des ICD Prozesses. Dabei ist deutlich das parallele Arbeiten der Fachdisziplinen zu erkennen. Alle Fachdisziplinen legen ihr Teilsystem basierend auf den vorhanden Inputdaten unabhängig von einander aus (Prozessschritt: *„Entwickle Teilsystem Design basierend auf Input Daten“*). Die dabei entstehenden neuen Daten werden den anderen Beteiligten über die Output Schnittstelle zur Verfügung gestellt (Prozessschritt: *„Output Daten von allen Teilsystemen zusammenführen“*). Da die verschiedenen Teilsysteme parallel arbeiten, müssen die Ergebnisse zusammengeführt und auf Konvergenz geprüft werden (Prozessschritt: *„Design konvergiert?“*). Sollten die Ergebnisse der einzelnen Teilsysteme noch nicht zusammenpassen (konvergieren) werden die Inputdaten mit den neuen Outputdaten aktualisiert und eine neue Schleife beginnt (Prozessschritt: *„Aktualisiere Teilsystem Input-Daten“*). Dieses iterative Vorgehen findet während den Teamsitzungen statt. Dazwischen können die Teammitglieder Ergebnisse für die nächste Teamsitzung vorbereiten.

Obwohl der Ansatz eine sehr gute Lösung für eine bestimmte Art von Problemstellungen darstellt, fasst Parkin drei Einschränkungen des ICD Ansatzes zusammen (Parkin, Sercel et al. 2003 – Icemaker™: an Excel-based environment, S. 10):

- **ICD benötigt gute Ingenieure**

  Der ICD Ansatz kann einen guten Entwicklungsprozess verbessern, aber er kann keine Probleme bei einem schlechten Prozess beheben. Das Team, welches den ICD Ansatz umsetzt, muss weiterhin die gleiche Disziplin und das formale Vorgehen bei der Entwicklung anwenden wie traditionelle Entwicklungsteams.

- **ICD automatisiert nicht vollständig die Systementwicklung**

  Ein ICD Team wird durch seine Erfahrungen und Fähigkeiten hervorgehoben, nicht durch die Software und Werkzeuge die es einsetzt. Ein ICD Team kann als ein „organisches Expertensystem“ betrachtet werden. Der ICD Ansatz ist kein Automatismus, bei dem der Kunde oder Systemingenieur einfach nur einen Satz an Anforderungen eingibt, einen Knopf drückt, und eine fertige Lösung präsentiert bekommt.

- **ICD unterstützt zum Teil keine detaillierten Simulationen.**

  Einzelne Analysen des aktuellen Designs (z.B. Finite Element Analysen) können die Arbeit des restlichen Teams blockieren. Diese Arten von Simulationen, die besonders viel Zeit benötigen, sollten daher am besten außerhalb der ICD Sitzungen abgehalten werden. Die Ergebnisse aus diesen Simulationen können dann in der folgenden Designsitzung eingebracht werden.

Das Werkzeug am JPL

Der ICD-Prozess hat gezeigt, wie die verschiedenen Disziplinen parallel ihre Teilsystemauslegung durchführen. Die entscheidende Phase in dem Prozess ist die Zusammenführung der verschiedenen Teilergebnisse zu einem konvergenten Ergebnis. Um die Zusammenführung während der Teamsitzungen zu unterstützen wurde ein zentrales Werkzeug entwickelt. Dieses Werkzeug – Relational Parameter Exchange Tool (R-PET) – basiert auf Microsoft Excel als Modellierungsumgebung und Benutzerschnittstelle und verwendete in der ersten Version

FrameMaker als zentrale Datenbank für den Datenaustausch (Sercel 1998 – Introduction to Integrated Concurrent Engineering).

In dieser Datenbank sind alle Input-Parameter des Modells abgebildet. Die Anzahl variiert zwischen 11 bis 519 Parametern mit einem Durchschnitt von 76 Parametern pro Teilsystem (Mark G.: Extreme collaboration, S. 91). Neben der großen Anzahl von Parametern die pro Teilsystem anfallen, beschreibt Mark auch Probleme bei Veränderung des Modells. Es sind **Inkonsistenzen** im Modell **aufgetreten**, wenn neue Elemente dem Modell hinzugefügt und diese nicht an alle Beteiligten kommuniziert wurden.

Durch die Erfahrungen, die das JPL und „The Aerospace Corporation" gesammelt haben, wurden neue Werkzeuge für den Ersatz von R-PET entwickelt. Sercel (Sercel, Clymer et al. 1999 – The Product Attributes Database PAD) schlägt dazu die sogenannte Product Attribute Database (PAD) vor, welches ähnlich wie das R-PET aufgebaut ist. PAD basiert bereits auf einer Oracle Datenbank und bietet als Benutzeroberfläche das Product Attribute Conversation Tool (PACT). PACT ermöglicht dem Benutzer Zugriff auf die Produktdaten. Gleichzeitig werden verschiedenen anderen Software-Werkzeugen über eine C-API[6] der Zugriff auf die PAD Daten ermöglicht.

McInnes (McInnes, Lang et al. 2003 – Integrated Data Exchange Architecture IDEA) schlägt die Integrated Data Exchange Architecture (IDEA) vor. Auch hier gibt es einen Datenserver und einen Client. Dabei können die Clients entweder in Microsoft Excel oder in einer anderen beliebigen Software implementiert sein, die über eine C-API[7] für ODBC[8] auf die Datenbank zugreift (siehe Abbildung 3-2).

---

[6] API Application Programming Interface steht für Standart Programmierschnittstelle vgl. Programmierschnittstelle 27.04.2007. C-API steht für eine API in der Programmiersprache C.

[7] Die Programmierschnittstelle (Application Programming Interface, API) definiert in ihrere Syntax und Semantik die Funktionen des Betriebssystems in Form von Systemdiensten (system services). Schneider, Werner et al. 2001 – Taschenbuch der Informatik

[8] Open Data-Base Connectivity (ODBC) ist ein durch Microsoft initiierter offener Standard eines Application Programmer Interface (API) für die Programmiersprache C Schneider, Werner et al. 2001 – Taschenbuch der Informatik.

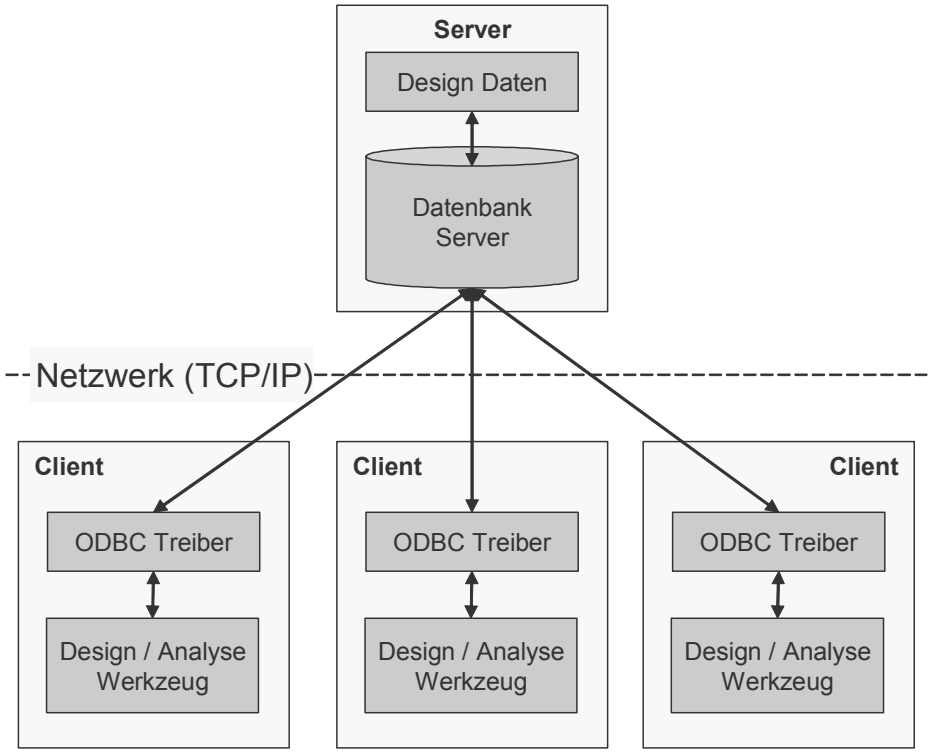


Abbildung 3-2 IDEA Software Architekture (McInnes, Lang et al. 2003 – Integrated Data Exchange Architecture IDEA, S. 11)

Diese Aufteilung von Client Server Architektur findet man auch in der nachfolgender ICE-Maker™ Lösung und in dem ESA IDM Ansatz.

Die Umsetzung Caltech – ICEMaker™

Am California Institute of Technology (Caltech) wurde zusammen mit dem JPL und TRW (12.12.2002 von Northrop Grumman übernommen) das Werkzeug ICEMaker für den ICD Ansatz entwickelt. Parkin (Parkin, Sercel et al. 2003 – Icemaker™: an Excel-based environment) beschreibt das Werkzeug ICEMaker durch den dahinterliegenden Modellierungsansatz, die Funktionalität und die Umsetzung in der Software. Die Idee hinter dem Werkzeug ist, ähnlich wie schon bei R-PET, die Eigenschaften (Parameter) des Systems in einer relationalen Datenbank abzuspeichern. Die Clients, Werkzeuge des jeweiligen Teilsystems, greifen auf die notwendigen Daten in der Datenbank zu. Diese Daten können auch Parameter eines anderen Teilsystems sein.

Da Microsoft Excel von fast allen Ingenieuren verstanden und benutzt wird oder ihnen zumindest zur Verfügung steht, spielt es eine zentrale Rolle bei dem ICEMaker Ansatz. Jedes

Teilsystem wird genauso wie beim NASA- und ESA-Ansatz durch eine eigene Microsoft Excel-Datei repräsentiert. Der Microsoft Excel Ansatz erlaubt sowohl eine „quick and dirty" Modellierung als auch eine genauere Auslegungsberechnung durch die in Microsoft Excel vorhanden Funktionen. Des Weiteren kann man durch Visual Basic for Applications (VBA) komplexe Berechnungen und Analysen implementieren. Zusätzlich bieten kommerzielle Analysewerkzeuge wie Mathematica, Matlab oder LabView bereits eine integrierte Microsoft Excel Schnittstelle an, die auch bei CAD Werkzeugen wie SolidWorks oder CATIA vorhanden sind. Aus Microsoft Excel heraus können zudem auch selbstentwickelte DLL[9] aufgerufen werden.

Durch diese Flexibilität in Microsoft Excel ist sichergestellt, dass alle Beteiligten ihr bevorzugtes Simulations- und Designwerkzeug verwenden können. Gleichzeitig ist sichergestellt, dass alle Systemparameter aktuell in einer Datenbank vorliegen. Abbildung 3-3 zeigt die zugrundeliegende Architektur von ICEMaker, die weitgehend mit dem JPL Ansatz (R-PET und IDEA) übereinstimmt, da es vermutlich auf dem R-PET Ansatz der NASA aufbaut.

Die Modularität des Systems ist durch die Verwendung von Microsoft Excel-Dateien, die als Teilsysteme bezeichnet werden, implementiert. Die Kommunikation zwischen allen Teilsystemen wird durch die Client-Server Architektur (Abbildung 3-3) bewerkstelligt. Der zentrale Server übernimmt hierbei die Aufgabe des Datenaustausches zwischen den Clients (Teilsystemen).

In der ursprünglichen Version von ICEMaker™ war die zentrale Datenbank auch als Microsoft Excel-Datei realisiert worden (vgl. ESA-IDM Ansatz, Abschnitt 3.2). Dabei wurde durch Makros in den einzelnen Microsoft Excel-Dateien der Datenaustausch realisiert. Dieser Ansatz wurde in den folgenden Versionen von ICEMaker™ durch eine leistungsfähigere, eigenständige Visual Basic Anwendung ersetzt.

[9] Dynamic Link Library (DLL) bezeichnet allgemein eine Dynamische Bibliothek und können Programmcode, Daten, und Ressourcen, in irgendeiner Kombination enthalten.Schneider, Werner et al. 2001 – Taschenbuch der Informatik

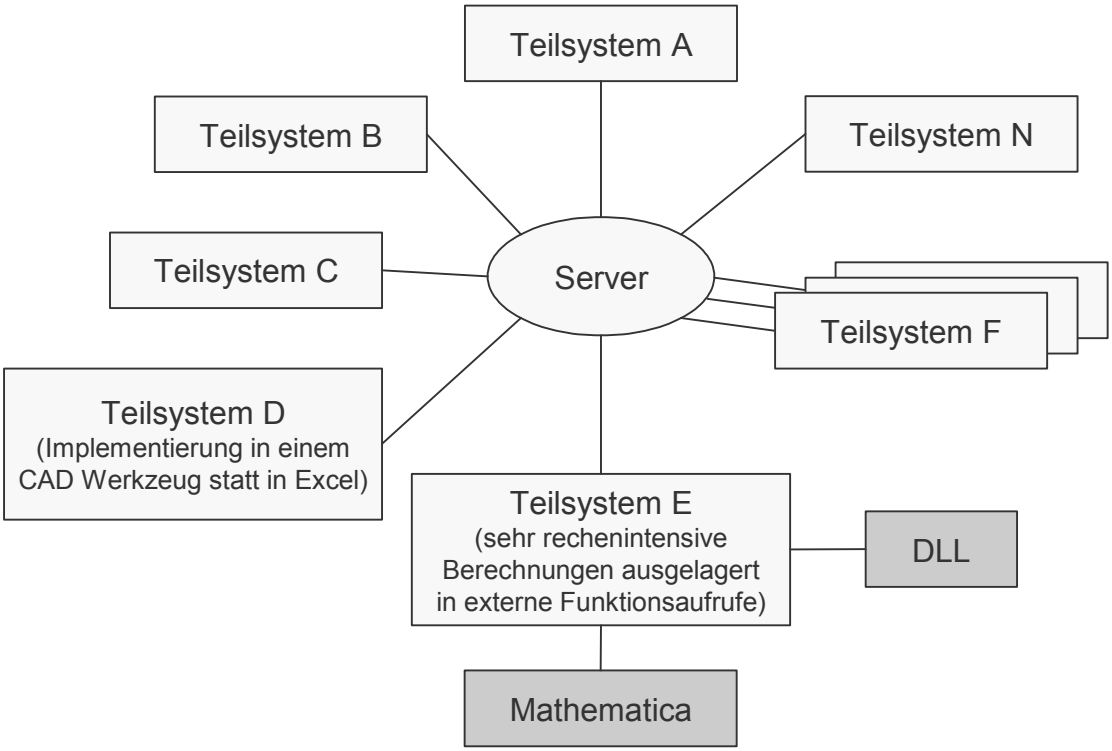


Abbildung 3-3 ICEMaker™ Client Server Architektur mit Zentralem Server und den verschiedenen Teilsystemen die als Clients angebunden sind (Parkin, Sercel et al. 2003 – Icemaker™: an Excel-based environment)

Bei dem Ansatz von Caltech steht jedem Projekt ein eigener Server zur Verfügung. Dadurch werden die Daten innerhalb eines Projektes gekapselt. Ein Austausch von Modellen zwischen verschieden Projekten wird dadurch erschwert.

Zusammenfassend bietet ICEMaker™ eine sehr leistungsfähige Entwicklungsumgebung. Es verwendet eine zentrale Datenbank, die aber nur auf ein Projekt beschränkt ist. Die Datenbankschnittstelle muss bei ICEMaker™ genauso wie bei dem NASA- und ESA-Ansatz auf der Client Seite implementiert werden. Dieser Ansatz für den Datenbankzugriff kann bei einer gegebenenfalls notwendigen Zugriffskontrolle Probleme verursachen.

3.2 Der ESA-Ansatz

Zur Unterstützung des CE Ansatzes der ESA wurde ein eigenes CDC, die Concurrent Design Facility (CDF), eingerichtet und ein eigenes Modell für den modellbasierten, integrierten Satellitenentwurf entwickelt. Dieses Model, genannt Integrated Data Model (IDM), basiert rein

auf Microsoft Excel -Dateien. Für jedes Missionselement und Satelliten-Teilsystem gibt es eine dedizierte Microsoft Excel-Datei. Zur Verknüpfung der Microsoft Excel-Dateien gibt es zwei zentrale Dateien, über die alle Daten ausgetauscht werden (siehe Abbildung 3-4).

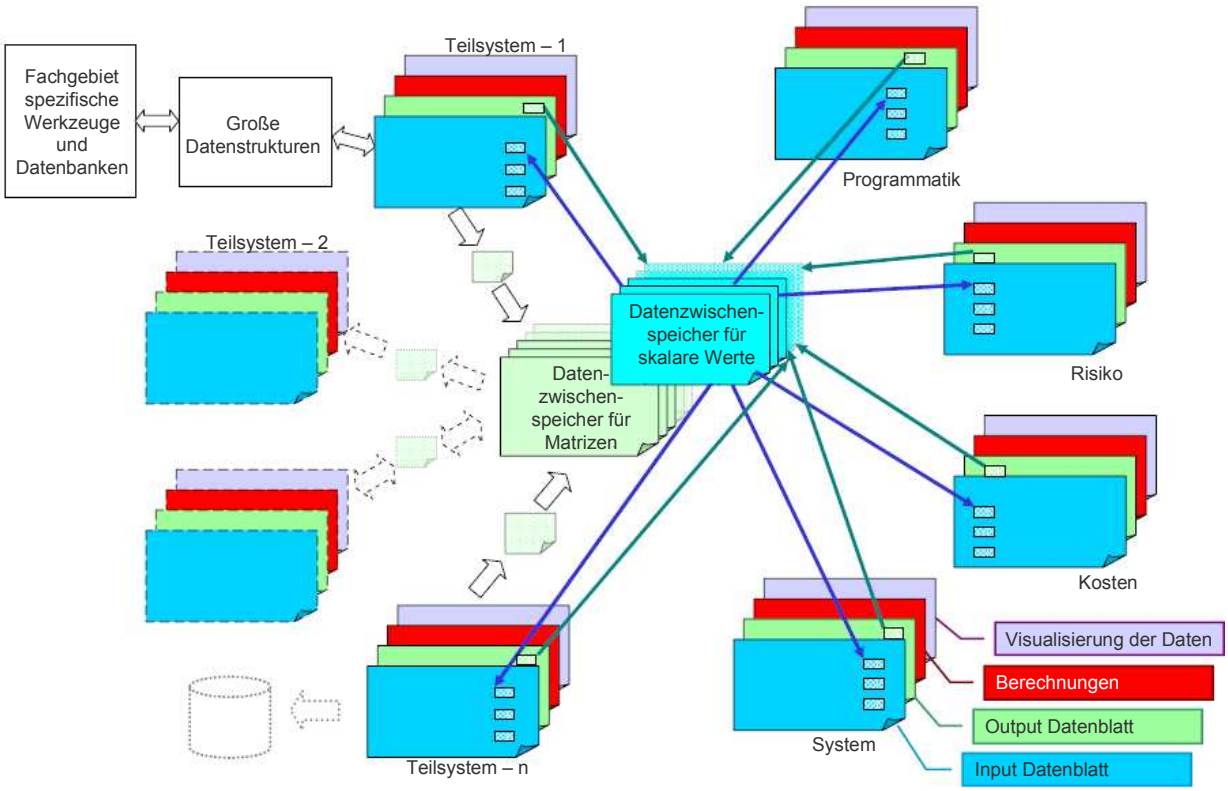


Abbildung 3-4 Software Architektur des Integrated Data Models der ESA verwendet im CDF (Bandecchi, Robin Biesbroek 2004 – The ESTEC Concurrent Design Facility, S. 2)

Während des 2nd Concurrent Engineering for Space Applications Workshop 2006 (CEW06) in Noordwijk wurden verschiedene Ansätze und Implementierungen für das CE und die CDCs diskutiert. Der Workshop hat gezeigt, dass neben der ESA das IDM auch schon von Firmen wie z.B. EADS-Astrium, der Französische Raumfahrt Agentur (CNESS) und von Universitäten eingesetzt wird.

Da das IDM nur eine Grundstruktur aus den Microsoft Excel-Dateien darstellt, können Hochschulen und Industrie diesen Ansatz leicht an eigene Bedürfnisse anpassen. Dadurch ersparen sich die Nutzer eine Eigenentwicklung für das Grundmodell. Die Funktionen zur Berechnung und Auslegung von Satellitensystemen müssen von den Anwendern selber entwickelt werden.

Aufgrund der Modellierung in Microsoft Excel sind die Schnittstellen und die Abhängigkeiten im Modell nur schwierig nachzuvollziehen. Gaudenzi (Gaudenzi, Morelli 2006 – Basic con-

current design procedures) schlägt daher eine Möglichkeit zur Beschreibung der einzelnen Modelle durch Diagramme vor. Dabei basiert sein Vorschlag auf dem Klassen-Diagramm aus der UML. Die einzelnen Diagramme werden nicht direkt mit dem Modell verknüpft, so dass sich Änderungen im Modell nicht automatisch im Diagramm widerspiegeln.

Ein weiteres Problem des ESA-IDM Ansatzes ist die fehlende Abstraktion im IDM-Modell. Bei einem objektorientierten Modellierungsansatz werden die Werte von der Elementbeschreibung getrennt. Die Elementbeschreibung wird in der abstrakten Klassendefinition festgelegt. Dadurch kann man relativ einfach durch Austausch der Klassendefinition ein aktuelleres Modell einbinden. Dieses Problem wurde von Bandecchi (Bandecchi 20.06.2006 – Probleme bei dem Austausch) im Rahmen des CEW06 angesprochen.

Die ESA plant derzeit eine Weiterentwicklung des IDM Ansatzes, bei der eine zentrale Datenbank die Modelldaten speichern soll. Zusätzlich wird die Möglichkeit untersucht, CDC und die Modelle von verschiedenen Partnern (Firmen und Forschungseinrichtungen) zusammenzuschalten. Hierbei sind komplizierte Mechanismen für den Datenaustausch notwendig, da die Partner zum Teil ihr Modelle nicht an andere Partner herausgeben möchten.

3.3 Der Münchner-Ansatz

Igenbergs (Igenbergs 1993-2007 – Grundlagen der Systemtechnik, S. 3) definiert ein allgemein gültiges Vorgehen für die Modellierung von Systemen durch die folgenden vier Regeln:

1. **Ein System besteht aus Elementen**
2. **Ein Element hat Attribute**
3. **Elemente stehen miteinander in Wechselwirkung**
4. **Ein Element kann ein System sein.**

Durch diese vier Regeln wird ein System durch eine Struktur (Hierarchie) und durch Abhängigkeiten (Wechselwirkung) beschrieben. Dadurch können beliebige Systeme modelliert werden. Dieses Vorgehen wird auch als Elementkonzept bezeichnet. Ehrlenspiel (Ehrlenspiel 2007 – Integrierte Produktentwicklung) schlägt einen ähnlichen Ansatz für die Systemmodellierung vor (Abbildung 3-5). Bei diesem Ansatz können die Elemente als Black Box betrachtet werden. Auch hier können die Elemente Teilsysteme eines übergeordneten Systems sein bzw. selber untergeordnete Elemente besitzen. Die dadurch entstehenden Beziehungen wer-

den als Struktur des Systems definiert. Bei Ehrlenspiel wird zudem noch auf die Bedeutung der Systemumwelt als ein wichtiges Element bei der Systemdefinition hingewiesen.

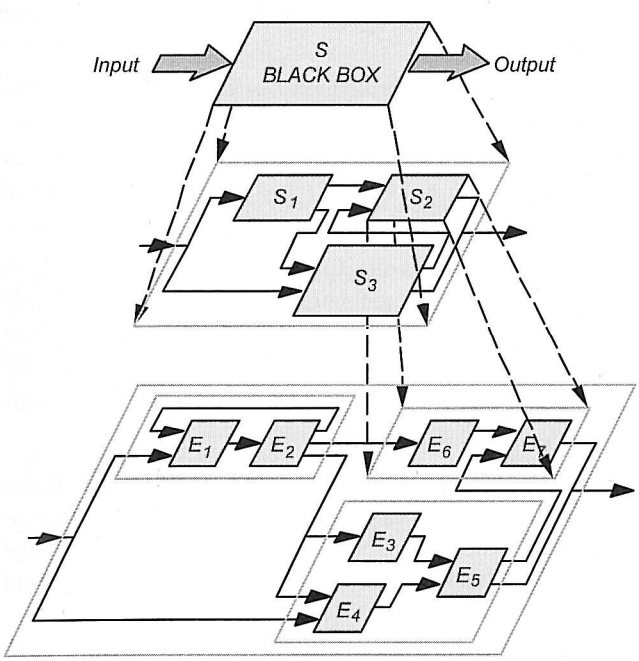


Abbildung 3-5 Systemdefinition für ein allgemeines Systemmodell (Ehrlenspiel 2007 – Integrierte Produktentwicklung, S. 22). Das Gesamtsystem ist hierbei als Black Box dargestellt und in Teilsysteme (S_1-S_3) und diese wiederum in Elemente (E_1-E_7) unterteilt.

In dem Elementkonzept werden die notwendigen Informationen zur Beschreibung des Elementes, auch Attribute genannt, durch das Element selbst gekapselt. Durch diese Definition des Elementkonzeptes kann man diesen Ansatz als objektbasiert bezeichnen. Zu den Attributen eines Elementes oder Objektes zählen die Inputs, Outputs, Parameter und Funktionen. Die Inputs und Outputs definieren dabei die Schnittstellen des Elementes, die Parameter sind die Charakteristika des Elementes und die Funktionen definieren die Abhängigkeiten zwischen den Inputs, Parametern und Outputs. Der Münchner-Ansatz für die Modellierung definiert keine Systemabstraktion in Form von Klassen. Eine Relation, genauer Flussrelation (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 47), definiert einen Informationsfluss zwischen einem Input und Output. Es ist aber nicht näher definiert wie diese Information in Bezug auf Sender oder Empfänger übertragen wird. Es wird nicht genauer spezifiziert, wer der aktive und wer der passive Partner bei der Informationsübertragung ist.

Walther (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 28–29) konkretisiert das Elementkonzept von Igenbergs und schlägt zur Abbildung der Zusammenhänge innerhalb

eines Elementes sowie des Gesamtmodells eine Matrix-Darstellung vor. Er definiert dabei verschiedene Arten und Typen von Relationen (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 47ff). Die beiden Arten von Relationen sind die Systemrelation und die Flussrelation. Die Systemrelationen definieren den strukturellen Aufbau, die Flussrelationen den Informationsaustausch zwischen den Elementen. Zur genaueren Spezifikation der Flussrelationen dienen Relationstypen. Walther schlägt für die Katalogisierung der Typen eine hierarchische Struktur vor und definiert dabei eine Reihe von Relationstypen. Diese Relationstypen sind vergleichbar mit den von Otto und Wood (Otto, Wood 2001 – Product design, S. 1011ff) vorgeschlagenen Flussrelationstypen für die Funktionsmodellierung.

Da beide Informationen, Struktur und Abhängigkeit, in einem Modell existieren, schlägt Walther zwei unterschiedliche Diagramme für die Aufbau- und Ablaufstruktur vor (Abbildung 3-6).

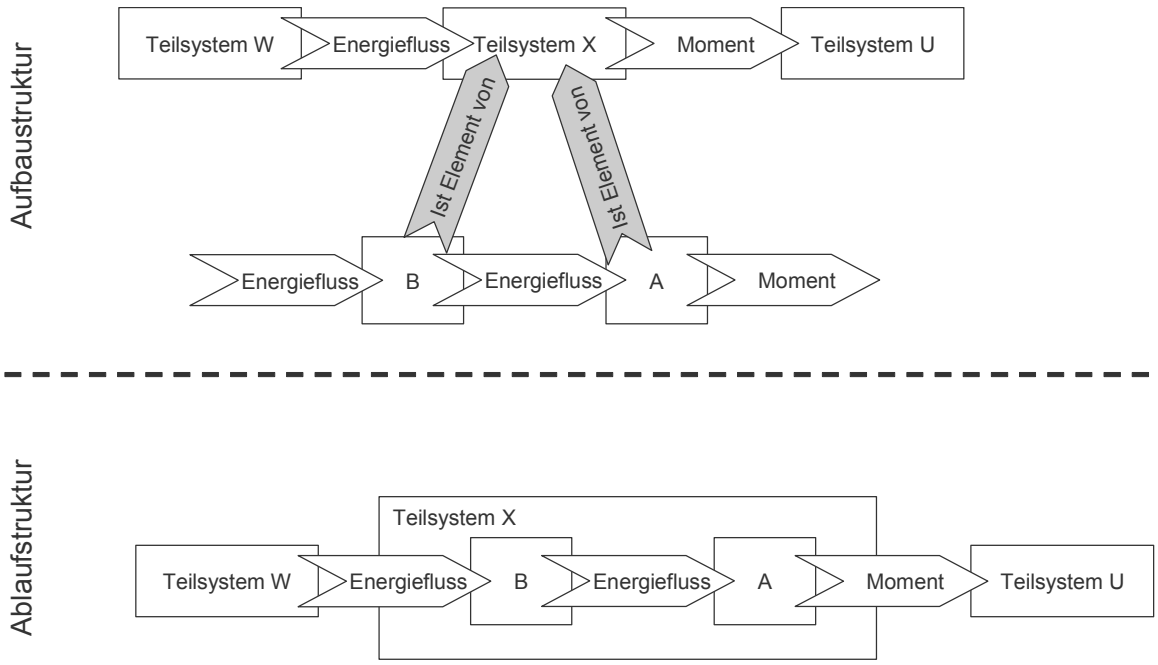


Abbildung 3-6 Aufbau- und Ablaufstruktur eines Systems in einer Beschreibungsform bestehend aus Teilsystemen, Strukturrelationen und Flussrelationen (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 65)

Des Weiteren schlägt Walther (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 57) einen dreistufigen Ansatz für die Modellierung von Systemen vor. In einem ersten Schritt wird ein grobes Modell mit allen Elementen und Relationen des Systems erzeugt. Im folgenden zweiten Schritt werden die Elemente um die Attribute erweitert, wobei die Funktion als

qualitative Abhängigkeit abgebildet wird. Bei dem letzten Schritt wird den Elementen eine quantitative Funktion hinzugefügt. Walther zeigt zudem wie man mit dem Modellierungsansatz verschieden Systemtypen abbilden und miteinander verknüpfen kann. Er unterscheidet dabei ein Zielsystem (Anforderungskatalog), Objektsystem (Produkt) und Handlungssystem (Entwicklungsbereich) (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 85).

3.3.1 MuSSys

Negele (Negele 1998 – Systemtechnische Methodik zur ganzeitlichen Modellierung) erweitert den Ansatz von Walther zur Abbildung des Prozesssystems. Zusätzlich zu dem Modellierungsansatz definiert er einen systematischen Prozess für die integrierte Produktentwicklung genannt ZOPH. ZOPH wird aus den Anfangsbuchstaben der vier Modellierungsbereichen eines Produktes Zielsystem, Objektsystem, Prozesssystem, and Handlungssystem abgeleitet. Dieser Modellierungsansatz wurde in der Anwendung MuSSys (Modellierung und Simulation von Systemen) implementiert.

3.3.2 IPO

Für die Darstellung eines Prozesssystems wurde von Fricke (Fricke, Negele et al. 06.02.1998 – Grundlagen der Prozeßmodellierung; Negele, Fricke et al. – Modelling of Integrated Product Development) basierend auf dem Elementkonzept ein eigener Modellierungsansatz für die Prozessmodellierung basierend auf Input – Prozess – Output (IPO) abgeleitet. Dieser verwendet für die Prozessmodellierung die gleichen vier Merkmale wie zur Modellierung eines Systems. Zusätzlich wurde die Elementdarstellung um spezielle Informationen wie Ressourcen, Verantwortlichkeit und einer Ampel, die sowohl das Prozessrisiko als auch den Status der Schnittstellen darstellt, erweitert.

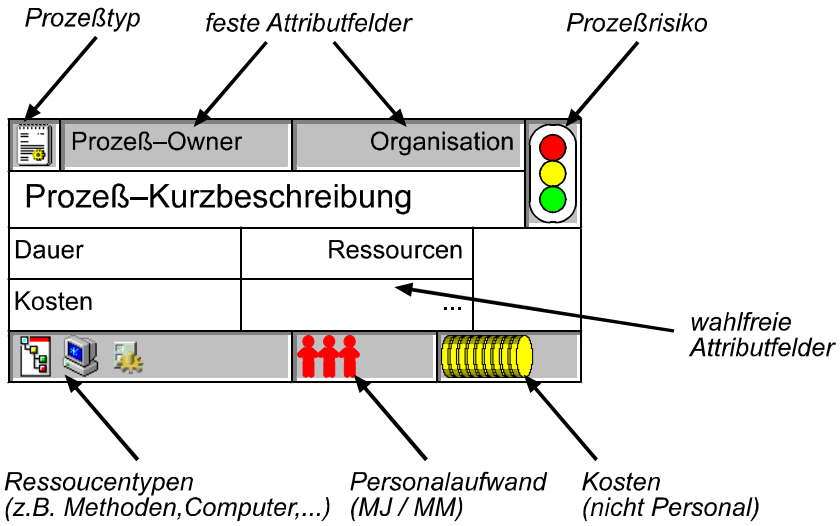


Abbildung 3-7 Visualisierung eines einzelnen Prozesses in der IPO Darstellung (Fricke, Negele et al. 06.02.1998 – Grundlagen der Prozeßmodellierung)

3.3.3 MuSSat

Wilke (Wilke 2002 – Integrierter modellbasierter Satellitenentwurf)und Quirmbach (Quirmbach 2001 – Integration der System- und Kostenentwicklung) haben in dem Werkzeug „**M**odellierung **u**nd **S**imulation von **Sat**ellitensystemen“ (MuSSat) die Grundkonzepte von Walther (Walther – Systemtechnisches Verfahren zur Bestimmung) und Negele (Negele 1998 – Systemtechnische Methodik zur ganzeitlichen Modellierung) aufgegriffen und an die Anforderungen der integrierten, modellbasierten Satellitenentwicklung angepasst. Dabei hat Wilke das Objektsystem (Satellitensystem) und Quirmbach das Kostensystem von Walther bzw. Prozess- und Handlungssystem von Negele in MuSSat implementiert und miteinander vereint.

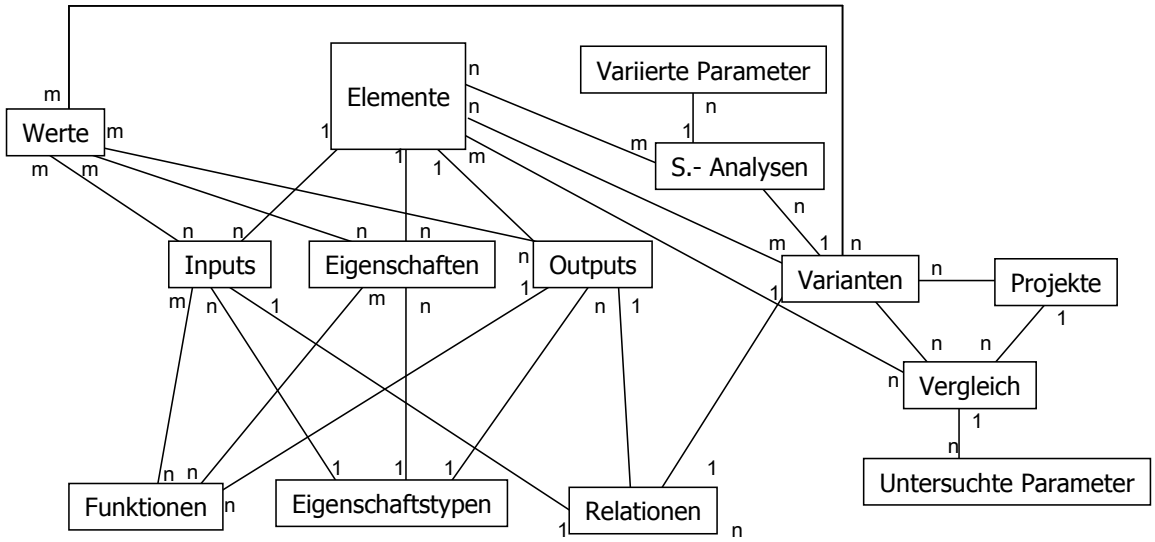


Abbildung 3-8 Relationales Datenmodell des Objektsystems in MuSSat dargestellt im Entity-Relationship Modell[10] (Wilke 2002 – Integrierter modellbasierter Satellitenentwurf, S. 88)

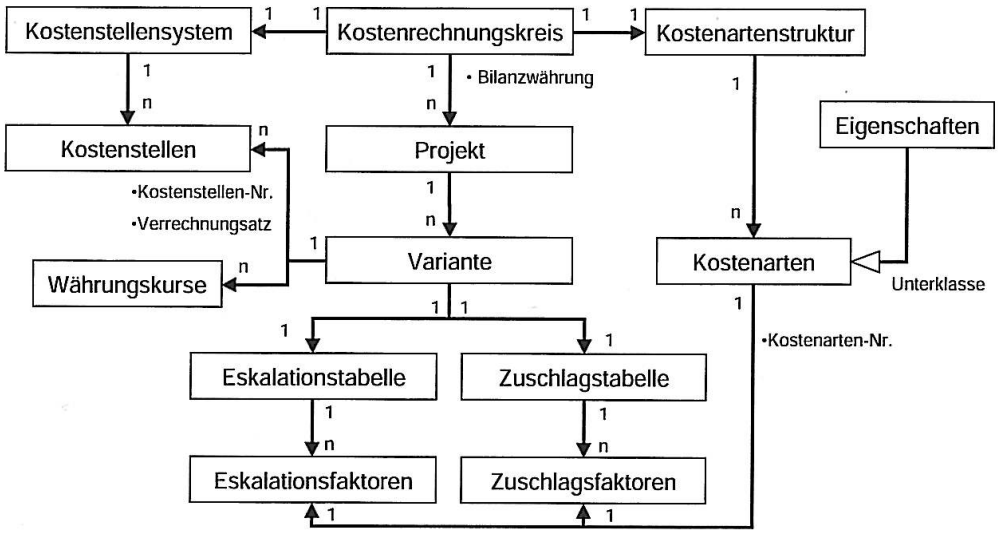


Abbildung 3-9 Aufbau des allgemeinen Projektmodells für die Kostenberechnung dargestellt in einem Entity-Relationship Modell (Quirmbach 2001 – Integration der System- und Kostenentwicklung, S. 49)

[10] Entity-Relationship Modell: Die Entity-Relationship Modellierung basiert auf Entitätstypen und Beziehungen zwischen diesen. Ein Entitätstyp beschreibt eine Menge gleichartiger Dinge, die im Datenmodell gespeichert werden. Eine Beziehung verknüpft Entitätstypen untereinander und deren Bedeutung wird durch die Ordnung ausgedrückt. Diese Ordnung wird durch annotierte Zahlen definiert, die angeben, wie viele Entitäten der beteiligten Entitätstypen an einer Beziehung partizipieren können.

Abbildung 3-8 (Produktmodell) und Abbildung 3-9 (Kostenmodell) zeigen dabei das Datenmodell, wie es in der relationalen Datenbank implementiert wurde. Man erkennt hierbei zum Einen die Trennung der Modelle für Kosten und Objektsystem in der Datenbank. Zum Anderen zeigt die (n-m)-Beziehung zwischen Element und Variante, dass ein Element in mehreren Varianten vorkommen kann. Genau genommen kann ein Element nur ein Mal in einer Variante existieren. Dies führt zu folgenden Problemen:

- **Eine Wiederverwendung eines Elementes innerhalb einer Variante ist nicht möglich (nur als Kopie).**
- **Änderungen an einem Elemente wirken sich auf alle Varianten und Projekte in dem es verwendet wird aus. Dies gilt auch für die Varianten und Projekte die bereits abgeschlossen sind.**
- **Auf Produktebene findet Modellierung basierend auf Klassen statt.**

MuSSat stellt somit ein Modellierungswerkzeug für die Satellitenentwicklung zur Verfügung, das auf Objektebene einen objektbasierten Ansatz verfolgt. Gleichzeitig wird auf der Kostenseite einen objektorientierter Modellierungsansatz mit Klassen und Vererbung (Quirmbach 2001 – Integration der System- und Kostenentwicklung, S. 47), wie es bereits ansatzweise von Negele (Negele 1998 – Systemtechnische Methodik zur ganzeitlichen Modellierung, S. 117) durch die Definition von Elementtypen umgesetzt wurde, implementiert. Dabei können die Elementtypen nicht von einander abgeleitet werden.

MuSSat wurde in enger Kooperation mit dem Industriepartner DASA (heute Astrium-EADS) entwickelt und zum Teil von der Firma eingesetzt. Daneben wurde MuSSat bei der Ausbildung von Studenten im Rahmen des „Concurrent Design Center“ Workshops (siehe Abschnitt 4.4) erfolgreich eingesetzt und diente als Grundlage für weitere Forschungs- und Weiterentwicklungsprojekte im Rahmen von BaiCES (siehe Abschnitt 4.5).

3.4 Lösungsansätze außerhalb der Raumfahrt – UML und SysML

Wie die vorherigen Abschnitte gezeigt haben, werden bereits seit mehreren Jahren Werkzeuge für den modellbasierten Entwurf von Satellitensystemen entwickelt und diese auch eingesetzt. Im Gegensatz dazu werden in diesem Abschnitt Modellierungsansätze außerhalb der Raumfahrt beschrieben und deren möglicher Einsatz für die integrierte, modellbasierte Satelliten-

entwicklung (IMSE) betrachtet. Im speziellen werden die graphischen Modellierungsansätze aus der Informatik (UML) und seine Ableitung für den Systementwurf (SysML) besprochen.

UML und SysML sind beide graphische Modellierungssprachen. Die graphische Darstellung einer Problemstellung, bei UML ist es Software und bei SysML ein komplexes System, erlaubt dem Entwickler das Problem klar zu strukturieren und zu analysieren. UML und SysML überlässt dem Entwickler die Einsatzweise des Modells. Im Falle von UML wird normalerweise eine lauffähige Software aus dem Modell generiert. Diese Generierung wird aber nicht durch UML definiert, sondern ist eine Fähigkeit des UML - Modellierungswerkzeuges. Dies ist bei der Beurteilung von UML und SysML als Einsatz bei der IMSE zu bedenken, da hier nicht nur ein Datenmodell oder Modell eines Raumfahrtsystems gebraucht wird, sondern ein Werkzeug, welches auch die notwendigen Funktionen für die Auslegungsrechnung, Designanalyse und dem Datenaustausch zur Verfügung stellt.

UML

Im Bereich der Softwareentwicklung hat sich die Unified Modeling Language (UML) als Standart für die Modellierung von Softwaresystemen durchgesetzt. UML ist eine allgemeingültige Modellierungssprache, die standardisierte graphische Notationen enthält. Mit diesen Notationen kann ein abstraktes Modell von einem Softwaresystem, UML Modell genannt, erzeugt werden. UML kann durch Profile und Stereotypen an die Bedürfnisse des Entwicklers angepasst werden.

UML wird offiziell von der Object Management Group (OMG; http://www.omg.net/) definiert. UML wird durch ein UML Meta-Modell – Meta-Object Facility Meta Model (MOF) – beschrieben. Das UML Meta-Modell sowie alle UML Modelle können in einer XML[11] Metadata Interchange (XMI) abgespeichert werden.

Ein UML Modell kann aus den verschiedensten Elementen wie zum Beispiel Paketen, Klassen und Assoziationen bestehen. Die korrespondierenden Diagramme sind graphische Repräsentationen von Teilen des UML Modell. UML Diagramme enthalten graphische Elemente

[11] XML: Extensible Markup Language – XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien Extensible Markup Language 06.07.2007.

(Knoten, die über Pfade mit einander verbunden sind), die die Elemente des UML Modell widerspiegeln.

Beispiel (Object Management Group (Hg.) Februar 2007 – Unified Modeling Language, S. 675):
Zwei assoziierte Klassen sind in einem Paket definiert. In dem Paket-Diagramm werden die Klassen mit dem Klassensymbol dargestellt und ein Assoziationspfad verbindet beide Klassensymbole.

Die Diagramme in UML werden in zwei Kategorien aufgeteilt:

- **Struktur-Diagramme**

  Die Struktur-Diagramme zeigen den Aufbau der Software. Zu den Struktur-Diagrammen zählen unter anderem das Klassen-Diagramm oder das Paket-Diagramm.

- **Verhalten-Diagramme**

  Die Verhalten-Diagramme beschreiben, wie sich die Software dynamisch verhält. Dazu gehören unter anderem das State-Machine-, das Aktivitäts- oder das Use-Case- Diagramm.

Ein wichtiger Punkt bei UML ist die Tatsache, dass UML nur eine graphische Darstellung eines Modells bzw. einer Software ist. Die am Markt befindlichen Werkzeuge zur Modellierung mit UML erzeugen aus den Informationen, die in dem UML Modell abgelegt sind, einen vorgefertigten Softwarecode, der dann von dem Entwickler noch erweitert werden muss. Sollte mit UML ein Satellitensystem beschrieben werden, wird der aus dem UML Modell gewonnene Softwarecode selbst ein dynamisches Modell des Satelliten darstellen aber kein Modell zur Auslegung einer technischen Lösung.

SysML

Um auch komplexe Systeme wie Satelliten modellieren zu können wird derzeit an der Systems Modeling Language (SysML) gearbeitet. Die Ursprünge von SysML können auf eine strategische Entscheidung des INCOSE zurückverfolgt werden und basiert auf UML (Object Management Group (Hg.) Mai 2006 – OMG Systems Modeling Language OMG, S. 1). Daher ist SysML genauso wie UML eine reine graphische Darstellung von Systemen. Mit SysML

kann die Spezifikation, Analyse, Design, Verifikation und Validierung von komplexen Systemen durchgeführt werden. Dazu können in SysML Hardware, Software, Informationen, Prozesse, Personal und Einrichtungen abgebildet werden.

Die derzeit als „Final Adopted Specification“ vorliegende SysML Spezifikation wurde im März 2007 dem Kontrollgremium der OMG vorgelegt. Sobald diese die Spezifikation verabschiedet, liegt OMG SysML Specification v1.0. vor. Eine Übersicht der in SysML vorhanden Diagramme liefert Abbildung 3-10.

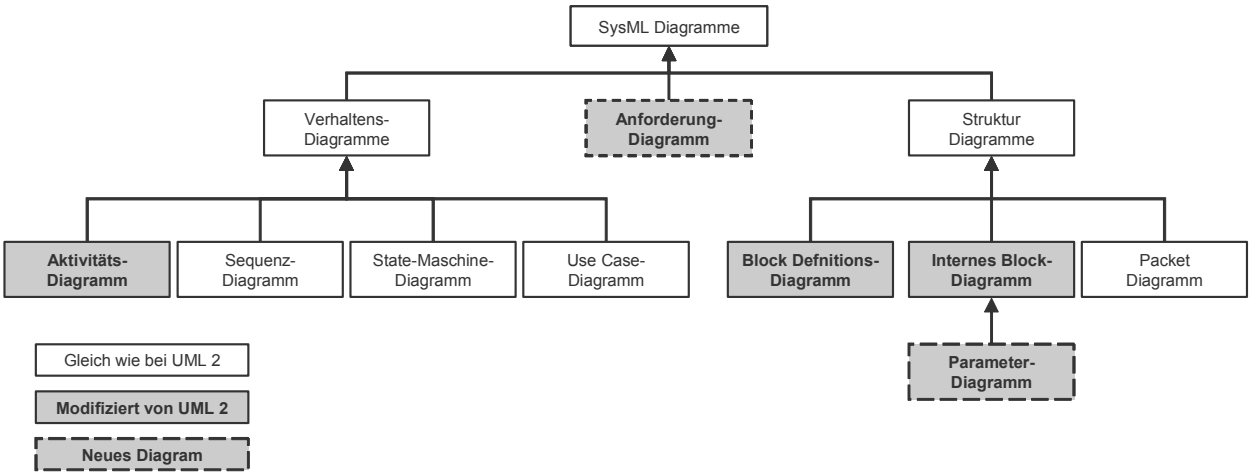


Abbildung 3-10 In SysML definierte Diagramme und ihre Abstammung (OMG SysML)

SysML ist genauso wie UML eine graphische Modellierungssprache bestehend aus einer Semantik (Bedeutung) und einer Notation (Darstellung der Bedeutung). Beide sowohl SysML als auch UML sind keine Software-Werkzeuge. Somit können von diesen Ansätzen die graphischen Darstellungen übernommen werden. Für die Definition der neuen Modellierungsmethode wurden das Klassen-Diagramm für die Klassendarstellung und das Packet-Diagramm für die System-Darstellung als Grundlage genommen.

3.5 Software Architektur

Das Kapitel hat die bisherigen Modellierungsansätze aus der Raumfahrt, Informatik und dem Systems Engineering dargestellt. Für die Implementierung und die erfolgreiche Umsetzung des Modellierungsansatzes wird abschließend die mögliche Software Architektur für die Implementierung der neuen Modellierungsmethodik diskutiert. Für die Definition der geeigneten Software Architektur wird von der Ähnlichkeit zwischen den Entwicklungsteams und einer

virtuellen Firma ausgegangen. Dies lässt sich mit der Tatsache begründen, dass die Teammitglieder in den Entwicklungsteams bei der IMSE Repräsentanten verschiedener Fachabteilungen sind. Somit stellt das Team eine virtuelle Firma innerhalb einer Firma dar. Dieser Eindruck wird durch die in den Firmen vorhandenen „Produktcenter“ in den Firmen verstärkt. Jedes Produktcenter wird als eigene Firma innerhalb eines Konzerns geführt. Für die Auswahl der Softwarearchitektur wird daher das Thema Informationsverarbeitung (IV) bei virtuellen Firmen angesprochen. Es wird dabei näher auf den Ansatz für die Softwareinfrastruktur von virtuellen Unternehmen (VU) nach Faisst näher eingegangen (Faisst Wolfgang 1995 – Welche IV-Systeme sollte ein Virtuelles). Faisst baut seinen Ansatz dabei auf einer Untergliederung von VU nach der Definition von Snow (Snow, Miles et al. 1992/0 – Managing 21st century network organizations, S. 5–20) auf.

Definition von Virtuellen Unternehmen

Snow definiert drei Unternehmensnetzwerktypen, zwischen denen ein fließender Übergang besteht:

- **Internes Netzwerk**
- **Stabiles Netzwerk**
- **Dynamisches Netzwerk**

Die folgenden Abbildung 3-11 zeigt die unterschiedlichen Konfigurationen der Unternehmensnetzwerktypen.

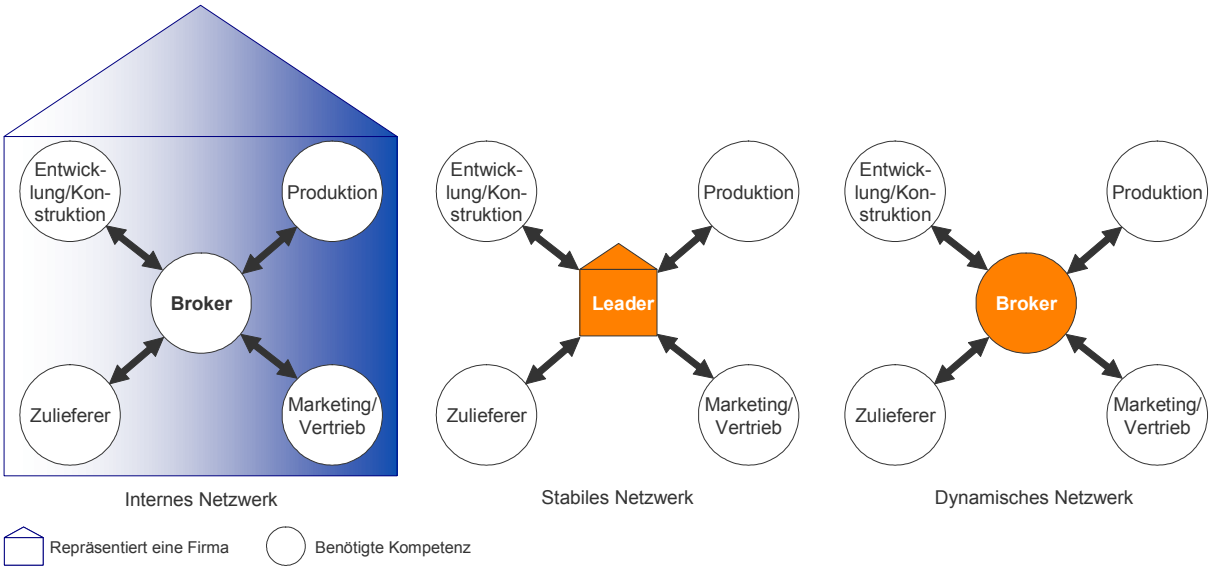


Abbildung 3-11 Netzwerktypen nach Snow (Faisst Wolfgang 1995 – Welche IV-Systeme sollte ein Virtuelles, S. 5)

„Das **interne Netzwerk** verkörpert die dezentralisierte Organisationsstruktur eines Unternehmens, das aus Profit Centern besteht. Im stabilen Netzwerk schart ein führendes Unternehmen zahlreiche Zulieferer um sich, die für einen Großteil der gesamten Wertschöpfung verantwortlich sind. Ebenso kann ein Unternehmen sich für die Befriedigung spezieller Kundenwünsche einen Pool von Firmen halten, die zuvor ausgewählt und zertifiziert werden. Im dynamischen Netzwerk führt der Broker je nach Situation unterschiedliche Partner zusammen. Dies stellt eine extreme Form von Outsourcing betrieblicher Funktionen dar (Beispiel: Galoob Toys). Die oben vorgestellten Netzwerktypen sind Grundlage für VU.“ (Snow, Miles et al. 1992/0 – Managing 21st century network organizations, S. 5–20)

Zusammenfassend sind VU real nicht existierende Unternehmen. Diese VU entstehen durch einen Zusammenschluss verschiedenen Abteilungen und Firmen um eine dedizierte Aufgabe zu lösen oder Produkte auf den Markt zu bringen.

Bei der IMSE findet man die Netzwerktypen internes Netzwerk oder stabiles Netzwerk. Das interne Netzwerk ist vergleichbar mit einem feststehenden Team ohne wechselnde Teammitglieder. Bei einem Team das je nach Bedarf mit Vertretern der verschiedenen Fachdisziplinen zusammengestellt wird, kann es sich um ein stabiles oder dynamisches Netzwerk handeln. Ob es sich um ein stabiles oder dynamisches Netzwerk handelt, hängt von der Art und Weise ab,

wie sich das Team zusammenfindet (Leader oder Broker). Da in der Raumfahrt der Projektleiter das Team zusammenstellt, ist hier das stabile Netzwerk vorherrschend.

Nachdem die Struktur der von virtuellen Unternehmen und Teams definiert und verglichen wurde, können die verschiedenen Ansätze für die Softwarestruktur diskutiert werden.

Struktur von Informationsverarbeitenden Systemen

Faisst (Faisst Wolfgang 1995 – Welche IV-Systeme sollte ein Virtuelles, S. 12) definiert drei Stufen für die Kopplung von IV-Systemen, die die Kommunikation auf unterschiedlichem Niveau und in unterschiedlicher Intensität unterstützen:

Tabelle 3-1 Stufen für die Koppelung von IV-Systemen nach Faisst (Faisst Wolfgang 1995 – Welche IV-Systeme sollte ein Virtuelles, S. 12)

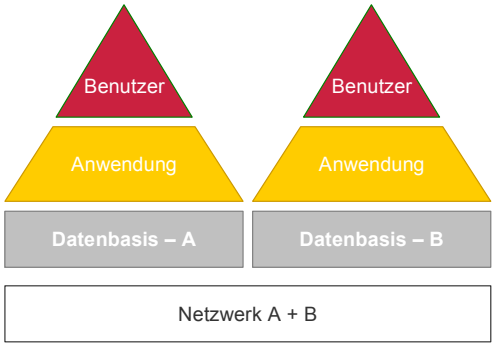


Abbildung 3-12 Applikations-Kommunikation

Die unterste Ebene bei der Koppelung von IV-Systemen bildet die Applikations-Kommunikation über einheitliche Kommunikationsschnittstellen und -standards.

Dieser Ansatz kann mit der verbalen Kommunikation zwischen allen Teammitgliedern verglichen werden.

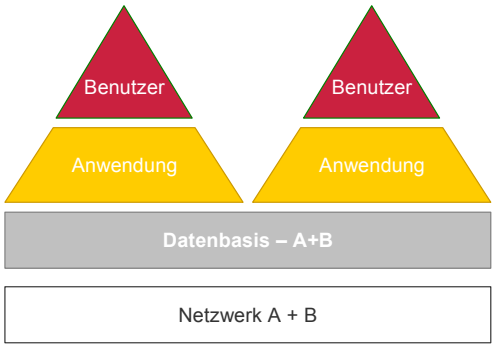


Abbildung 3-13 Datenaustausch

Die nächste höher entwickelte Variante ist der Datenaustausch zum Zugriff auf gemeinsame Datenbestände unter Berücksichtigung der Datenintegrität und -sicherheit. Beim Datenaustausch ist festzulegen, auf welche Daten externe Partner zugreifen dürfen und welche Informationen aus Sicherheitsgründen nur intern zur Verfügung stehen. Die Zugriffsrechte auf die Datenbanken der Partner können in verschiedenen Abstufungen vergeben werden.

Verglichen mit einem Entwicklungsteam würden alle Teammitglieder ihre Daten in einer zentralen Dateiablage auf einem Server abspeichern. Verglichen mit den Ansätzen die es für den integrierten modellbasierten Satellitenentwurf gibt, kann man den NASA- und ESA-Ansatz mit dieser Variante vergleichen.

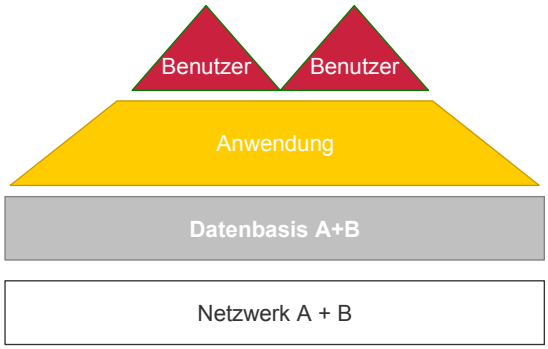


Abbildung 3-14 Application Sharing

Beim Applications-Sharing „werden die Anwendersysteme der Partner so in einer Art und Weise vereint, dass Multi-User-Systeme im Sinne von Groupware entstehen“. Die Kooperationspartner müssen in Abhängigkeit vom jeweiligen Bereich der Zusammenarbeit entscheiden, wie intensiv die Applikationen verflochten werden sollen.

Dieser Ansatz stellt die idealste Lösung für den integrierten modellbasierten Satellitenentwurf da, da hiermit alle Beteiligten direkten Zugriff auf alle Informationen bekommen. In MuSSat (Wilke 2002 – Integrierter modellbasierter Satellitenentwurf, S. 109ff; Quirmbach 2001 – Integration der System- und Kostenentwicklung) wurde diese Tiefe der IV-System Verknüpfung erreicht.

3.6 Zusammenfassung

In diesem Kapitel wurden die verschiedenen Ansätze für Werkzeuge zur Unterstützung des IMSE vorgestellt. Es kann folgender Entwicklungsstand festgestellt werden:

- **Objektbasierte Modellierung**

  Obwohl die Softwareentwicklung gezeigt hat, welche Vorteile objektorientierte Modellierung hat, baut **kein vorgestellter Ansatz** für den integrierten, modellbasierten Satellitenentwurf vollständig auf **objektorientiertes Modellieren** auf.
  MuSSat stellt hier eine Ausnahme dar, da hier im Bereich der Kostenmodelle objektorientiert modelliert wird. Die technische Modellierung ist objektbasiert, d.h. es verzichtet auf Klassen, Vererbung und einer sauberen Instanziierung.

- **Microsoft Excel als Modellierungswerkzeug**

  Alle Ansätze setzen zur Modellierung oder zumindest für den Zugriff auf die Daten Microsoft Excel ein.

- **Graphische Oberfläche**

  Die Stärke von UML und SysML ist ihre graphische Darstellung der Modelle. Daher sollten die Werkzeuge für den IMSE eine graphische Oberfläche bieten. Bis auf MuSSat bietet keiner der anderen vorgestellten Ansätze eine graphische Oberfläche an, die dem Anwender den Aufbau seines Modells darstellt.

- **Zentrale Datenbank für alle Projekte und Optionen**

  Eine gemeinsame Datenbank für alle Projekte erleichtert den Zugriff auf bestehende Daten und somit Wissen. Zudem können darüber auch verschiedene Satellitendesigns leichter miteinander verglichen werden. Obwohl einige Ansätze eine zentrale Datenbank vorsehen, hat nur MuSSat eine integrierte Datenbank für alle Projekte und Optionen.

- **Data-Sharing versus Application-Sharing**

  Faisst schlägt das Application-Sharing als ideale Softwarestruktur bei VU vor. Bis auf MuSSat verwenden alle Ansätze zwar das gleiche Werkzeug für die Modellierung von Satellitensystemen. Dabei wird aber der Zugriff auf die Daten in die Clientsoftware und somit nach Microsoft Excel verschoben. Für ein Application-Sharing wie es Faisst vorschlägt ist dieser Ansatz nicht geeignet, da jeder Benutzer eine andere Benutzerschnittstelle hat. Des Weiteren erschwert der Data-Sharing Ansatz eine Zugriffskontrolle auf die Daten.

Bevor die neue Modellierungsmethodik für Systeme im Rahmen der integrierten, modellbasierten Satellitenentwicklung vorgestellt wird, zeigt das folgende Kapitel eine Übersicht über die Erfahrungen, die der Autor im Rahmen von verschiedenen Projekten gesammelt hat. Dies ist von Bedeutung, da diese Erfahrungen den im Rahmen dieser Arbeit vorgestellten Ansatz entscheidend mit beeinflusst haben.

4 Durchgeführte Projekte in den Bereichen Concurrent Engineering und Concurrent Design Center

Der Autor hat in Rahmen von mehreren Projekten mit der Industrie, durch Praktika und Projekte am Lehrstuhl zahlreiche Erfahrungen im Bereich IMSE gesammelt. Zudem war der Autor an der Definition und der Implementierung von Concurrent Design Centern beteiligt. Darüber hinaus hat der Autor über acht Jahre in verschiedenen Projekten Erfahrungen in der Nutzung von Concurrent Design Centern gesammelt. In diesem Kapitel werden nun diese Erfahrungen des Autors im Bereich Satellitenentwicklung, CE und CDC aufgezeigt. Diese Erfahrungen haben den Autor dazu bewegt, einen neuen Modellierungsansatz für die IMSE zu entwickeln. Tabelle 4-1 bietet einen Überblick über die verschiedenen Projekte und deren zeitlichen Zusammenhang. Im Anschluss an die Tabelle werden die Projekte und Erfahrungen ausführlich beschrieben.

Tabelle 4-1 Übersicht über Tätigkeiten im Rahmen dieser Arbeit in den Bereichen Satellitenentwicklung, Concurrent Engineering und Concurrent Design Center

Projekt / Jahr	Beschreibung
LunarSat 1996-1999	In dem Studentenprojekt LunarSat wurde von ca. 20 Studenten aus Europa ein Mikrosatellit (< 100 kg) mit dem Missionsziel entwickelt, hochauflösenden Bildern der Mondpole zu liefern.
Satellite Design Office 1999-2000	Im Rahmen des Projektes wurde bei EADS-Astrium Friedrichshafen und Ottobrunn eine Software für den integrierten Satellitenentwurf (MuSSat), das in dem CDC der Firma zum Einsatz kommen sollte, entwickelt und implementiert. Gleichzeitig wurde an einem zweiten Standort mit der gleichen Software ein Modell für eine kommerzielle Mission (LEO-1) erzeugt. Gleichzeitig wurde beim Aufbau eines zweiten CDC die Firma beratend untersützt.
Solargenerator 2000	Im Rahmen dieses Projektes wurde das Werkzeug für die Auslegung von Solargeneratoren in MuSSat überführt.
"Concept Design Center" – Workshop 2000-2004	Zur Vermittlung der notwendigen Fähigkeiten für den multidisziplinären Satellitenentwurf wurde am Lehrstuhl ein eigenes CDC aufgebaut. In diesem CDC wurden jährlich ein Mal einwöchige Workshops mit Student(inn)en durchgeführt. Der Workshop diente auch zur Verifikation der am LRT entwickelten Methodiken und Werkzeuge.
BaiCES 2001-2003	Das BaiCES Projekt, gefördert durch die Bayerische Forschungsstiftung (BFS), hat sich mit den Themengebieten: Anforderungs- und Funktionsmodellierung befasst.
Customer Demand Center 2001-2002	Das Customer Demand Center (CDC) wurde bei Fairchild Dornier Company (FDC) aufgebaut. Hierzu wurde mit dem Team des Kunden der Angebotsprozess für Kundenaufträge für das CE angepasst, das CDC definiert und aufgebaut und der neue Prozess eingeführt.
Lunar Exploration Orbiter 2007	Mit einer Gruppe von 13 Student(inn)en wurde im Rahmen des Raumfahrttechnischen Praktikums eine Phase 0 Studie der deutschen Lunar Exploration Orbiter Mission durchgeführt.

4.1 LunarSat

In dem Studentenprojekt LunarSat wurde von ca. 20 Studenten aus ganz Europa ein Mikrosatellit (< 100 kg) mit dem Hauptmissionsziel entwickelt, hochauflösende Bilder der Mondpole zu erzeugen. Mit diesen hochauflösenden Fotos sollten mögliche Gebiete auf dem Mondsüdpol entdeckt werden, die permanent von der Sonne bestrahlt werden und solche die immer im Schatten liegen. Dies ist insofern von Bedeutung, da an den Stellen mit permanenter Sonneneinstrahlung die Energieversorgung mit Solarkollektoren für eine zukünftige bemannte Mondmission aufgebaut werden kann; es würde immer ausreichend Energie zur Verfügung stehen. Darüber hinaus erhoffte man sich in den Bereichen des ewigen Schattens Wasser für die Menschen einer bemannten Mondmission zu finden.

Im Rahmen des Projektes mussten die verschiedensten Fachdisziplinen nicht nur am Standort TU München miteinander kommunizieren und zusammen ein optimales Design erarbeiten, sondern auch die Ergebnisse und das Design mit anderen Standorten koordinieren. Abbildung 4-1 zeigt dies an der geplanten Integration von LunarSat, die an drei Standorten stattfinden sollte. Die Planung sah eine Integration in mehreren Schritten über mehrere Standorte verteil vor. So sollte die Struktur in der Technischen Universität München gefertigt werden. Die Nutzlast wäre dann mit dem oberen Teil der Struktur bei Surrey Satellite Technology Limited in England (SSTL) integriert worden. Das Antriebssystem wäre von der DASA mit der Zentralstruktur vereint worden. Die Gesamtintegration des Satelliten hätte dann wieder SSTL übernommen.

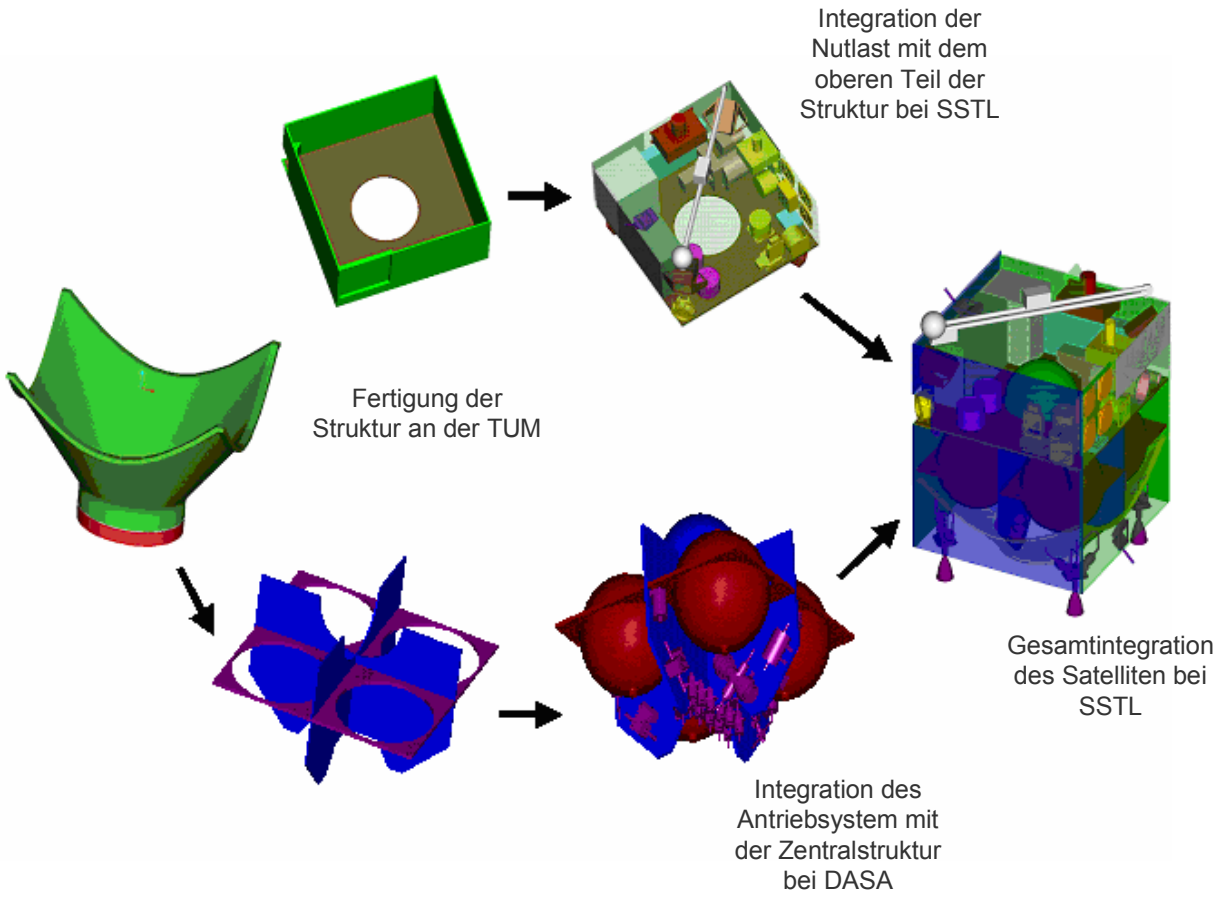


Abbildung 4-1 Geplante Integration von LunarSat mit mehreren Partnern (Eckart, Kesselmann et al. 1999 – LunarSat AIV, S. 4)

Erfahrungen

Im Rahmen der Auslegung des Antriebssystems für den Satelliten konnten die ersten Erfahrungen und die Bedeutung eines integrierten, modellbasierten Satellitenentwurfs gesammelt werden. Im Rahmen von LunarSat wurde kein integriertes Modell für den Satellitenentwurf verwendet. Deshalb konnten verschiedene Optionen nur auf Teilsystemebene komplett durchgerechnet werden. Ein Beispiel hierfür ist die Diskussion, ob ein elektrischer oder chemischer Antrieb für den Transfer zwischen der Erde und dem Mond verwendet werden sollte. Die Analysen für den elektrischen Antrieb haben zwar die Energieversorgung berücksichtig, aber die Gesamtkonfiguration wurde nicht untersucht.

Dies hat gezeigt wie notwendig

- **ein zentrales Datenmodell des Satelliten**
- **in einem Modellierungswerkzeug, mit dem verschiedene Optionen schnell untersucht werden können,**

ist. Schließlich wurde auf Grund des Entwicklungsstandes von elektrischen Antrieben gegen diese Option entschieden.

4.2 Satellite Design Office

Bei dem EADS-Astrium Standort Friedrichshafen wurde das Satellite Design Office (SDO) aufgebaut, das erste europäische CDC in der Raumfahrtindustrie. Der Lehrstuhl für Raumfahrttechnik entwickelte im Rahmen dieses Projektes die Software MuSSat, die als zentrales Entwicklungswerkzeug im SDO vorgesehen war, auf Basis des MuSSys - Ansatzes von Negele weiter (Quirmbach 2001 – Integration der System- und Kostenentwicklung; Wilke 2002 – Integrierter modellbasierter Satellitenentwurf). In Zusammenarbeit mit dem Standort Friedrichshafen wurden die Anforderungen an die Software definiert und von dem Entwicklungsteam am LRT umgesetzt. Darüber hinaus wurde mit dem Designteam der Firma Astrium am Standort Friedrichshafen ein erstes Modell zur Unterstützung des integrierten, modellbasierten Satellitenentwurfs erarbeitet.

Im Rahmen dieses Projektes wurde zu dem am Standort Ottobrunn der Einsatz von CE, CDC und MuSSat den Verantwortlichen vorgestellt. Dazu wurde ein Modell der LEO-1 Satelliten, die als Relay-Satelliten für Kurzmitteilungen fungieren sollten, modelliert. Das Modell umfasste alle Teilsysteme und die notwendigen Missionselemente (Bodenstation, Orbits, Trägersystem). Gleichzeitig war man bei dem Aufbau des SDO an diesem zweiten Standort beratend tätig.

Das LEO-1 Projekt wurde nie realisiert, da der Kunde keine Finanzierung für das Projekt von Investoren bekommen hatte. Zu dieser Zeit gingen die Satellitenkonstellationen Iridium und Globalstar in Konkurs, wodurch die Kapitalgeber für neue Konstellationen sehr zurückhaltend waren.

Erfahrungen

Bei der Arbeit mit dem Werkzeug auf der einen Seite und dem späteren Anwender auf der anderen Seite hat es sich gezeigt, dass folgende Rahmenbedingungen notwendig sind:

- **Schulung**

  Bei dem Einsatz von einer neuen komplexen Software ist eine Schulung der Benutzer unabdingbar. Das gleiche gilt auch für den Prozess, mit dem die Benutzer (das Team) in einem CDC arbeiten.

- **Vorbereitung von Modellen**

  Da im Rahmen der CDC-Arbeit das Design ausgelegt, analysiert und diskutiert werden soll, müssen die Modelle vorher extra erzeugt werden. Dieser Aufwand darf nicht unterschätzt werden und muss in Zusammenarbeit mit dem Team erfolgen, da sonst Verständnisprobleme auf Seiten des Entwicklers und des späteren Benutzers vorliegen. Für die Modellerzeugung muss der spätere Benutzer wissen, welche Auslegungsrechnungen und Analysen für sein Teilsystem notwendig sind.

4.3 Solargenerator

Im Rahmen des Solargenerator Projektes wurde in Zusammenarbeit mit Mitarbeitern des Auftraggebers das im Hause entwickelte Werkzeug für die Auslegung von Solargeneratoren auf MuSSat umgesetzt. Dies war insofern möglich, da das vorhandene Werkzeug in Microsoft Excel implementiert war. Durch die Umstellung sollten die Mitarbeiter bei der Angebotserstellung unterstützt werden, damit die Angebote einfacher und schneller angefertigt und auf Kundenänderungswünsche schneller hätte reagiert werden können.

Erfahrung

Obwohl der Transfer des Werkzeuges von einer integrierten Lösung in eine für MuSSat angepasste Lösung nicht vollständig abgeschlossen werden konnte, brachte das Teilprojekt ein **besseres Verständnis des** Solargenerator-**Modells** und der **Zusammenhänge** im Modell für

die Mitarbeiter. Eine wichtige **Anforderung**, die im Rahmen dieses Projektes an MuSSat gestellt wurde, aber von MuSSat nicht unterstützt wird, war eine **objektorientierte Modellierung** in der technischen Auslegung.

4.4 „Concept Design Center“ – Workshop

Das Space System Concept Center (S_2C_2) am Lehrstuhl für Raumfahrttechnik war eines der ersten Design Center für den Satellitenentwurf in Deutschland. Es wurde hauptsächlich vom Autor am Lehrstuhl eingerichtet, um Studenten diesen neuen innovativen Ansatz des Satellitendesigns, speziell in der frühen Phase eines Projektes näher zu bringen und gleichzeitig in Teamarbeit und Satellitendesign einzuführen. Zudem erlaubt das S_2C_2, die in Zusammenarbeit mit der Industrie gesammelt Erfahrungen in neue Ideen umzusetzen und zu testen.

Zu diesem Zwecke wurde am Lehrstuhl jedes Jahr ein „Concept Design Center“-Workshop mit Teams von ca. acht Studenten durchgeführt. Der Workshop dauerte eine Woche. Innerhalb dieser Zeit wurde den Studenten eine Missionsaufgabe gestellt (z.B. Satellitenkonstellation für Kommunikationsnetzwerke), die Anforderungen abgeleitet und eine Phase 0 Studie von den Studenten durchgeführt. Der dabei verwendete Zeitplan ist in Abbildung 4-2 dargestellt. Hervorzuheben ist hierbei, dass bis zum Freitagvormittag ein Designkonzept erarbeitet wurde, und verschiedene Teilsystemoptionen analysiert werden mussten.

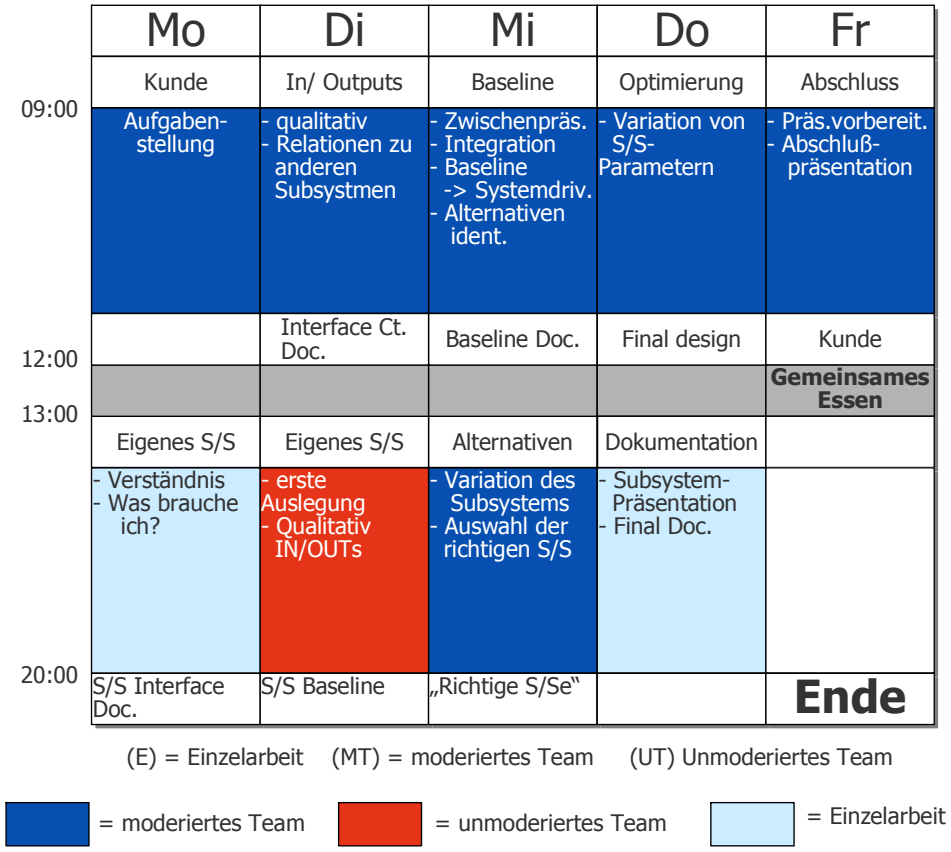


Abbildung 4-2 Prozessablauf während eines Studentenworkshops

Für die Auslegung und Analyse von verschiedenen Designoptionen, wurde ein Modell in MuSSat implementiert und den Studenten zur Verfügung gestellt. Die Studenten hatten vor dem Workshop eine eintägige Schulung in MuSSat und konnten sich danach noch einen Tag mit den vorgefertigten Modellen vertraut machen.

Bei den Workshops arbeiteten die Studenten den ersten Tag ohne Moderation an ihren Teilsystemen. Am nächsten Tag wurden die Schnittstellen zwischen den Teilsystemen (Interface Doc. in Abbildung 4-2) abgeglichen. Der Abgleich der Schnittstellen zeigte eine deutliche Inkonsistenzen im Modell. Die Inkonsistenz entstand dadurch, dass die Teilnehmer nur ihr eigenes Teilsystem betrachteten, aber sich nicht mit anderen Teilnehmern absprachen. Diese Probleme zeigten den Teilnehmern deutlich wie wichtig es ist, bereits während der Modellerstellung miteinander zu reden und Ergebnisse abzustimmen. Um eine zielgerichtete Diskussion im Team zu gewährleisten, war es notwendig, dass ein Moderator durch die Phasen der Teamarbeit führt.

Die folgenden Abbildungen zeigen die geometrische Auslegung einiger Satelliten, die während der „Concept Design Center“ Workshops angefertigt wurden. Diese Modelle wurden direkt aus den Auslegungsmodellen in MuSSat erzeugt und halfen den Studenten ihre Konfiguration besser zu verstehen. Es hat sich gezeigt, dass die drei 3D-Darstellung eines Satellitenentwurfs für den erfolgreichen modellbasierten Entwurf unerlässlich ist. Dabei sollte wie im Falle des „Concept Design Center“ Workshops eine direkte Verbindung zwischen dem Auslegungsmodell und dem 3D-Modell bestehen. Nur dadurch können die Änderungen unmittelbar visualisiert werden.

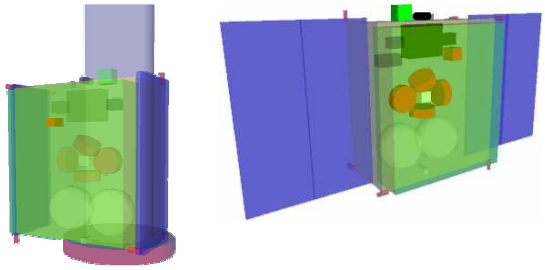


Abbildung 4-3 Kommunikationssatellit auf der Trägerstruktur (links) und mit aufgefalteten Solarzellenflächen (rechts) als Ergebnis eines Workshops

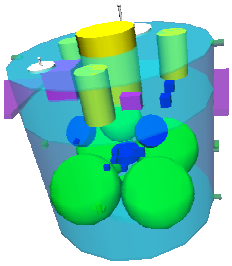


Abbildung 4-4 Erdbeobachtungssatellit ohne Solarzellenflächen als Ergebnis eines Workshops

4.5 BaiCES

Das Projekt Bavarian Center of Excellence in Satellite Constellation (BaiCES) war hervorgegangen aus dem SDO-Vorgängerprojekt mit der Industrie und wurde von der Bayerischen Forschungsstiftung gefördert. Im Rahmen dieses Projektes wurden verschiedene Teilsystemmodelle (Kommunikation und Antrieb) für MuSSat entwickelt aber auch die Anwendung MuSSat um neue Funktionalitäten für die Abbildung der funktionalen Beschreibung und funktionalen Anforderungen erweitert.

Ziel der neuen Funktionalitäten in MuSSat war es, die Zusammenhänge zwischen Anforderungen, funktionale Beschreibung und dem technischen Modell abzubilden. Da bereits die Zusammenhänge zwischen dem technischen Modell (Wilke 2002 – Integrierter modellbasier-

ter Satellitenentwurf) und den Kosten im MuSSat implementiert waren (Quirmbach 2001 – Integration der System- und Kostenentwicklung), konnte mit der erweiterten Funktionalität der **gesamte Zusammenhang** beginnend bei den **Anforderung bis zu den Kosten in einem Modell** dargestellt werden. Dazu wurden verschiedene Ansätze für die Funktionsmodellierung untersucht und der Ansatz von Otto und Wood (Otto, Wood 2001 – Product design) als Grundlage für die Modellierungsmethodik ausgewählt und implementiert. Daneben wurde eine relativ einfache Abbildung von Anforderungen implementiert. Der Grund für die vereinfachte Anforderungsmodellierung lag in dem Umstand, dass qualitativ hochwertige Spezialwerkzeuge für das Anforderungsmanagement existieren und der Projektpartner eines dieser Werkzeuge bereits im Einsatz hatte. Es wurde daher eine Schnittstelle für den Austausch der notwendigen Informationen implementiert.

Abbildung 4-5 zeigt die Programmstruktur von MuSSat wie sie nach Ende des Projektes BaiCES vorhanden war. Die Modellierung der Objektstruktur, Projektstruktur und Kostenstellen war bereits von Wilke und Quirmbach realisiert worden. Darüber hinaus gab es schon die Analyse und Bewertung von Systemdesigns. Bei der Bewertung konnten nur Optionen eines einzigen Projektes verglichen werden. Im Rahmen des BaiCES Projektes wurde die Modellierung um die Funktionsstrukturen und der Anforderungen von Systemen erweitert. Ein wichtiger Punkt hierbei war die Verknüpfung dieser Systembeschreibungen untereinander und mit den bestehenden Teilmodellen wie Objektstruktur und Projektstruktur. Die in der Abbildung dargestellten Komponentendatenbank war nicht Teil von MuSSat selber, sondern war in den Excel Dateien abgelegt.

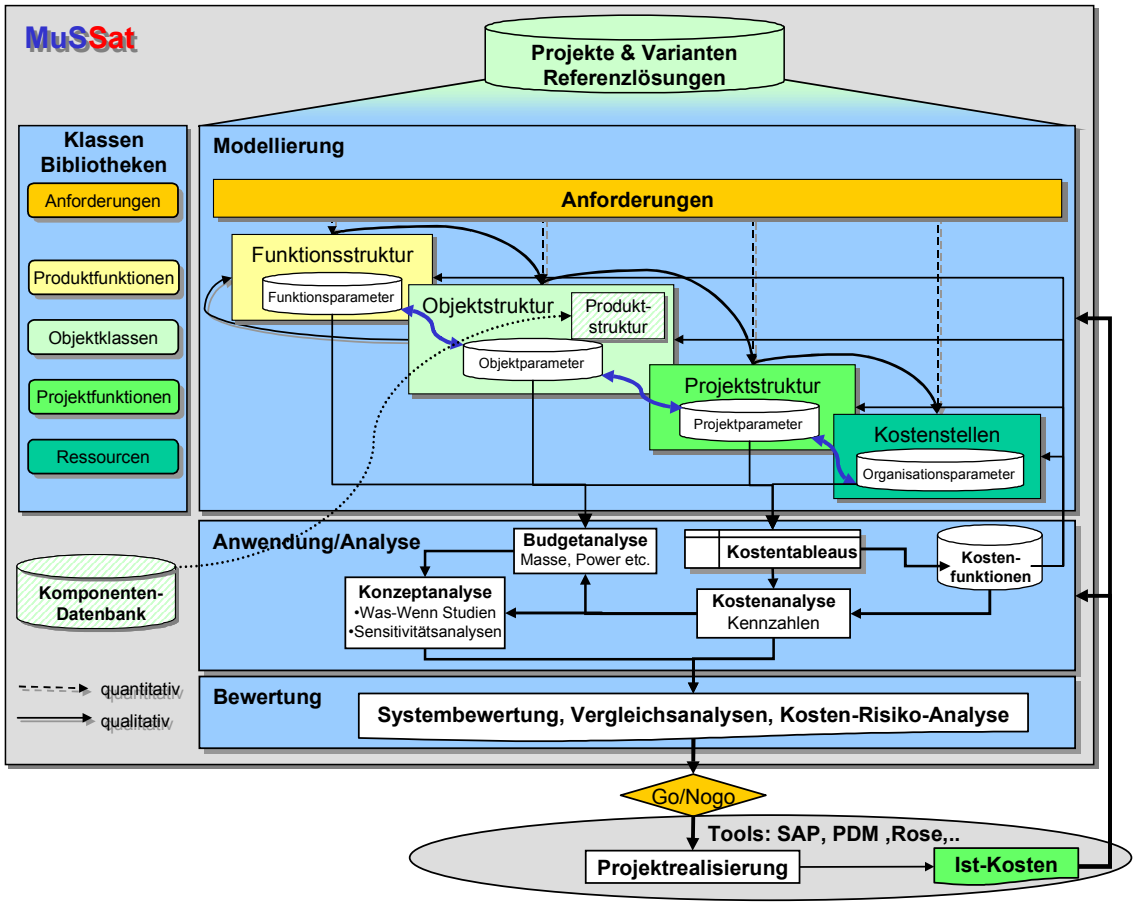


Abbildung 4-5 Programmaufbau von MuSSat nach Abschluss des Projektes BaiCES (Wilke, Quirmbach et al. 1999 – Modell der integrierten System)

Erfahrung

Bei der Entwicklung und Implementierung der neuen Funktionalitäten sind folgende Nachteile mit dem bisherigen Ansatz basierend auf MuSSat deutlich geworden:

- **Getrenntes Datenmodell (Technik und Kosten) trotz der Zielsetzung eines integrierten Satellitenmodells**

  MuSSat erlaubt es, Satellitensysteme durch Kosten und eine technische Beschreibung integriert in einem Modell abzubilden. Obwohl beide Teile des Modells auf die gleiche Elementdefinition zugreifen und beide miteinander verknüpft werden können, sind diese in unterschiedlichen Datenmodellen abgespeichert. Durch diese Trennung des Datenmodells wird der Entwickler

(Modell und Applikation MuSSat) in dem Umgang mit der Software eingeschränkt.

- **Gemischte objektorientierte und objektbasierte Modellierung**

  Bei der Modellierung des Kostenmodells war es bereits möglich, objektorientiert einen Projektstrukturplan zur Kostenberechnung zu modellieren. Im Detail war es möglich Prozessobjekte im PSP von selbst definierten Klassen abzuleiten. Dagegen war die Modellierung des technischen Modells zwar objektbasiert, aber es gab keine saubere Kapselung der Objekte in Klassen. Somit mussten unterschiedliche Modellierungsweisen bei der Erstellung der Satellitenmodelle von dem Benutzer berücksichtigt werden.

- **Keine Kapselung der Objekte auf Projektebene**

  Da nur die Werte der Objekte auf Projektebene gekapselt waren, hatten Änderungen der abstrakten Informationen (Attribute) eines Objektes Auswirkungen auf alle Projekte in dem es verwendet wurde. Auch auf solche Projekte, die bereits abgeschlossen waren. Zudem konnte ein Objekt in einem Projekt nur ein Mal verwendet werden, so dass eine einfache Wiederverwendung von Objekten nicht möglich war.

4.6 Fairchild Dornier Company – Customer Demand Center

Die Firma Fairchild Dornier Company (FDC) stellte bis Sommer 2005 am Standort Oberpfaffenhofen bei München zweistrahlige Regionalflugzeuge her. Das bekannteste Flugzeug von FDC war die Do-328, welches zu Beginn als Zwei-Propeller Maschine und später dann als zweistrahliger Düsenjet angeboten wurde.

FDC stellte im Bereich der Kundenanfragen und -änderungswünsche Verbesserungspotenzial in ihren Prozessen fest. Diese Anfragen umfassten die für jede Fluglinie anders zu gestaltende Inneneinrichtung wie Sitzreihen, Küchen- und Toiletten-Aufteilung bis hin zu Sonderanfertigungen für private Kunden. Gleichzeitig mussten Technologieentwicklungen zur Verbesserung des Gesamtproduktes durchgeführt werden. All diese Änderungen an das Grundflugzeug wurden als Request For X (RFX) bezeichnet.

Der RFX ist somit ein Änderungsprozess für ein bestehendes Produkt und keine Produktneuentwicklung. Trotzdem sind die gleichen Vorgehensweisen wie bei einer Neuentwicklung anzusetzen. Hinzukommt, dass alle betroffenen Systeme, Zeichnungen und Dokumente identifiziert und angepasst werden müssen.

Das Ziel des Projektes war es, den gesamten RFX Prozess auf CE in Verbindung mit einem CDC zu überführen. Das CDC wurde bei FDC Customer Demand Center (CDC) genannt. Im Rahmen der Tätigkeiten wurden die bestehenden Prozesse, die den Bereich RFX abdeckten dokumentiert und gemeinsam mit dem FDC-Team auf Stärken und Schwächen analysiert. Anschließend daran wurden mit dem Team die Prozesse in einen neuen Prozess überführt, wobei darauf geachtet wurde, die Stärken beizubehalten.

Aufbauend auf die Prozessdefinition wurde das CDC spezifiziert, implementiert und der neue Prozess von dem Team „gelebt“.

Die durch die Umstellung des RFX Prozesses gewonnenen Verbesserungen wurde anhand der zeitlichen Dauer der verschiedenen RFX vor und nach der Implementierung dokumentiert. Da aber nicht jedes Jahr zur gleichen Zeit der gleiche RFX initiiert wird, kann man nur eine Aussage über die durchschnittliche Verbesserung treffen. Diese ist in Abbildung 4-6 dokumentiert.

Erfahrung

Dieses Projekt hat gezeigt wie wichtig es ist, den Entwicklungsprozess mit dem Team, das diesen Prozess später umsetzen soll, anzupassen. Nur dadurch identifiziert sich das Team mit dem neuen Prozess und setzt diesen auch erfolgreich um.

Im Rahmen dieses Projektes wurden die drei Bereiche Team, Prozess und Infrastruktur verbessert. Die eingesetzten Werkzeuge wurden nicht verändert. Es wurde aber eine zentrale Datenbank zur Überwachung des Status bei verschieden RFX entwickelt und implementiert.

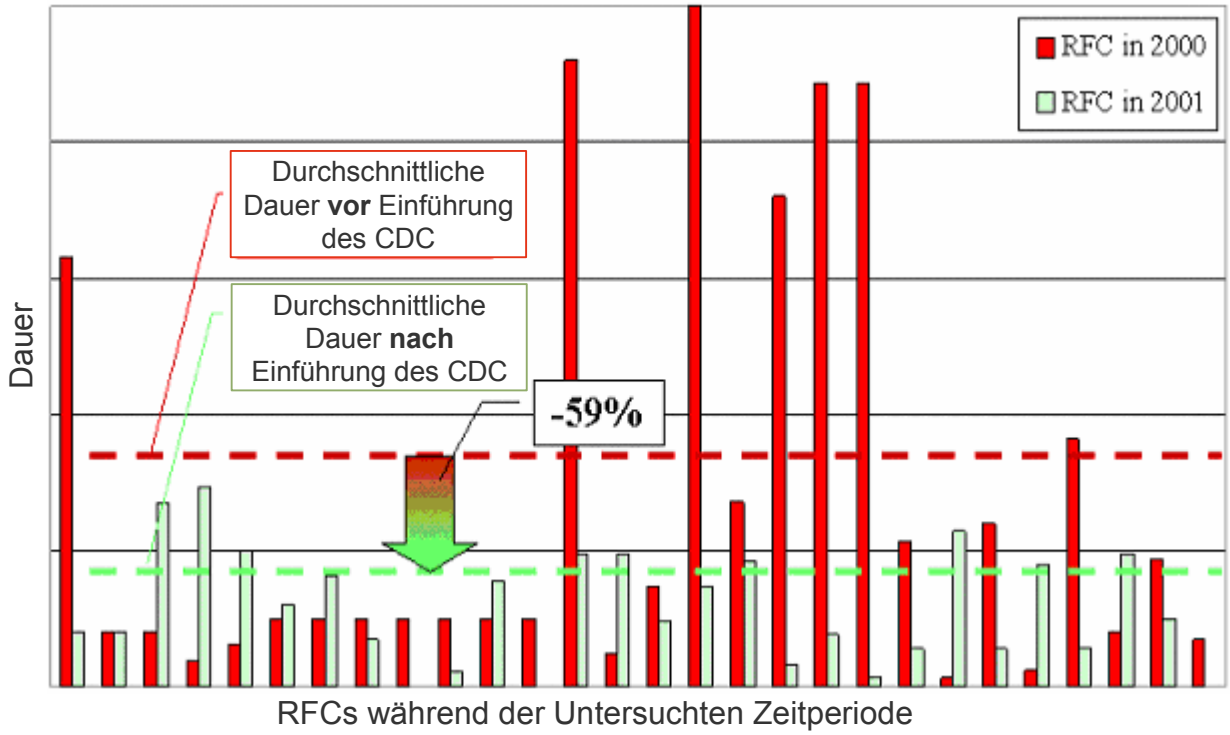


Abbildung 4-6 Durchschnittliche Verbesserung der Durchlaufzeit für RFC während der untersuchten Zweitperiode. Eine quantitative Aussage über die Dauer ist von der Firma nicht genehmigt worden.(Finkel S. 2002 – Design Centers –Transferring Experiences, S. 6)

5 Die neue Modellierungsmethode für die simultane Systementwicklung

Die Erfahrungen aus Kapitel 4 haben einen Verbesserungsbedarf bei den derzeit existierenden Modellierungswerkzeugen gezeigt. Demgegenüber fehlen bei UML oder SysML die notwendigen Funktionalitäten für die IMSE. Der Autor hat im Rahmen dieser Arbeit eine neue Modellierungsmethode entwickelt. Die Methode besteht aus einem **Systemmodell**, einer **graphischen Darstellung** des Modells**,** einer **Systemarchitektur** und einem **Datenmodell** zur Speicherung der Satellitendaten (siehe Abbildung 5-1). Die Systemarchitektur definiert dabei das Zusammenspiel zwischen Clients und Servern sowie eine Sammlung von Funktionalitäten die zwingend notwendig sind. Die gesamte Methode, im folgenden als integrierte Systemmodellierung (ISM) bezeichnet, vereint Elemente aus UML, SysML und dem Münchner-Ansatz und wird auf den folgenden Seiten erläutert.

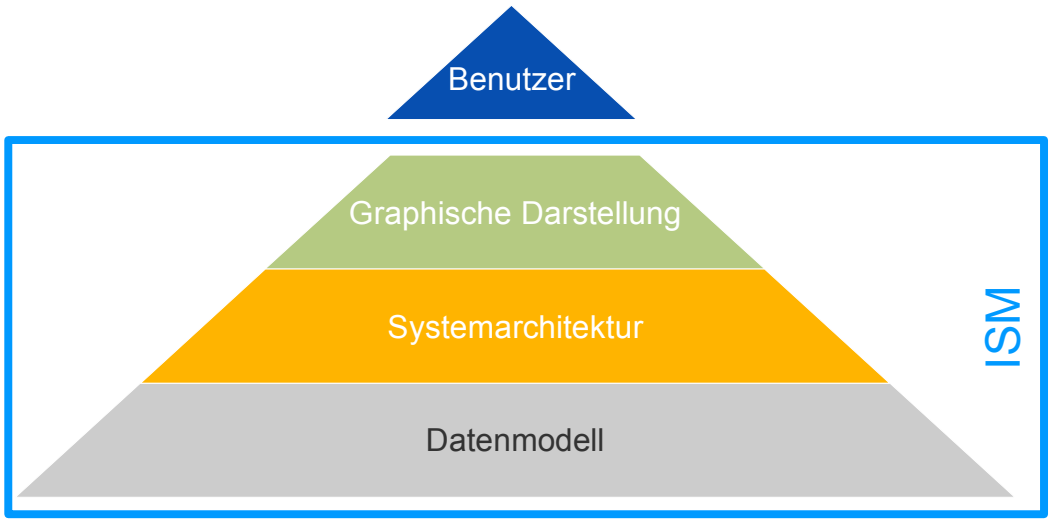


Abbildung 5-1 Die drei Ebenen der ISM

Zur Verifikation des ISM Ansatzes wurde die Software (v)Sys-ed (virtuelles System – Editor, siehe 5.8) von dem Autor entwickelt. Diese Software, bestehend aus einem zentralen Server und beliebige Clients für die Fachdisziplinen, wurde bereits im Rahmen von mehreren Hochschulpraktika und Projekten am LRT erfolgreich eingesetzt.

Ein wichtiges Merkmal an dem ISM-Ansatz ist die Speicherung der Daten in einem zentralen Datenmodell. Das Datenmodell erlaubt **mehrere Projekte** in einer einzigen Datenbank zu speichern. Alle Projekte, die mit ISM abgebildet werden und somit in der gleichen Datendatei vorliegen, verwenden dabei die **gleichen Datentypen und Klassen**. Dadurch kann ein Standard für die Modellierung innerhalb einer Firma definiert werden. Zudem werden nicht die Regeln zum Aufbau der Modelle in der Datenbank abgelegt, sondern der **aktuelle Zustand** des Systemmodells bestehend aus allen Objekten, Werten und Relationen.

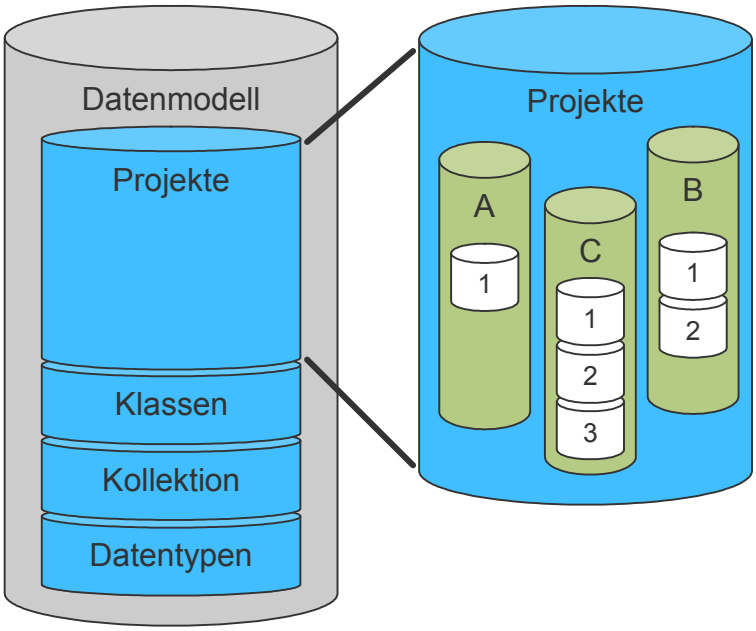


Abbildung 5-2 Aufbau des Datenmodells bestehend aus den Datentypen, Kollektionen, Klassen, Projekte (A-C) und Optionen (1-3)

Die Projekte (A, B, C) selber sind wiederum in Optionen (1, 2, 3) untergliedert. Die Optionen stellen dabei verschiedene Designlösungen für das jeweilige Problem dar.

5.1 Element-, Klassen- und Objektdefinition

5.1.1 Allgemeine Definition

Bei der objektorientierten Modellierung werden die in den Modellen verwendeten Objekte durch eine Instanziierung einer abstrakten Beschreibung erzeugt. Die abstrakte Beschreibung wird als Klasse bezeichnet. Des Weiteren ermöglicht die objektorientierte Modellierung auf Klassenebene eine Vererbung von Klassenattributen. Die Klassenattribute sind die Schnittstellen (Input und Outputs), die Parameter sowie die Methoden einer Klasse. Die Inputs, Parameter und Outputs werden durch einen Datentyp definiert. Die Definition von Datentypen ist im nachfolgenden Kapitel 5.2 beschrieben.

Die Schnittstellen (Inputs und Outputs) dienen der Kommunikation mit anderen Objekten. Dazu werden die Inputs und Outputs über Flussrelationen verbunden. Die Outputwerte können nur von dem Objekt selber durch Methoden verändert werden. Genauso können die In-

putwerte nur durch das eigene Objekt mit dem auf dem Output des anderen Objektes anliegenden Wert gesetzt werden. Somit sind die Inputs und Outputs von anderen Objekten sichtbar aber nicht veränderbar.

Methoden definieren die Abhängigkeiten zwischen den Inputs, Parametern und Outputs. Die Methoden einer Klasse / eines Objektes sind in einem Funktionsblock zusammengefasst. Diese Kapselung erlaubt es, die Abhängigkeiten durch eine beliebige Modellierungsmethode zu beschreiben (siehe Abbildung 5-3). Dies kann sowohl eine einfache Textbeschreibung, Matrix (siehe Walther – Systemtechnisches Verfahren zur Bestimmung, S. 59–71), externe Software wie Microsoft Excel, beliebig andere ausführbare Dateien oder auch ein Diagramm (Aktivitätsdiagramm, State-Machine, o. ä.) sein. Für eine dynamische Modellierung eines Systems oder für die Auslegungsberechnung sind die ersten beiden Möglichkeiten nicht geeignet. Durch die Kapselung der Methoden in Funktionsblöcke können die Methoden nur bei der Erzeugung einer Kindklasse vererbt bzw. einzelne Methoden, wie in C++ möglich, nicht mehr überladen werden.

Im Rahmen der Verifikation von ISM wurden die ersten vier möglichen Funktionsmodellierungsmöglichkeiten aus Abbildung 5-3 implementiert. Da alle Modellierungsmöglichkeiten die gleiche Abhängigkeiten zwischen Inputs, Parametern und Outputs auf unterschiedliche Art und Weise abbilden, ist, soweit möglich, auf eine Konsistenz der Modelle zu achten[12].

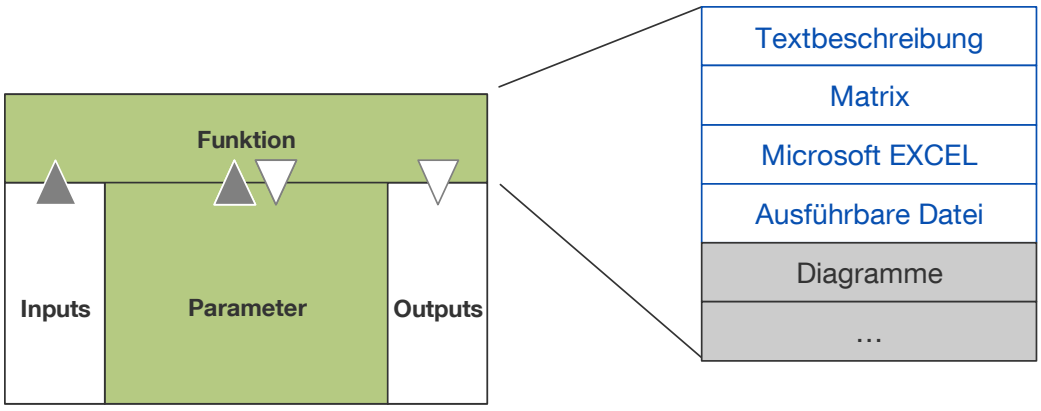


Abbildung 5-3 Datenmodel einer Klasse oder eines Objektes bestehend aus den Schnittstellen (Input und Outputs), den Parametern und der Funktion zur Abbildung der Abhängigkeiten

[12] Bei der Implementierung von ISM in (v)Sys-ed wird die Matrix bei Änderungen der Microsoft EXCEL Modelle automatisch aktualisiert.

Durch die Vererbung, Ableitung einer Klasse von einer bestehenden Klasse, können Gemeinsamkeiten bei Klassen in einer gemeinsamen Vaterklasse definiert werden. Dies erlaubt eine strukturierte und klare Modellierung der benötigten Klassen für die Abbildung eines Gesamtsystems.

Beispiel:

Bei der Modellierung von Komponenten (Triebwerk, On-Board Computer, Struktur, Solarzellen, usw.) können die Informationen über Masse und Größe in einer Klasse „Komponenten" definiert werden. Die spezialisierten Klassen für die Modellierung von Triebwerken, Computer, Struktur oder Solarzellen müssen nur noch von dieser Klasse „Komponenten" abgeleitet werden und besitzen automatisch die Attribute Masse und Größe.

Da in machen Fällen eine Vererbung der Attribute nicht gewünscht ist, gibt es in ISM Sichtbarkeiten zu denen in UML. Sichtbarkeit definiert die Sichtbarkeits- und Zugriffsrestriktionen, der ein Attribut unterworfen ist, d.h. welche anderen Objekte oder Klassen das Attribut "sehen" können beziehungsweise seinen Wert lesen und überschreiben dürfen. Vier verschiedene Sichtbarkeitsmodi werden in UML unterschieden in (Jeckle 2004 – UML 2 glasklar, S. 46):

- **public**

  Jede andere Systemkomponente hat uneingeschränkten Zugriff.

- **private**

  Nur Ausprägungen der das Attribut beherbergenden Klasse dürfen zugreifen.

- **protected**

  Das Attribut ist nur in der definierenden und jenen Klassen sicht- und zugreifbar, die als Spezialisierungen von ihr definiert sind.

- **package**

  Das Attribut ist für alle Klassen, die sich im selben Paket wie die definierende Klasse befinden, sicht- und zugreifbar.

Abgeleitet aus diesen vier Modi werden, abhängig ob es sich um einen Input, Parameter oder Output handelt, drei Sichtbarkeiten und Zugriffsmodi für ISM definiert (siehe Tabelle 5-1).

Zur Vereinfachung der wortreichen Sichtbarkeitsspezifikationen wird in ISM wie in UML für jede Sichtbarkeitsvariante ein dem Attribut vorangestelltes Symbolkürzel verwendet.

Tabelle 5-1 Sichtbarkeiten und Zugriffsrechte der Attribute in ISM

Attribut	Bezeichnung (Symbol)	Sichtbarkeit und Zugriffsrecht
Input / Parameter	`private (-)`	Der Input oder Parameter **wird nicht** an abgeleitete Klassen **vererbt**. Der Wert kann nur von der Klasse/dem Objekt selber verändert werden.
Parameter	`private constant (-)`	Der Parameter **wird nicht** an abgeleitete Klassen **vererbt**. Der Wert des Parameters kann nicht verändert werden (z.B. Gravitationskonstante). Der Wert für den Parameter muss daher während der Initialisierung des Objektes definiert sein.
Input / Parameter	`protected (#)`	Der Input oder Parameter **wird** an abgeleitete Klassen **vererbt**. Der Wert kann nur von der Klasse/dem Objekt selber verändert werden.
Parameter	`protected constant (#)`	Der Parameter **wird** an abgeleitete Klassen **vererbt**. Der Wert des Parameters kann nicht verändert werden (z.B. Gravitationskonstante). Der Wert für den Parameter muss daher während der Initialisierung des Objektes definiert sein.

Output	`local public (-)`	Der Output **wird nicht** an abgeleitete Klassen **vererbt**. Der Output kann von allen anderen Klassen/Objekten gelesen werden.
Output	`public (+)`	Der Output **wird** an abgeleitete Klassen vererbt. Der Output kann von allen anderen Klassen/ Objekten gelesen werden.

5.1.2 Differenzierung Element, Klasse und Objekt

Objekte sind reine Instanzen von Klassen und besitzen dadurch konkrete Werte. Durch den objektorientierten Ansatz können mehrere Objekte der gleichen Klasse nicht nur einfach erzeugt werden sondern auch Änderungen bei allen Objekten durch Veränderung der Klasse erzwungen werden.

Neben den Begriffen Klasse und Objekt soll hier noch der Begriff Element im Rahmen von ISM definiert werden. Abbildung 5-4 zeigt diesen Zusammenhang. Die Bezeichnung Element bezieht sich sowohl auf Klassen als auch auf Objekte und wird dann verwendet, wenn sich eine Aussage sowohl auf Klassen als auch auf Objekte bezieht.

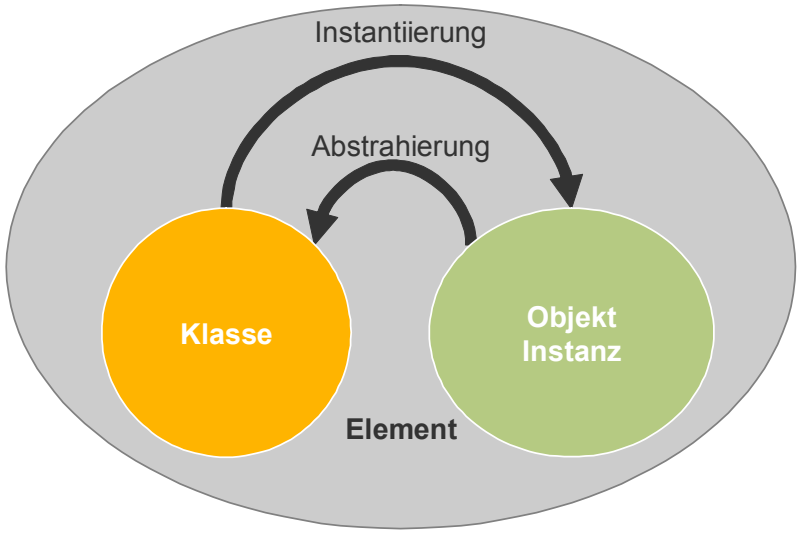


Abbildung 5-4 Zusammenhang zwischen Klasse, Objekt und Element

5.1.3 Initialisierung von Objekten

Im Rahmen von ISM gibt es zwei wichtige Merkmale bezüglich der Definition von Klassen und Objekten. Die erste Besonderheit bezieht sich auf die Definition der Initialisierung von Objekten. Dies wird in UML durch die Object Constrain Language (OCL) abgebildet oder direkt in der Konstruktor– Methode einer Klasse implementiert. In Gegensatz dazu werden bei ISM auf Klassenebene den Attributen Initialisierungswerte zugewiesen. Bei der Erzeugung eines neuen Objektes werden diese Initialisierungswerte den Objekten zugewiesen.

Die Initialisierungswerte können zudem zum Testen der Klassenmethoden verwendet werden. Dadurch können die Funktionsblöcke bereits einzeln auf Klassenebene getestet und verifiziert werden.

5.1.4 Schattenklassen

ISM bietet neben der Definition von Initialisierungswerten für Objekte ein zweites wichtiges Unterscheidungsmerkmal zu UML, SysML oder auch dem Münchner-Ansatz. Dieses Merkmal wird in ISM als Schattenklasse bezeichnet. Obwohl jedes Objekt ursprünglich von einer Klasse aus dem Klassenbaum abgeleitet wird, besitzt es eine eigene Klasse, die nicht im Klassenbaum definiert ist.

Es gibt drei Gründe für die Definition von Schattenklassen. Zum Einen ist es durch die **Speicherung der Daten** in einer Datenbank bedingt. Die zweite Ursache für die Schattenklassen liegt in der Anforderung den **Designzustand einfrieren** zu können. In diesem Fall dürfen keine Änderungen mehr an den Optionen und somit Objekten durchgeführt werden. Dazu müssen die Objekte von den Klassen unabhängig werden. Ein weiterer wichtiger Grund liegt in der Anforderung, Benutzern das **Anpassen der Objekte auf Objektebene** zu erlauben, ohne dabei die Klasse zu verändern. Diese Anforderung stammt aus den Erfahrungen des Autors, dass Ingenieure immer noch Schwierigkeiten mit der objektorientierten Modellierung haben. Durch den Ansatz der Schattenklassen wird eine objektorientierte Modellierung innerhalb der Modellierung beibehalten, dem Anwender aber eine klassische objektbasierte bzw. funktionale Modellierung vorgetäuscht. Abbildung 5-5 verdeutlich den Zusammenhang zwischen Klasse, Objekt und Schattenklasse.

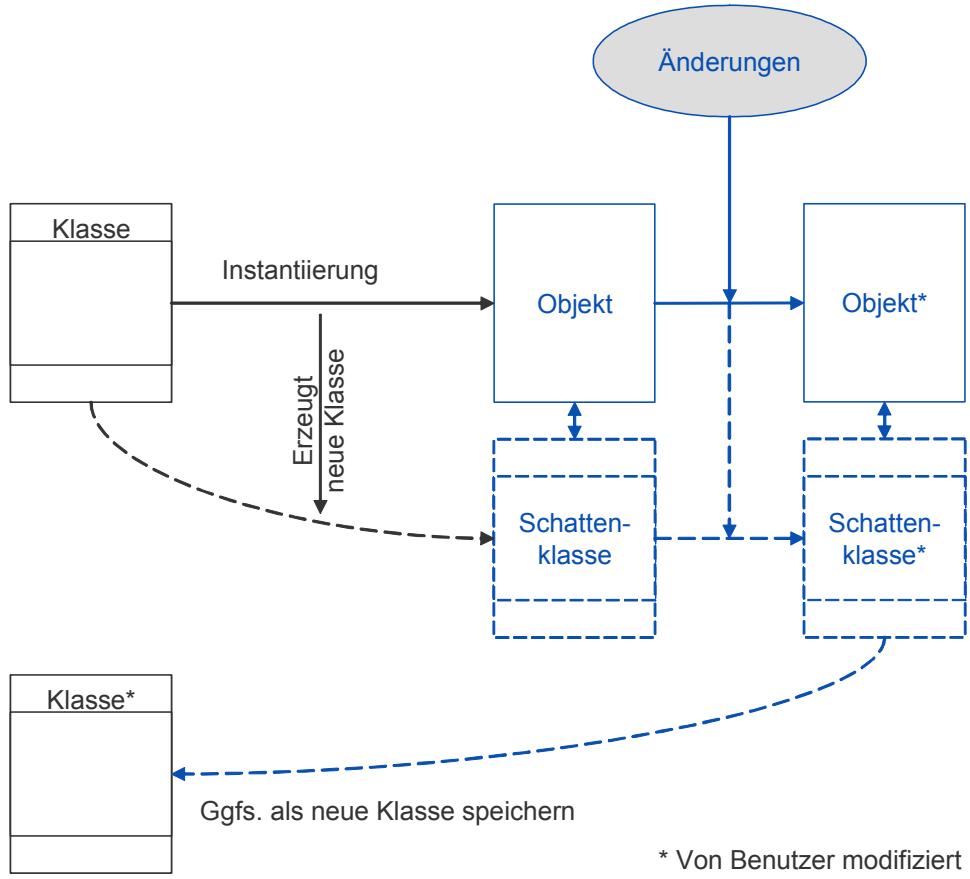


Abbildung 5-5 Zusammenhang zwischen Klasse, Objekt und Schattenklasse

Der Benutzer erzeugt ein neues Objekt in dem Modell als Instanz einer Klasse. Durch die Instantiierung wird in der Datenbank das Objekt erzeugt. Dabei werden alle Definitionen der ausgewählten Klasse und aller übergeordneter Klassen als Kopie in der Datenbank angelegt. Diese Kopie stellt in dem Modell das Objekt aber gleichzeitig auch die Klassendefinition des Objektes dar und wird daher als **Schattenklasse** bezeichnet. Der Benutzer kann das Objekt nun nach den Bedürfnissen ändern, so dass sich das Objekt (Objekt*) und seine Schattenklasse (Schattenklasse*) von der Originalklasse unterscheidet. Diese neue Definition kann bei Bedarf in den Klassenbaum eingebunden werden. Dazu wird die Definition der Schattenklasse abzüglich der Definition der neuen Vaterklassen in dem Klassenbaum abgelegt (Klasse*)

Da in dem Datenbankmodell die Zuordnung zu der ursprünglichen Klasse gespeichert ist, können Änderungen an der Klasse dem Objekt zugewiesen werden. Genauso ist es möglich veränderte Objekte wieder auf die ursprüngliche Klassendefinition zurückzuführen. Die Vorteile dieser Lösung liegen in:

- **Objektbasierte Modellierung**

  Nicht alle Benutzer sind mit der objektorientierten Modellierung vertraut und haben Schwierigkeiten, die objektorientierte Analyse durchzuführen. Diese Benutzer sind eher mit der objektbasierten Modellierung vertraut. Durch die Schattenklasse wird dem Benutzer eine objektbasierte Modellierung vorgetäuscht, die im weiteren Verlauf des Projektes durch Speicherung der Schattenklasse im Klassenbaum in eine objektorientierte Modellierung überführt werden kann.

- **Flexible Modellierung**

  Im Rahmen von Phase 0/A Studien müssen verschiedene Konzepte untersucht werden. Im Rahmen dieser Untersuchungen müssen teilweise vorhandene Modelle an neue Anforderungen angepasst werden. Diese Änderungen können unter Anderem neuartige Berechnungsmethoden oder andere technische Lösungen sein. Damit die Modelle schneller an diese neuen Bedürfnisse angepasst werden können, erlaubt ISM die neuen Berechnungen erst in dem Objekt zu implementieren und später im Klassenbaum allen Projekten zur Verfügung zustellen.

5.2 Datentypisierung

In ISM werden die verfügbaren Datentypen für Inputs, Parameter und Outputs von dem Benutzer definiert. Nur die in den Datentypkatalog vorhandenen Datentypen können vom Benutzer bei der Modellierung verwendet werden. Ein Datentyp wird in ISM durch folgende Kriterien definiert:

- **Name**
- **Formelzeichen**
- **Einheit**
- **Beschreibung, die die Bedeutung des Datentyps erklärt**
- **Typ des Wertes (Ganzzahl, Gleitkommazahl, Text, Aufzählung)**

- **im Falle einer Aufzählung die Werte, die der Datentyp annehmen kann und**
- **eine Operation**

Die Möglichkeit, eine Operation bereits bei der Datentypdefinition festzulegen, erleichtert dem Modellierer sogenannte Budgets zu erzeugen. Budgets fassen charakteristische Parameter eines Satelliten zusammen. Die häufigsten Budgets sind Masse, Energiebedarf, Risiko, Ausrichtungsgenauigkeit oder Kosten. Folgende Operationen können bei einem Datentypen definiert werden:

- **Keine Operation**
- **Benutzerdefinierte Operation**

  Dadurch können auch komplexere Funktionen in dem Modell vom Benutzer selbst definiert werden.

- **Summe**

$$w_v = \sum_{i=1}^{n} w_{k,i}$$

- **Maximal Wert**

$$w_v = \max(w_{k,i}; i = [1; n])$$

- **Minimum Wert**

$$w_v = \min(w_{k,i}; i = [1; n])$$

- **Arithmetischer Mittelwert**

$$w_v = \frac{\sum_{i=1}^{n} w_{k,i}}{n}$$

- **Quadratischer Mittelwert**

$$w_v = \sqrt{\frac{\sum_{i=1}^{n} \left(w_{k,i}\right)^2}{n}}$$

mit
w = Wert eines Attributes eines Objektes
v = hierarchisch übergeordnetes Objekt (Vater)
k = hierarchisch gleiche Objekte (Kinder)
n = Anzahl hierarchisch gleicher Objekte (Anzahl Kinder)

Diese Charakterisierung des Datentyps ist vergleichbar mit einem komplexen Datentyps der wie folgt als Klasse in C++ implementiert werden könnte:

```
class NAME {
   public:
         NAME ();   // Initialisierung des Datentyps
         ~NAME ();// Destructor des Datentyps

         // Operation für den Datentyp
         string operation(NAME werte[]) {…}

   protected:
         string Formelzeichen;
         string Einheit;
         string Beschreibung;
         enum Werttype {integer, real, text, document, enum};
         // gespeicherter Wert
         string Werte;
         // Liste möglicher Werte im Falle eines Enum Werttyps
         enum *enum_Werte[];
}
```

ISM definiert drei Standartdatentypen, die alle Klassen und Objekte besitzen müssen:

- Name eines Objektes

  Definition:

Name:	Name
Formelzeichen:	name
Einheit:	--
Beschreibung des Datentyps:	Definiert den Namen eines Objektes
Typ des Wertes:	Text
Operation:	keine

- Beschreibung eines Objektes

Definition:

Name:	Beschreibung
Formelzeichen:	desc
Einheit:	--
Beschreibung des Datentyps:	Eine kurze Beschreibung der Aufgabe/Sinn des Objektes
Typ des Wertes:	Text
Operation:	keine

- eineindeutiger Bezeichner

Definition:

Name:	Identifizierung
Formelzeichen:	id
Einheit:	--
Beschreibung des Datentyps:	Eineindeutige Identifizierung eines Objektes. Dadurch können mehrere Objekte den gleichen Namen besitzen
Typ des Wertes:	Text
Operation:	keine

Diese Datentypen dienen der Beschreibung und Identifizierung der Objekte im Modell und sind somit zwingend erforderlich.

5.3 Kollektion

Die im Rahmen von verschiedenen Projekten durchgeführten Modellierungen zeigen wie wichtig es ist, mehrere Datentypen zusammenfassen zu können. Daher bietet ISM die Möglichkeit eine Sammlung von Datentypen in sogenannte Kollektionen zusammen zu fassen. Kollektionen können zur Definition von Inputs, Parametern, Outputs und auch Flussrelationen verwendet werden. Bei einer Kombination von Outputs, Flussrelationen und Inputs, die auf einer Kollektion basieren hat der Benutzer die Möglichkeit durch Änderung der Kollektionsdefinition diese Änderungen automatisch auf die Schnittstellen und Relationen im Modell aufzuprägen.

Beispiel:

Komponenten werden alle durch ihre Masse, Größe, Trägheit und Energiebedarf charakterisiert. Diese Informationen können durch den objektorientierten Modellierungsansatz nun einfach in einer Klasse definiert werden.

Wenn nun aber die Informationen über Energiebedarf zwischen zwei Objekten (z. B. Verbraucher und Batterie) ausgetauscht werden müssen, kann man über eine Kollektion, in der alle Datentypen für die Beschreibung des Energiebedarfs (z. B. Spannung U sowie Stromstärke I) zusammengefasst sind, die Flussrelation zwischen beiden Objekten basierend auf einer Kollektion schnell erzeugen.

Kollektionen können wie Klassen vererbt und stereotypisiert werden. Die Stereotypisierung dient dazu, in der graphischen Darstellung die unterschiedlichen Flussrelationen auf Basis von Kollektionen farblich hervorzuheben (siehe Tabelle 5-3). Für die Sichtbarkeit und Vererbung der Datentypen innerhalb der Kollektionen sind `private` und `protected` als Sichtbarkeit für die Kollektionen definiert. Diese Sichtbarkeiten werden durch die jeweiligen Sichtbarkeiten der Attributdefinitin der Klasse überschrieben.

Bei ISM ist eine Standardkollektion („Basic Collection“) definiert. In dieser Kollektion sind die drei Standarddatentypen für Name, Beschreibung und Bezeichner zusammengefasst. Diese Kollektion muss auch der „Basic Class“ – Klasse im Klassen-Diagramm zugewiesen werden (siehe auch Abschnitt 5.6.1). Durch diese Regel wird sichergestellt, dass alle Objekte mindestens diese drei Parameter besitzen.

Tabelle 5-2 Sichtbarkeiten und Zugriffsrechte der Datentypen einer Kollektion

Attribute	Bezeichnung	Sichtbarkeit und Zugriffsrecht
Datentype	`private`	Der Datentyp **wird nicht** an eine abgeleitete Kollektion **vererbt**.
Datentype	`protected`	Der Datentyp **wird** an eine abgeleitete Kollektion **vererbt**.

5.4 Relationen

Relationen beschreiben den Zusammenhang zwischen zwei Elementen. Wie bereits von Walther definiert, gibt es Struktur- und Flussrelationen (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 47ff). Im Rahmen von ISM werden folgende Relationen definiert:

- **Strukturrelation**

  Die Strukturrelationen beschreiben den Aufbau des Systems bzw. Modells. Hierzu gibt es in UML prinzipiell zwei Typen von Strukturrelationen, die **Vererbung, Komposition** und **Aggregation**. Die Vererbung definiert eine Konkretisierung einer Klasse und ist somit eine Strukturrelation bei den Klassen und Kollektionen.
  Die Komposition kann als „besteht aus“ und die Aggregation als „enthält“ Beschreibung der Struktur interpretiert werden. Ein Objekt kann nur einem einzigen anderen Objekt durch Komposition zu gewiesen werden. Durch die Aggregation kann ein Objekt mehreren anderen Objekten zugewiesen werden. Somit besteht bei einer Aggregation das Objekt auch noch dann, wenn ein übergeordnetes Objekt entfernt wird. Bei der Komposition werden alle untergeordneten Objekte gelöscht. Aus rein technischer Sicht wäre die Aggregation die richtige Strukturrelation. Denn eine Komponente kann weiterhin existieren auch wenn das System aufgelöst wird (z.B. Demontage).

  Da der ISM Ansatz für die Modellierung von Systemen vorgesehen ist, wird im Rahmen von ISM nur die Komposition betrachtet. Die Komposition wurde ausgewählt, weil in dem Modell die untergeordneten Objekte gelöscht werden müssen sobald ein übergeordnetes Objekt aus dem Modell entfernt wird. Die Komposition kann als Zeiger auf das jeweilige andere Objekt betrachtet werden. Durch diesen Mechanismus können die übergeordneten Objekte auf die Parameter der untergeordneten Objekte lesend zugreifen.

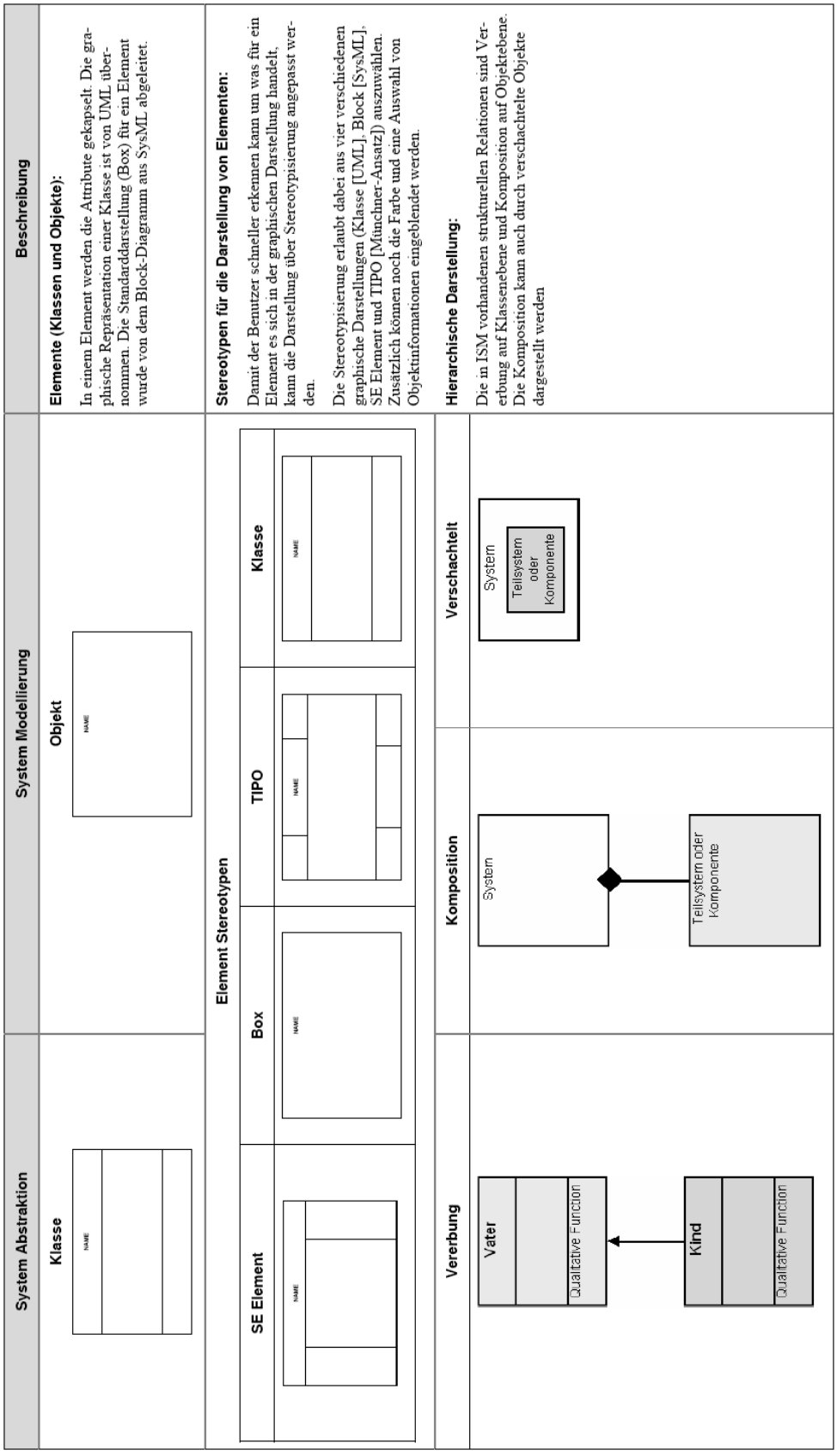
- **Flussrelation**

  Flussrelationen beschreiben die Verknüpfung von Outputs und Inputs von zwei verschiedenen Objekten. Dabei sind die Inputs bei der Datenübertragung aktiv, sie lesen die Outputwerte. Flussrelationen können von Kollektionen abgeleitet werden. In diesem Fall kann man die Kollektionen auch als Relationstypen bezeichnen. Durch diese Typisierung von Flussrelationen können mit ISM Funktionsmodelle nach Walther (Walther – Systemtechnisches Verfahren zur Bestimmung, S. 47ff) oder Otto (Otto, Wood 2001 – Product design, S. 1011ff) dargestellt werden.

5.5 Graphische Notation

Ein wichtiges Merkmal von ISM ist die Verwendung einer graphischen Darstellung des Modells. Dies ist ein wichtiger Unterschied zu den aktuellen Ansätzen wie NASA PAD, ESA-IDM oder ICEMaker. Durch die graphische Darstellung, die alle Objekte und Relationen darstellt, können die Benutzer das Gesamtmodell leichter verstehen und Zusammenhänge bzw. Abhängigkeiten erkennen. Dazu muss die graphische Darstellung eine Fachdisziplin - unabhängige Darstellung zulassen. Für die Abbildung eines Modells in ISM wird auf Objekt und Klassenebene eine Notation der graphischen Elemente angelehnt an UML und SysML definiert. Die verschiedenen Notationen und deren Bedeutung sind in Tabelle 5–3 aufgelistet. Daneben erlaubt ISM auf Klassenebene Regeln für den strukturellen Aufbau von Modellen zu definieren. Tabelle 5–4 listet die Graphische Notation für die beiden Regeln auf.

Tabelle 5-3 Übersicht über die Graphische Notation in ISM



System Abstraktion	System Modellierung		Beschreibung
	Komposition	**Verschachtelt**	**Flussrelation:**
			Flussrelationen definieren eine Verknüpfung zwischen Outputs und Inputs eines Elementes. Daher wird der Pfeil an der rechten Kante (Output) eines Elementes (Teilsystem A) zu der linken Kanten (Input) eines Elementes (Teilsystem B) gezeichnet. Flussrelationen sind vom Output zum Input gerichtet. Die Daten am Output werden von dem Input abgerufen. Ein Pfeil repräsentiert dabei **eine oder mehrere** Verknüpfungen zwischen einem Output Objekt und dem Input Objekt.
	Flussrelation zwischen über- und untergeordneten Objekten		**Flussrelation über Hierarchieebenen**
			In manchen Fällen müssen Informationen von einem übergeordneten Objekt an ein untergeordnetes Objekt (oder andersherum) übertragen werden. In diesem Fall beginnen und enden die Pfeile in der verschachtelten Ansicht, an der Innenseite des Objektes. Bei der Kompositionsansicht werden die Flussrelationen normal gezeichnet.
	Flussrelation zwischen verschiedenen Ästen des Modellbaums		**Flussrelation über Teilsystemgrenzen:**
			Sollen Flussrelationen über Teilsystemgrenzen hinaus dargestellt werden, so werden in der verschachtelten Ansicht die Pfeile über die übergeordneten Objekte geleitet. Somit wird verdeutlicht, wo Teilsystemgrenzen überschritten werden. Somit kann man eine SysML konforme Darstellung erzwingen. *Beispiel: Das Objekt **Teilsystem A** bekommt Daten von **Teilsystem B**. Beide Objekte sind in verschiedenen Ästen des Modellbaums angeordnet.*

Die neue Modellierungsmethode für die simultane Systementwicklung

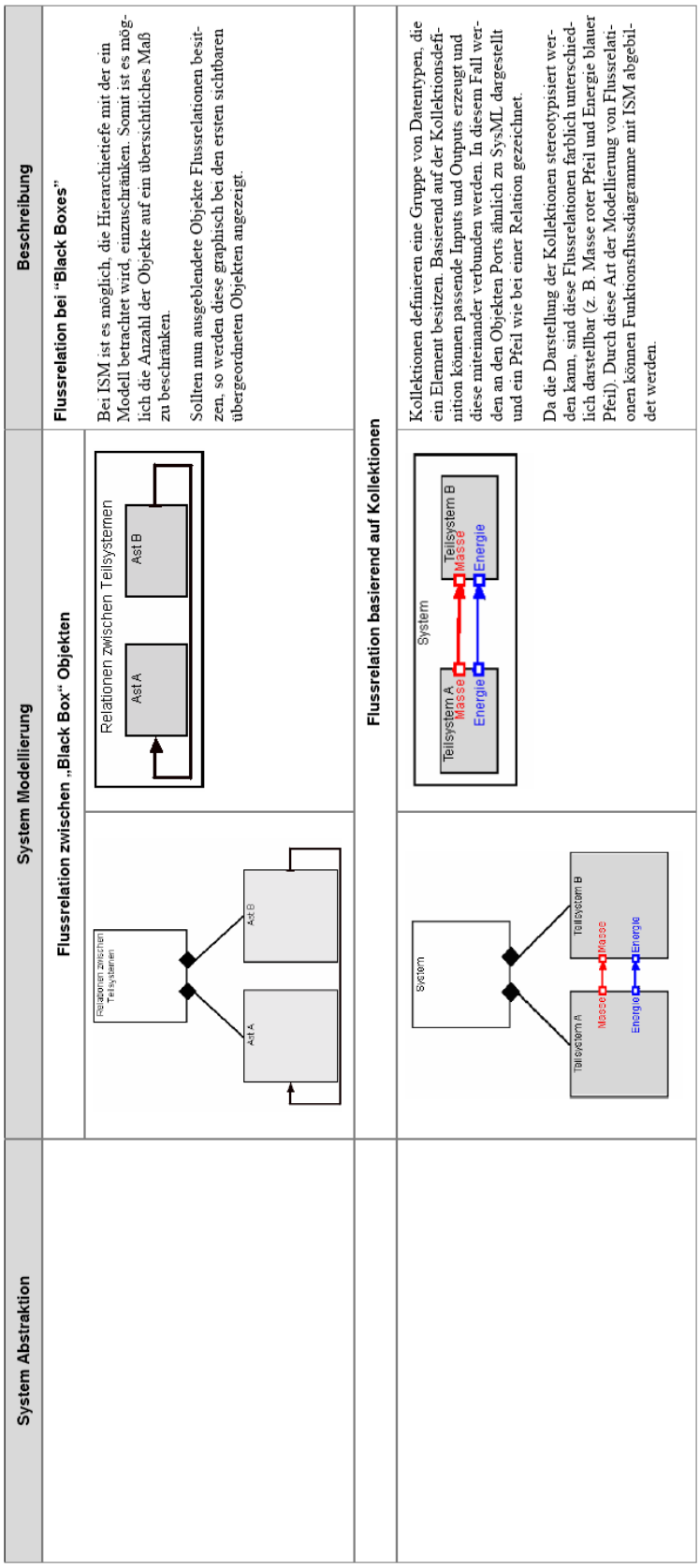


Tabelle 5-4 Graphische Notation für Regeln auf Klassenebene

System Abstraktion	System Modellierung	Beschreibung
Regeln für die Erzeugung von Modellen In ISM können Regeln sowohl für den strukturellen Aufbau (Strukturregeln) als auch für Flussrelationen bereits im Klassenmodell definiert werden. Die notwendigen graphischen Notationen für diese zwei Diagramme werden hier definiert.		
Strukturregeln		**Strukturregeln**
		Die Strukturregeln definieren bereits auf Klassenebene die Substrukturen für das System-Modell. Die Komposition wird durch einen ausgefüllten Diamanten und eine Bezugslinie dargestellt. Bei dem Diamanten befindet sich ein Multiplikator von 1, die Anzahl möglicher Subobjekte wird durch einen entsprechenden Multiplikator bei der zweiten Klasse definiert. Die Multiplikatoren sind nach UML Standard definiert. Es kann ein Bereich für die Anzahl definiert werden (n bis m) wobei ein „*" für m beliebig viele Objekte bedeutet.
Flussrelationsregeln		**Flussrelationsregeln**
		Die Flussrelationsregeln definieren im Klassendiagramm mögliche Flussrelationen im System-Modell. Durch diese Regeln können beim Erzeugen eines neuen Objekts die notwendigen Relationen automatisch mit erzeugt werden.

5.6 Diagramme

Diagramme bilden neben dem Datenmodell und der Sammlung von Funktionen den dritten Hauptkern von ISM. Dabei werden die in Tabelle 5-3 dargestellten Notationen zur Beschreibung der Modelle verwendet. Sie bieten den Entwicklern eine graphische Repräsentanz des verwendeten Modells. Dadurch kann der Entwickler schneller das Modell und die internen Abhängigkeiten erkennen. OMG definiert Diagramme wie folgt (Object Management Group (Hg.) Februar 2007 – Unified Modeling Language, S. 675):

> Jedes Diagramm hat einen *Inhaltsbereich*. Optional können ein *Rahmen* und eine *Überschrift* dargestellt werden.

Die folgenden Kapitel beschreiben den Inhaltsbereich der verschiedenen Struktur-Diagramme. Für ein besseres Verständnis werden bei Bedarf die Diagramme anhand eines Beispiels erklärt. Das Beispiel dient auch zur Verifikation von ISM und ist im Detail im nachfolgenden Kapitel beschrieben. Bei dem Beispiel handelt es sich um das Raumfahrzeug Philae, das von einem Satelliten abgekoppelt wird und auf einem Asteroiden landen soll.

Die Diagramme in ISM werden in Vererbungs-, Kompositionsregeln- und Flussregeln-Diagramm für die abstrakte Modellierung der Klassen und Kollektionen sowie System-Diagramm für die System Modellierung unterschieden. Das Kollektions-Diagramm wird hier nicht extra beschrieben, da es identisch mit dem Klassen-Vererbungs-Diagramm ist. Alle Klassen Diagramme und das Kollektions-Diagramm definieren die projektübergreifenden Daten in dem Datenmodell (vgl. Abbildung 5-2 auf Seite 76). Für die Definition der Datentypen gibt es keine Diagramme.

5.6.1 Klassen-Vererbungs-Diagramm

Das Klassen-Vererbungs-Diagramm dient zur Modellierung aller Klassen. Dieses Diagramm stellt die Klassenstruktur basierend auf der Vererbung dar. Es zeigt welche Attribute und Funktionsblöcke eine Klasse definiert. Die Wurzel der Baumstruktur, die durch die Klassenvererbung entsteht, ist die „Basic Class“ (siehe Abbildung 5-6). Diese Klasse ist der Ursprung aller Klassen und Objekte. In dieser Klasse werden die für alle Objekte notwendigen Grundinformationen anhand der „Basic Collection“ definiert. Unterhalb der „Basic Class“ werden die

Klassen für die verschiedenen Arten für die Systembeschreibung definiert. Diese Systemarten, auch Domänen genannt, können zum Beispiel die Anforderungen (Requirements), Kosten, Prozesse oder technischen Systembeschreibung (Product Tree) sein. ISM definiert keine dieser Arten, so dass der Entwickler frei in der Strukturierung seines Problemfeldes ist. Das hier zur Verifikation von ISM verwendete Beispielsystem wird durch die Anforderungen und die technische Systembeschreibung abgebildet.

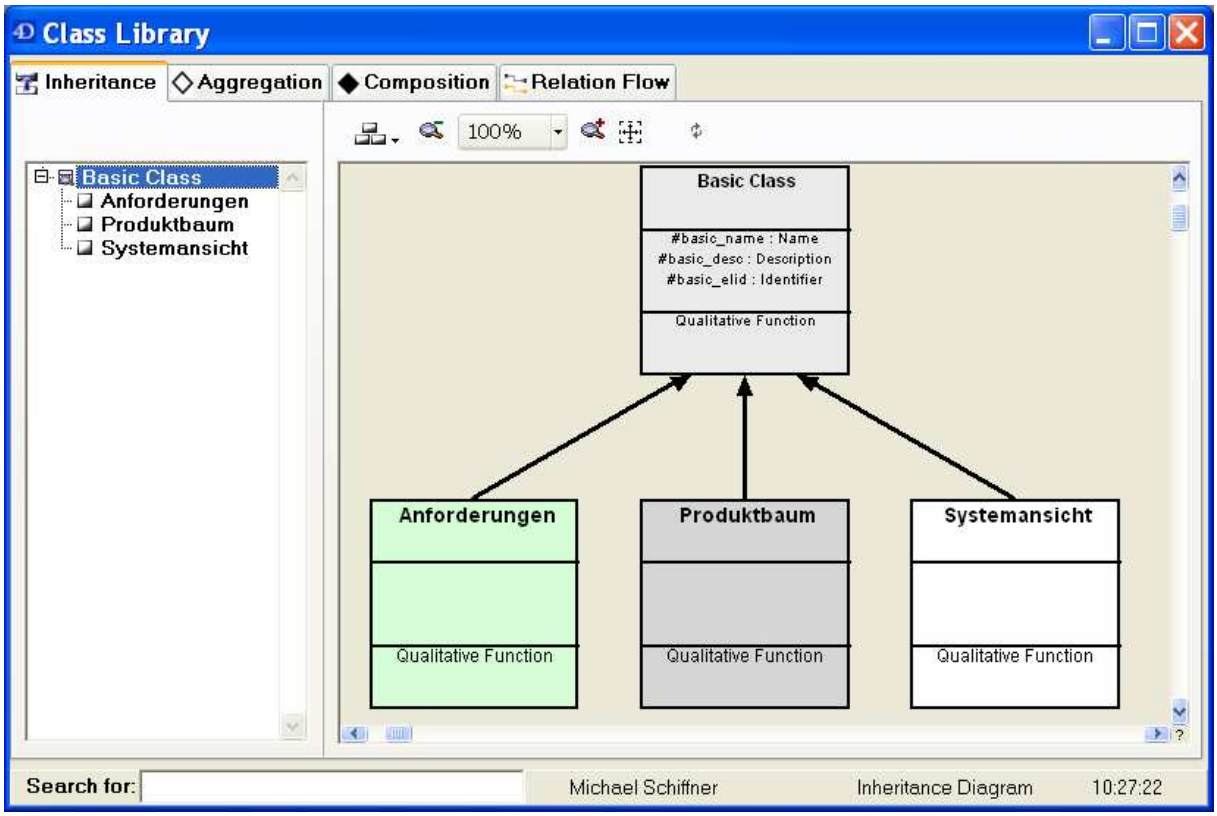


Abbildung 5-6 Klassen-Vererbungs-Diagramm umgesetzt in (v)Sys-ed

Die in Abbildung 5-6 dargestellten Klassen enthalten zudem noch eine Klasse „Systemansicht“. Diese Klasse definiert die Gesamtansicht auf das System und integriert alle Systembeschreibungen in einer Ansicht. In dem Systemmodell (Objektsystem) wird das Wurzelobjekt, welches gleichzeitig die Option repräsentiert, von dieser Klasse abgeleitet (siehe Abbildung 5-7).

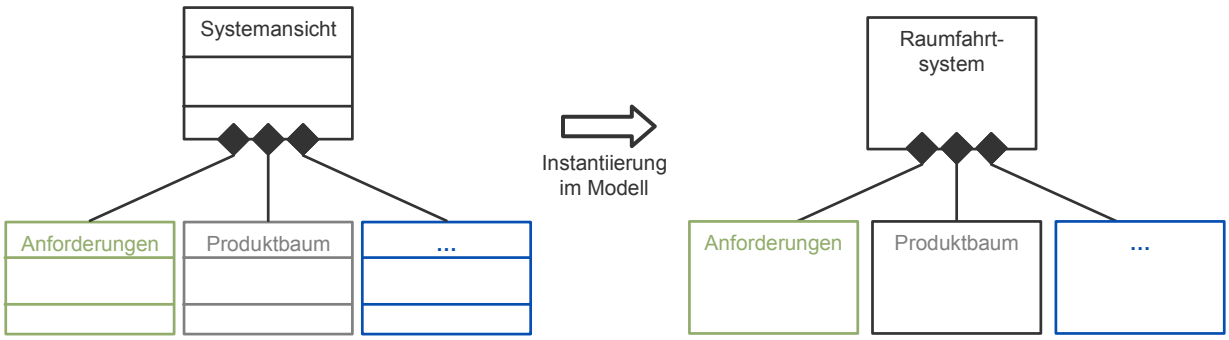


Abbildung 5-7 Zusammenhang zwischen Klassenbaum (links) und der Instantiierung des Systems (Philae) mit den verschiedenen Domänen (rechts)

Unterhalb der Klassen für die Systemdomänen können die domänen-spezifischen Klassen definiert werden. Zusammenfassend stellt das Klassen-Vererbungs-Diagramm den Klassen-Stammbaum in der Datenbank dar. Neben der strukturellen Repräsentation der Klassen werden auch die definierten Parameter und vorhanden Funktionen abgebildet. Durch Auswahl der Klassen in diesem oder in einem der anderen Klassen-Diagramme können die Klassendefinition im Detail angesehen und verändert werden.

5.6.2 Kompositionsregel-Diagramm

Das Kompositionsregel-Diagramm ist eines von zwei Diagrammen zur Abbildung von Regeln bei der Erzeugung von neuen Objekten. Eine Kompositionsregel bestehend aus zwei Klassen, dem Kompositionspfeil und dem Multiplikator. Der Kompositionspfeil ist durch einen ausgefüllten Diamanten und einer Verbindungslinie abgebildet. Der Diamant wird an der übergeordneten Klasse gezeichnet. ISM nutzt diesen Regelsatz bei der Erstellung eines neuen Systemmodells. Dazu wird zwischen der „System View“ Klasse für die Gesamtansicht und den von dem Benutzer definierten Domänen eine Kompositionsregel abgebildet. Abbildung 5-8 zeigt diesen Regelsatz. In dieser Abbildung sind die beiden Domänen Systembeschreibung (Product Tree) und Anforderungen (Requirements) Teile des Gesamtsystems (System View).

Gleichzeitig sind bei den Domänenklassen Multiplikatoren angezeigt. Die Multiplikatoren definieren wie viele Objekte von der jeweiligen Klasse erzeugt werden sollen. In diesem Beispiel ist es kein oder ein Objekt von der Klasse „Product Tree“ und genauso von der Klasse „Requirements“. Da nicht immer sichergestellt ist, dass das jeweilige Systemmodell auch die Anforderungen oder Systembeschreibung enthalten muss, ist hier der Multiplikator von Null

definiert. Mehr als eine Systembeschreibung „Product Tree“ ist für ein System im Normalfall auch nicht sinnvoll. Daher existiert maximal ein Unterobjekt im Systemmodell.

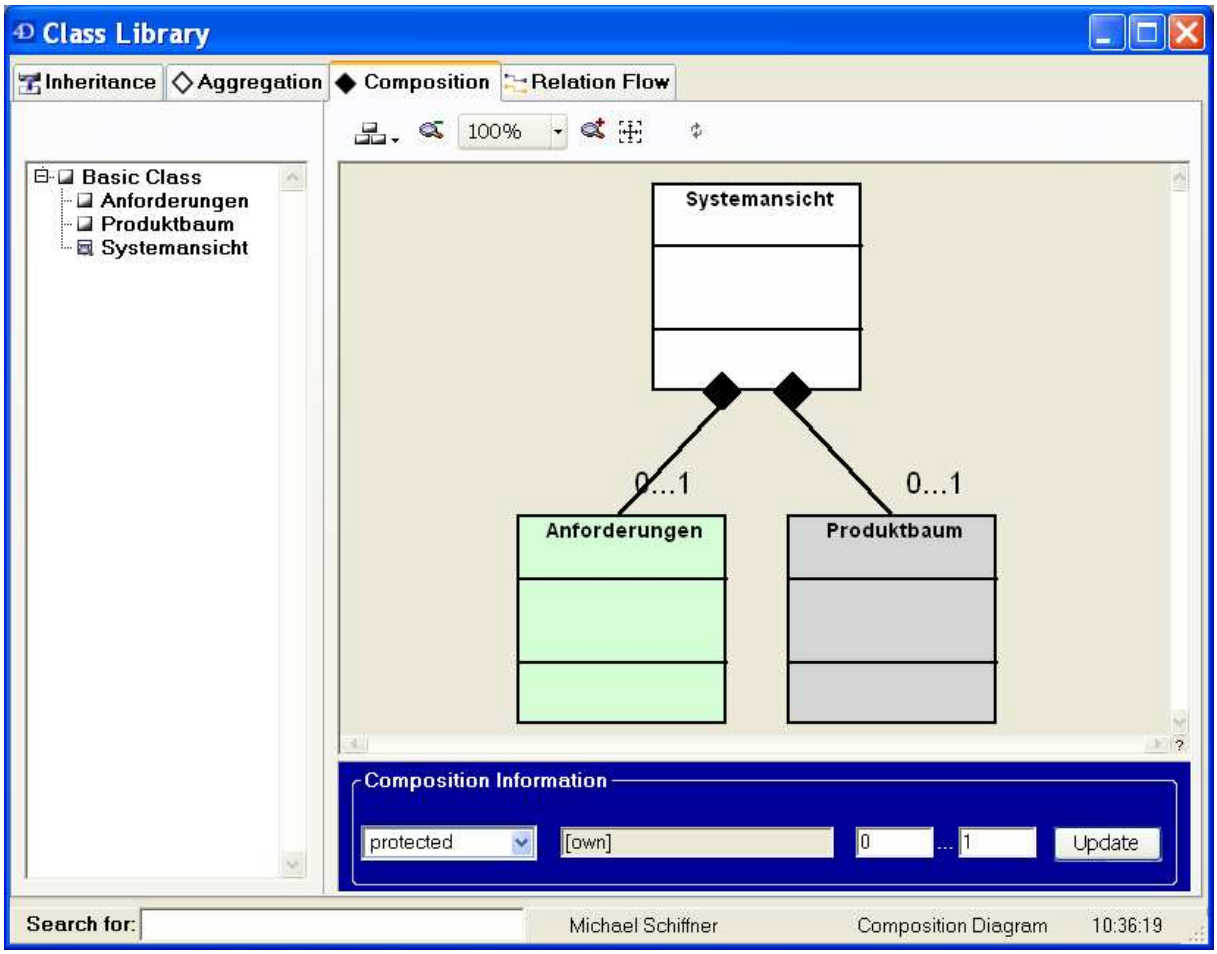


Abbildung 5-8 Kompositionsregel Diagramm in (v)Sys-ed

Die Definition des Multiplikators ist von UML übernommen und kann die folgenden Werte bereiche haben:

- **[0 … n]**

  Es sind zwischen keinem und n Objekte zu erzeugen.

- **[n … m]**

  Es sind zwischen m und n Objekte zu erzeugen.

- **[n]**

  Es sind genau n Objekte zu erzeugen.

- **[*]**

  Es sind beliebig viele aber mehr als null Objekte zu erzeugen.

- **[m … *]**

  Es sind zwischen m und beliebig vielen Objekte zu erzeugen.

Da nicht sichergestellt ist, dass eine eindeutige Anzahl von Unterobjekten erzeugt werden müssen (Fall [n]), muss der Benutzer bei der Erzeugung der übergeordneten Klasse angeben, wie viele Objekte erzeugt werden sollen. Die Objekte können hierbei basierend auf dieser Klasse oder einer der untergeordneten Klasse erzeugt werden.

Bei der Kompositionsregel können die beiden Sichtbarkeiten private und protected definiert werden.

Tabelle 5-5 Sichtbarkeiten und Zugriffsrechte bei einer Kompositionsregel

Attribute	Bezeichnung	Sichtbarkeit und Zugriffsrecht
Datentype	`private`	Die Regel **wird nicht** an eine abgeleitete Klasse **vererbt**.
Datentype	`protected`	Der Regel **wird** an eine abgeleitete Klasse **vererbt**.

5.6.3 Flussregel-Diagramm

Das Flussregel Diagramm ist das zweite Diagrammen zur Abbildung von Regeln bei der Erzeugung von neuen Objekten. Eine Flussregel besteht aus zwei Klassen, dem Relationsflusspfeil und ggf. dem Port. Der Port wird nur bei Flussrelationen, die auf einer Kollektion basieren, gezeichnet. Der Pfeilfuß wird an der Output-Klasse und die Pfeilspitze an der Input-Klasse gezeichnet (Abbildung 5-9).

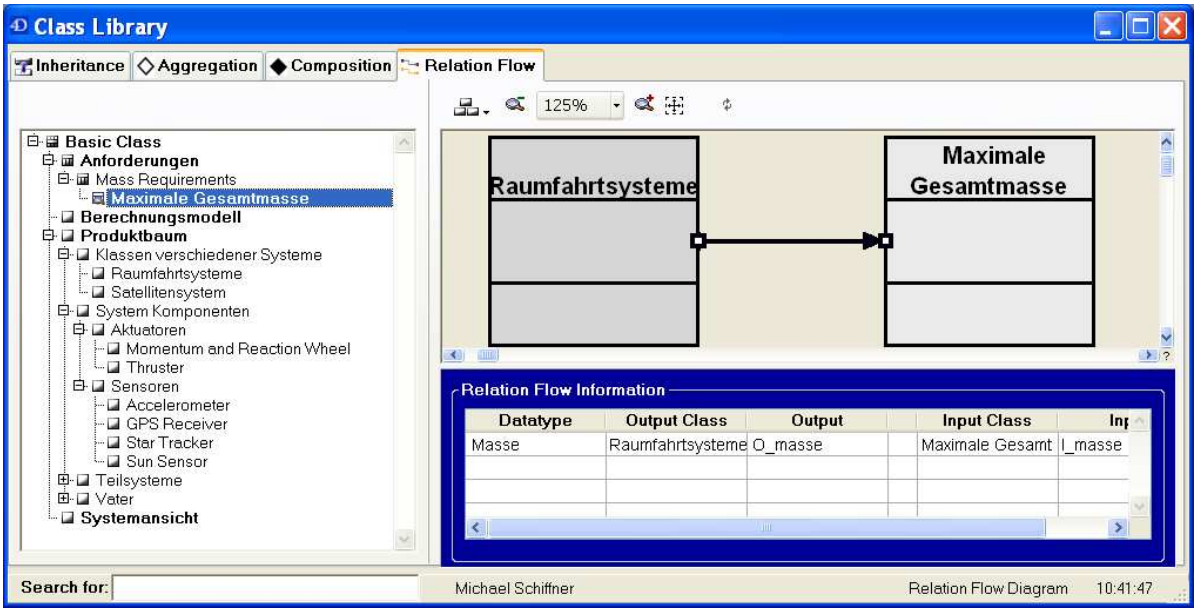


Abbildung 5-9 Flussregel Diagramm in (v)Sys-ed

Die Flussregeln werden bei der Erzeugung von einem Objekt angewendet, welches von einer der beiden betroffenen Klassen oder auch abgeleiteten Klassen erzeugt wird.

5.6.4 System Diagramm

Das System Diagramm beschreibt das Modell bestehend aus der Objektstruktur und den Flussrelationen. Dieses Diagramm erlaubt Modelle sowohl in der Kompositions-Darstellung (Abbildung 5-10) als auch in der verschachtelten Darstellung nach Igenbergs und Ehrlenspiel (Abbildung 5-11) darzustellen. Flussrelationen werden entsprechend der in Tabelle 5-3 definierten Notation in dem Diagramm gezeichnet.

Da die Gesamtmodelle eines Systems sehr umfangreich werden können, kann der Benutzer auch Ausschnitte aus dem Modell betrachten. Dazu kann das Wurzelobjekt (oberstes Objekt in der graphischen Darstellung) und die Anzahl der Hierarchieebenen entlang der Systemstruktur nach unten ausgewählt werden. Diese Art der Ausschnittbetrachtung ist auch bei den vorher beschriebenen Vererbungsdiagrammen möglich. Eine spezielle Ausschnittsbetrachtung ist die Darstellung von allen Blätterobjekten. Blätterobjekte sind Objekte, die selbst keine untergeordneten Objekte besitzen. Durch diese Darstellung ist die Abbildung von Funktionsflussdiagrammen möglich (Otto, Wood 2001 – Product design, S. 1011ff).

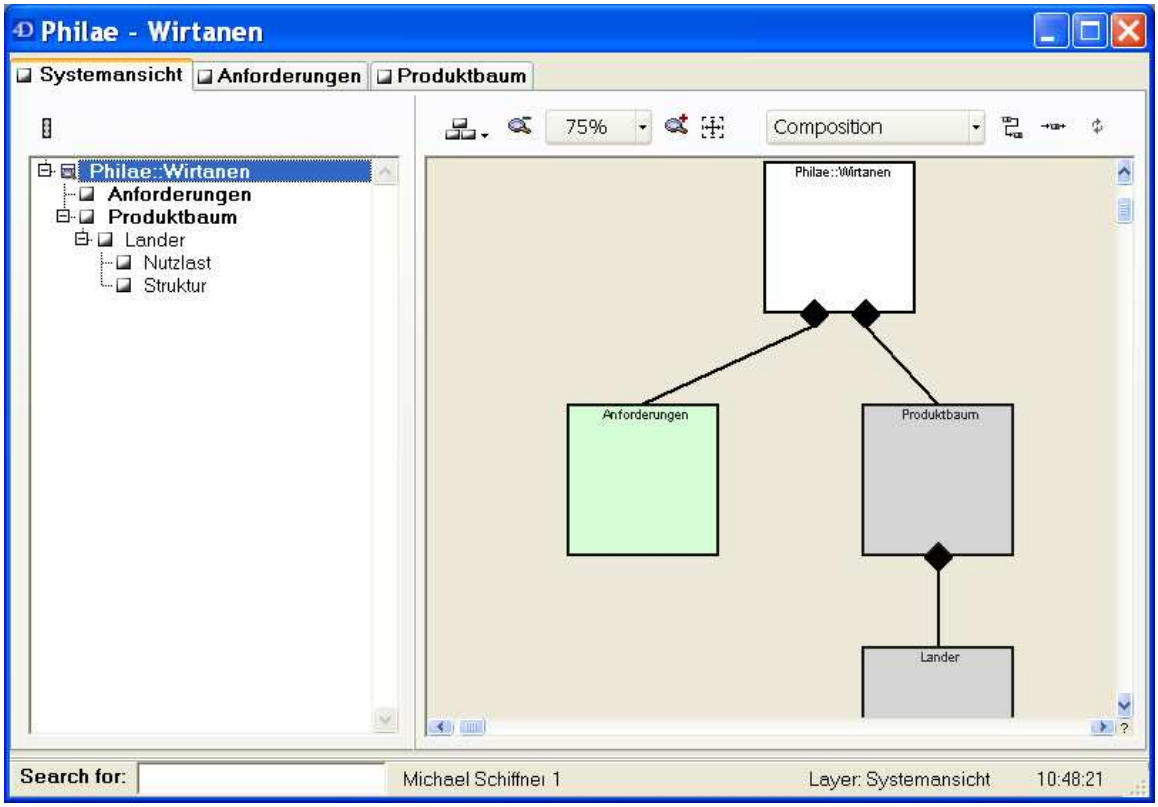


Abbildung 5-10 System Diagramm – Kompositionsdarstellung in (v)Sys-ed

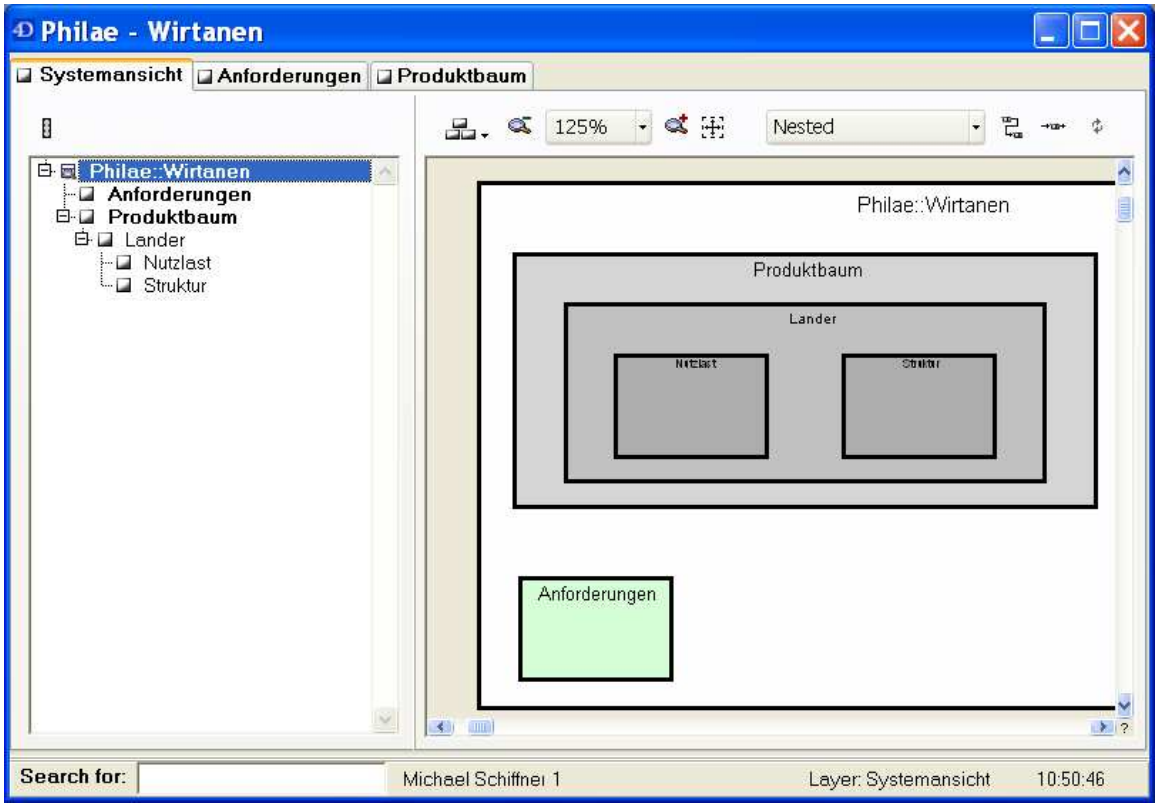


Abbildung 5-11 System Diagramm – verschachtelte Darstellung in (v)Sys-ed

5.7 Sammlung von Funktionalitäten

In dem ISM Ansatz wird nicht nur ein Modell für die Modellierung von Systemen und das notwendige Datenmodell definiert, sondern auch Funktionalitäten, die mit dem Systemmodell möglich sind. Diese Funktionalitäten werden durch die Sammlung von Funktionalitäten definiert.

Die drei Ebenen der Anwendung basieren auf dem Entwicklungsprozess für Satellitensysteme. Da dieser Prozess bei dem hier vorgestellten CE modellbasiert abgearbeitet werden soll, gibt es drei Hauptarbeitsschritte.

1. **Modellbildung und Auslegung**

   In diesem ersten Arbeitschritt wird das Systemmodell erzeugt und mit diesem Modell eine erste Auslegung erzeugt.

2. **Analyse und Simulation**

   Nachdem das Modell erzeugt wurde kann mit diesem gearbeitet werden. Dazu zählen die Analyse des Designverhaltens und die Simulation von Designänderungen.

3. **Evaluierung von Designs**

   Nachdem verschiedene mögliche Designs untersucht wurden, müssen diese Anhand von projektabhängigen Kriterien verglichen und bewertet werden.

5.7.1 Modellbildung und Auslegung

Bei der Modellierung von Systemen sind folgende Funktionalitäten mit zu berücksichtigen:

- **Multi-User Zugriff**

  Die Software, die den ISM Ansatzes umsetzt, muss das gleichzeitige Modellieren von mehreren Benutzern an der gleichen Option erlauben. Dazu gehört sicherzustellen, dass Änderungen an dem Modell unmittelbar an alle Betroffenen kommuniziert werden und immer nur eine Person Änderungen an einem Objekt, einer Klasse, Kollektion oder einem Datentyp durchführt.

Nur wenn die Software diese Anforderung erfüllt, ist ein integrierter und modellbasierter Satellitenentwurf möglich, an dem alle betroffenen Fachdisziplinen mitarbeiten.

- **Filter für Ansichten**

  Es ist wichtig dem Benutzer die Möglichkeit anzubieten, Informationen auszublenden. Das Ausblenden kann in den Diagrammen oder Editoren für die verschiedenen Elemente stattfinden. Unter Ausblenden kann hierbei ein ausgrauen der Informationen oder gar weglassen der jeweiligen Objekte oder Attribute verstanden werden.
  Die Filter können die Benutzer selber auf Basis von Datentypen und Kollektionen definieren. Durch diese flexible Anpassung der Ansichten ist eine disziplinabhängige Darstellung der Modelle möglich. Hierdurch kann ein Ingenieur, der sich nur für die elektrischen Zusammenhänge interessiert, alle für ihn unwichtigen Informationen ausblenden.

- **Automatische Archivierung der Werte**

  Während der Modellierung des Satelliten und der anschließenden Auslegung des Systems verändern sich die Werte der Attribute. Damit das Designteam die Veränderungen über die Projektlaufzeit nachvollziehen kann, müssen die Werte archiviert werden.

- **Schnittstellen**

  Um die Konsistenz des gesamten Modells sicherzustellen, sollte die Modellierung der Datentypen, Kollektionen, Klassen und des Systemmodells in einer zentralen Software implementiert werden. Die verschiedenen anderer Softwarepakete für die Realisierung des Funktionsblocks (vgl. Abbildung 5-3 auf Seite 77) sollten über eine geeignete Schnittstelle (z.B. Werkzeugabhängige Schnittstelle, Webservice oder XML) angebunden werden, so dass diese nur die Werte und nicht aber das Modell direkt verändern können.

- **Zugriffsverwaltung**

  Da in der Datenbank mehrere Projekte gleichzeitig abgespeichert werden können und die Datentypen, Kollektionen und Klassen projektübergreifend zur Verfügung stehen, ist eine Zugriffsverwaltung auf Daten-Typen, Kollektionen, Klassen und Objektebene zu empfehlen. Folgende Zugriffsregeln und Rollen werden daher vorgeschlagen:

  - Lese-/Schreib-Zugriff auf jeweils Datentyp-, Klassen- und Kollektion - Ebenen
  - Lese-/Schreib-/Modellierungs-Zugriff auf Projektebene
  - Rechte für die Verwaltung der Benutzer und Datenbankeinstellungen

Neben diesen speziellen Anforderungen muss die Software natürlich die durch die objektorientierte Modellierung bedingten Funktionalitäten wie zum Beispiel Vererbung und Erzeugen von Instanzen umsetzen.

5.7.2 Analyse und Simulation

Nachdem das Modell erzeugt und ein erstes Design für den Satelliten ausgelegt ist, kann dieses weiter untersucht werden. Zu diesen Tätigkeiten gehören die Analyse des Modells und die Simulation von Designänderungen. Eine zeitlich dynamische Simulation ist nicht Ziel der hier vorgestellten Modellierung. Zu der Analyse des Modells gehört unter anderem:

- **Analyse nach unabhängigen Attributen**

  Während des Designs können sich einzelne Attribute des Systems als Design-Treiber heraus kristallisieren. Design-Treiber sind Eigenschaften eines Systems, die einen bedeutenden, meist negativen, Einfluss auf die Gesamtlösung haben. Diese Parameter selbst werden teilweise durch andere Attribute beeinflusst. Um nun diese Design-Treiber beeinflussen zu können, müssen jene Attribute identifiziert werden, die diesen Design-Treiber beeinflussen und selbst unabhängig sind ($f(x) = a; a = const.$). Für diese Art der Analyse ist die Matrixbeschreibung für die Abhängigkeiten zwischen Inputs, Parameter und Outputs in den Klassen und Objekten am Besten geeignet.

- **Änderungsanalyse**

  Bei der Änderungsanalyse wird das Antwortverhalten des Designs auf Grund einer Änderung untersucht. Da für diese Analyse das Gesamtmodell neu berechnet werden muss, ist diese Art der Analyse durch eine Simulation des Modells realisierbar.

Bei der Simulation sind folgenden Anforderungen zu berücksichtigen:

- **Multi-User Simulation**

  Wie schon bei der Modellierung sollen mehrere Benutzer mit dem Modell arbeiten können. Dies ist bei der Simulation insofern von Bedeutung, als die Änderung von Werten, die Berechnung und Ergebnisdarstellung im Falle einer Multi-User Simulation auf mehrere Rechner verteilt werden kann und muss.

- **Simulation über Wertebereich**

  Wird bei einem Attribut der Wert nicht nur durch einen anderen Wert ausgetauscht sondern ein Wertebereich definiert, kann man eine Analyse der Empfindlichkeit, mit der das Systemdesign auf Änderungen reagiert, untersuchen. Werden dazu noch mehrere Attribute in dem Modell gleichzeitig variiert, kann die Designantwort über ein breiteres Spektrum der Anregung untersucht werden. Bei dieser Art der Simulation über mehrere Parameter verbunden mit einem Wertebereich können sowohl ein erheblicher Rechenaufwand als auch große Datenmengen entstehen. Dies sollte bei einer derartigen Simulation unbedingt berücksichtigt werden.

5.7.3 Evaluierung

Der letzte wichtige Bereich bei der Systemauslegung ist der Vergleich und die Bewertung von verschiedenen Designlösungen. Hier bietet der ISM-Ansatz durch seine strikte formale Modellierung einen einfachen Zugriff auf die notwendigen Daten in den Modellen.

Bei der Bewertung von verschiedenen Lösungen werden charakteristische Eigenschaften, wie zum Beispiel Kosten, Risiko oder Dauer des Projektes mit einander verglichen. Da diese Eigenschaften auf Datentypen aufbauen, sind die notwendigen Informationen anhand des aus-

gewählten Datentyps schnell und einfach im Modell identifizierbar. Zusätzlich sind bei dem ISM Ansatz alle Projekte und Optionen in einer einzigen Datenbank abgespeichert, so dass schnell auf die jeweiligen Daten zugegriffen werden kann.

5.8 Datenmodell

Das Datenmodell von ISM dient zum Speichern aller Daten in der Datenbank und kann auch als Grundlage für die Datenstruktur der Software dienen.

Da es derzeit noch üblich ist, Daten in einer relationalen Datenstruktur (XML oder Datenbank) abzuspeichern, wird das Datenmodell in Abbildung 5-12 in einem Entity-Relationship-Diagramm abgebildet, obwohl eine Darstellung in UML möglich ist. Bei diesem Diagramm handelt es sich um eine graphische Darstellung zwischen den Datenobjekten (Entities) und den Zusammenhängen (Relationship) zwischen den Objekten.

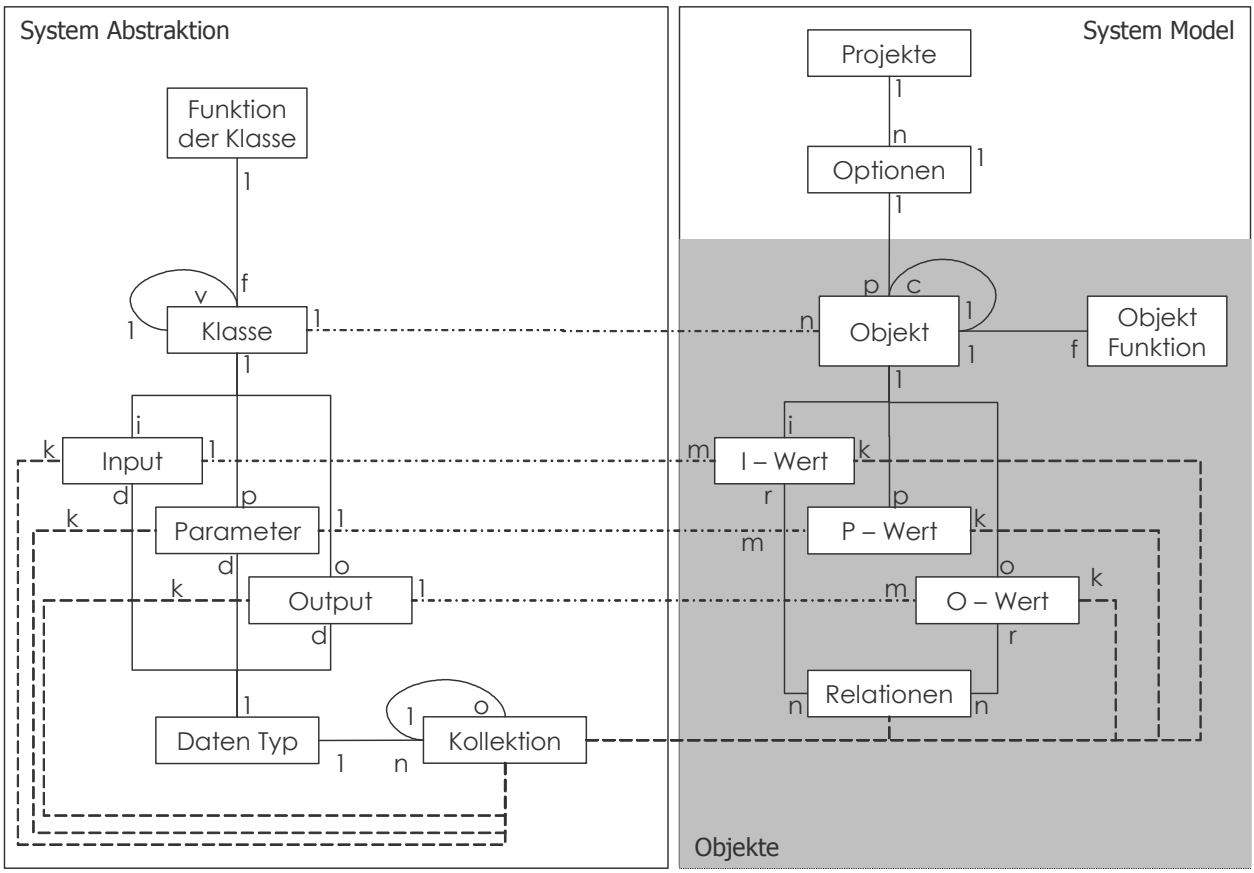


Abbildung 5-12 Definition des Daten Modells in Entity-Relationship Modell

Es wird derzeit im Rahmen eines studentischen Projektes an einem UML Modell und XML Datenformat gearbeitet, so dass die Daten mit anderen Werkzeugen ausgetauscht werden können. Dazu zählt insbesondere das Werkzeug (U)CML-ed für die Überprüfung von Kompatibilitäten (Referenz auf Brandstätter), das derzeit am Lehrstuhl für Raumfahrttechnik entwickelte wird. Dadurch können die in ISM gewonnen Daten in die spätere Phase der Satellitenentwicklung überführt werden.

5.9 Implementierung in (v)Sys-ed

Um die Anwendbarkeit und den Nutzen den ISM Ansatz zu verifizieren, wurde die Modellierungsmethode in der Software (v)Sys-ed umgesetzt. (v)Sys-ed steht hierbei für „virtuelles System – Editor“. Die Architektur der Software besteht aus zwei Hauptelementen (siehe Abbildung 5-13). Dem Server und einem Client. Der Server enthält die Datenbank und die Kommunikationsschnittstelle zu den Clients. In dem Clients sind die graphische Benutzerschnittstelle für die Modellierung der Systeme und der Funktionsblock implementiert. Zu dem Funktionsblock gehört neben den oben aufgelisteten Anforderungen auch die Kommunikation zwischen den Clients für die Aktualisierung bei Änderungen im Modell. Die Schnittstelle für die Modellierung des Funktionsblocks in Microsoft Excel basiert auf dem Dynamic Data Exchange (DDE). Mit Hilfe von Dynamic Data Exchange (DDE) werden Kommunikationskanäle zwischen zwei Anwendungen eingerichtet, über welche die Anwendungen Daten und Metadaten austauschen (Schneider, Werner et al. 2001 – Taschenbuch der Informatik, S. 681). Neben der DDE Schnittstelle wurde auch eine XML Schnittstelle für die Integration mit anderen Software Werkzeugen implementiert.

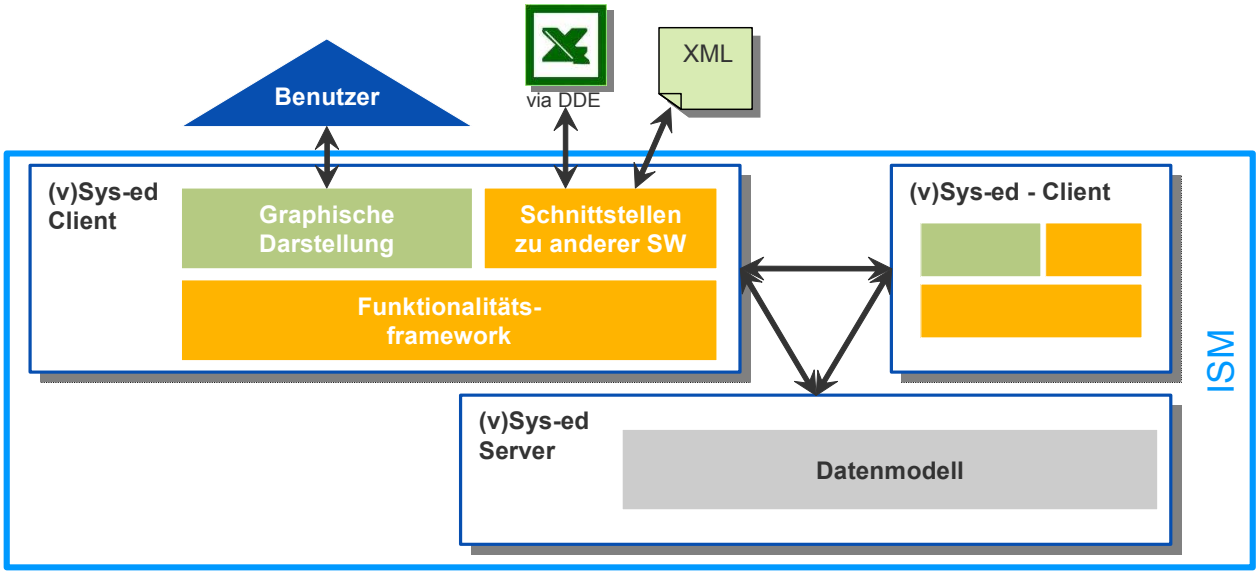


Abbildung 5-13 Architektur der Software (v)Sys-ed mit Darstellung der Benutzerschnittstelle und Schnittstellen zu anderen Softwarepaketen (SW)

Vergleicht man die Softwarearchitektur mit dem Ansatz der NASA so sieht man, dass alle Benutzer die gleiche Schnittstelle angeboten bekommen. Die Einbindung der benutzerspezifischen Anwendungen erfolgt indirekt über den Client. Abbildung 5-14 zeigt die allgemeine Softwarearchitektur.

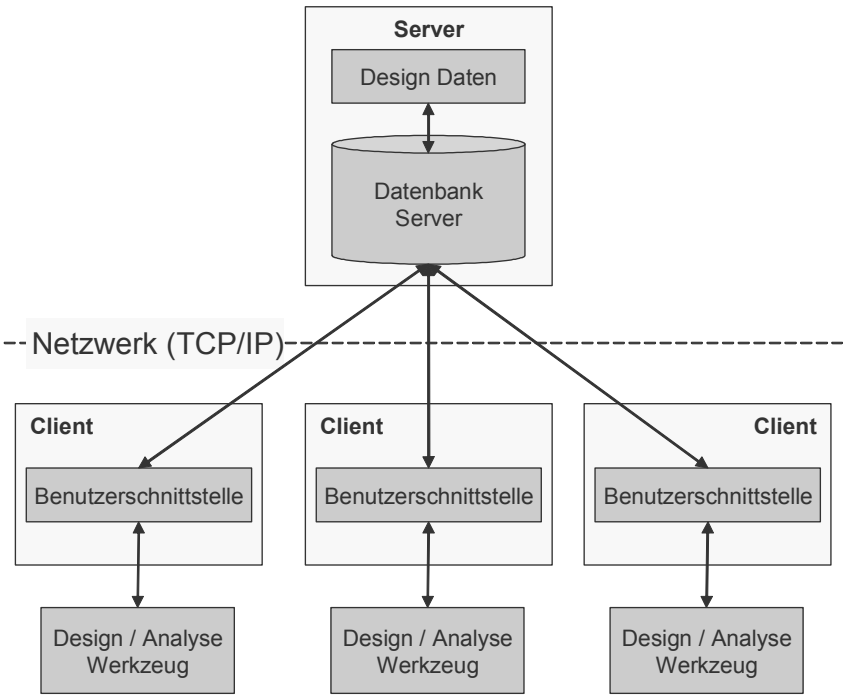


Abbildung 5-14 Allgemeine Architektur der Modellierungssoftware

(v)Sys-ed basiert auf dem Rapid Application Development Werkzeug 4th Dimension der Firma 4D. 4th Dimension integriert Relationale Datenbank mit einer Mehrplatz- und Einzelplatz-Umgebung, bietet eine Programmierumgebung und ermöglicht das Erstellen einer eigenen graphische Bedienoberfläche. Durch diese Fähigkeiten war es relativ einfach, der ISM Ansatz in einem Werkzeug zu realisieren und verschiedene Ideen und Lösungen zu implementieren und zu testen. Da 4th Dimension selber bereits eine Client-Server Infrastruktur anbietet, muss keine eigene Programmierung dieser komplexen Abläufe durchgeführt werden. Des Weiteren kann eine Einzelplatzversion ohne Änderungen sofort in eine Server-Client Version überführt werden. Darüber hinaus wird 4th Dimension seit mehreren Jahren für die Entwicklung der Modellierungswerkzeuge am LRT verwendet.

Die Software (v)Sys-ed wird seit dem Wintersemester 2006/2007 für den Lehrbetrieb im Rahmen von Praktika und Semester-/Diplomarbeiten eingesetzt. Sie dient zudem als Grundlage für die Verifikation der Modellierungsmethode, die nun im folgenden Kapitel beschrieben wird.

6 Verifizierung der neuen Modellierungsmethode

Der Autor war im Rahmen seiner Tätigkeiten am Lehrstuhl für Raumfahrttechnik am Rosetta Knowledge Management Video Approach (RKMVA) Projekt maßgeblich beteiligt. Das Projekt hatte die Aufgabenstellung, bei allen wissenschaftlichen Teams in Europa, die Experimente auf dem Rosetta Satelliten haben, Videointerviews durchzuführen. Das Ziel der Interviews war es, das vorhandene Wissen über die Experimente für die nächsten zehn Jahre auf Video zu konservieren. Dazu wurde ein entsprechendes Skript für die Interviews definiert, die Videos wurden in Mpeg2 komprimiert und ein Index über den Inhalt auf den Videos erzeugt. Das generelle Vorgehen und der Aufbau des Skriptes wurde von den beteiligten Kollegen in (Wilke, Finkel et al. 2003 – A video approach to knowledge) näher beschrieben.

Nach Fertigstellung des Rosetta-Satelliten musste wegen fehlender Zuverlässigkeit der Ariane-5 Trägerraketen der Start um ein Jahr verschoben werden und es musste ein anderer Zielkomet ausgewählt werden. Deshalb eignet sich das Rosetta-Projekt besonders gut für die Verifikation der vorgestellten Methodik. Die Rosetta-Mission ist die erste Mission, bei der nicht nur ein Satellit in einen Orbit um einen Kometen gebracht werden soll, sondern auch Experimente mit einem Lander auf dem Kometenkern landen sollen. Daher musste untersucht werden, ob die bestehende Konfiguration des Landers Philae eine sichere Landung auf dem neuen Kometen erlaubt.

In den folgenden Abschnitten wird die Mission selbst beschrieben, auf die Konfiguration eingegangen und die Modellierung von Philae und auf die Ergebnisse eingegangen. Dazu werden die Ergebnisse mit Simulationsergebnissen von Hilchenbach (Hilchenbach; Impact on a comet; Hilchenbach, Rosenbauer et al. 2004 – First Contact with a comet) verglichen.

6.1 Die Rosetta-Mission

Die folgende Beschreibung der Rosetta Mission wurde von der offiziellen ESA Rosetta Webseite übernommen (European Space Agency – ROSETTA at a Glance).

Der Name von Rosetta kommt von dem weltbekannten Rosetta Stein, der vor fast 200 Jahren das Entschlüsseln der Ägyptischen Hieroglyphen erlaubte. Auf ähnliche Weise hoffen die Wissenschaftler, durch die Rosetta Mission Aufschluss über die Entstehung unseres Sonnensystems zu bekommen.

Ursprünglich sollte Rosetta im Januar 2003 mit einer Ariane-5 Trägerrakete starten. Damals war als Zielkomet Wirtanen vorgesehen und der Satellit sollte diesen im Jahre 2011 erreichen. Aufgrund des Fehlstarts des Ariane 157 Fluges im Dezember 2002, bei dem zwei Satelliten verloren gegangen sind, haben ESA und Arianespace gemeinsam entschieden, den Start von Rosetta auf das Startfenster im Februar bis März 2004 zu verschieben. Somit konnte der ursprünglich geplante Komet Wirtanen nicht mehr angeflogen werden.

Im Mai 2003 wurden der neue Starttermin für Rosetta sowie der neue Zielkomet ausgewählt. Der Start erfolgte im 2. März 2004 auf einer Ariane-5 von Kourou in Französisch-Guayana. Rosetta soll den Kometen 67P/Churyumov-Gerasimenko im Jahr 2014 erreichen.

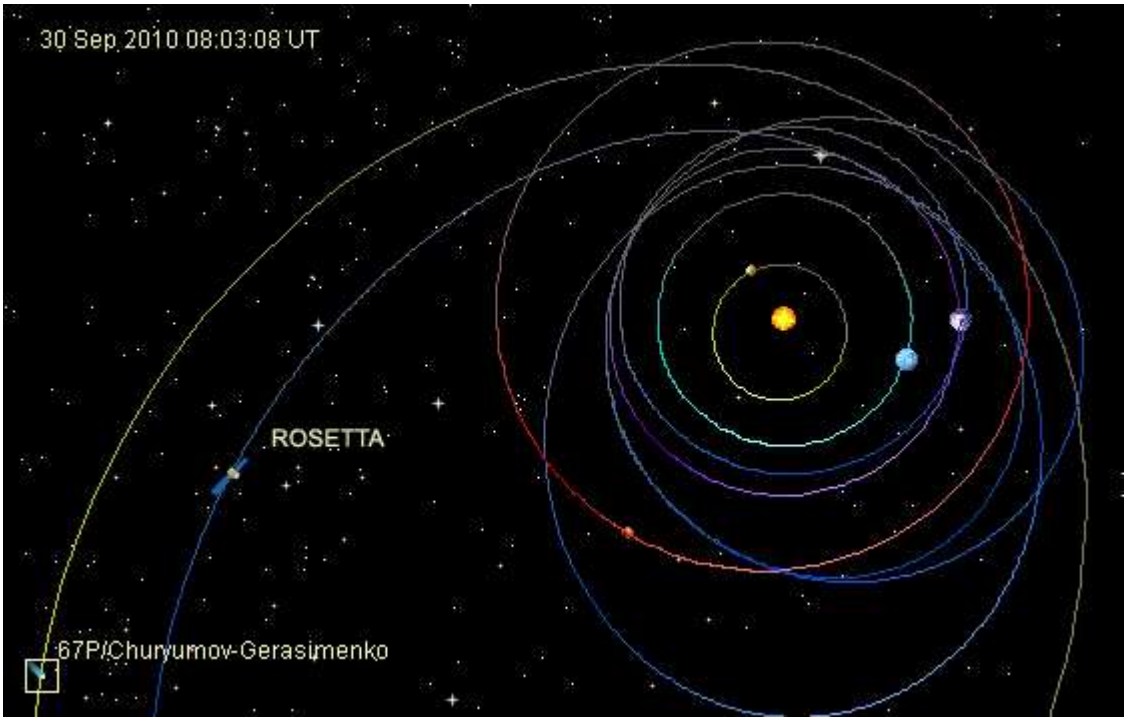


Abbildung 6-1 Rosetta Flugbahn zu Churyumov-Gerasimenko (European Space Agency – ESA Science & Technology)

Die Gesamtkosten des Projektes belaufen sich auf 1 Milliarde Euro und beinhalten den Start, den Satelliten mit der wissenschaftlichen Nutzlast (Instrumente und Lander) sowie die Betriebskosten über die gesamte Missionsdauer von ca. zehn Jahren.

Das Rosetta Raumfahrzeug der ESA wird als erstes Raumfahrzeug über einen längeren Zeitraum einen Kometen aus nächster Nähe untersuchen. Nachdem Rosetta in einen Orbit um 67P/Churyumov-Gerasimenko im Jahre 2014 eingeschwenkt sein wird, werden Aufnahmen des Kometen einen möglichen Landepunkt zeigen. Danach wird Philae auf dem eisigen Kern des Kometen landen. Rosetta wird den Kometen insgesamt für ca. ein Jahr umkreisen. Während dieser Phase nähert sich der Komet der Sonne. Nachdem der Komet und somit auch Rosetta das Perihel (kürzeste Distanz zur Sonne) durchschritten haben, wird Rosetta noch für ein weiteres halbes Jahr den Kometen umkreisen.

Kometen sind die Objekte aus der Urzeit des Sonnensystems und haben ihre chemische Zusammensetzung seit ihrer Entstehung nicht verändert. Daher liefern sie wichtige Informationen über die Entstehung unseres Sonnensystems. Ihre Zusammensetzung der Kerne spiegelt die Zusammensetzung des Sonnensystems vor mehr als 4.600 Millionen Jahren wider. Rosettas Instrumente werden, während sie den Kometen 67P/Churyumov-Gerasimenko umkreisen

bzw. auf seinem Kern landen, die notwendigen Informationen liefern, um die Geschichte unserer Umgebung im Weltall zu rekonstruieren.

Rosetta wird zudem helfen die Frage zu beantworten, ob möglicherweise Kometen, die auf der Erde eingeschlagen sind, komplexe organische Moleküle enthielten und damit zur Entstehung von Leben auf der Erde beitrugen. Außerdem vermutet man, dass leicht flüchtige Elemente in den Kometenkernen eine wichtige Rolle bei der Entstehung der Meere und Atmosphäre auf der Erde spielten.

Auf seiner Reise zum Kometen 67P/Churyumov-Gerasimenko wird Rosetta zwei Mal den Asteroidengürtel zwischen Jupiter und Mars durchqueren. Wissenschaftler haben bereits einige interessante Asteroiden entlang der Flugbahn von Rosetta identifiziert. Ein oder mehrere Asteroiden werden während der Reise dorthin für einen näheren Vorbeiflug ausgewählt. Während des Vorbeifluges sollen die Instrumente von Rosetta wissenschaftliche Ergebnisse über die Astroiden liefern.

Die folgende Aufstellung zeigt die Ereignisse auf der Reise von Rosetta in chronologischer Reihenfolge mit den erfolgten bzw. geplanten Zeitpunkten.

Ereignis	Zeitpunkt
Start	2. März 2004
Erstes Gravity Assist – Manöver an der Erde	4. März 2005
Mars Gravity Assist	25. Februar 2007
Zweites Gravity Assist Manöver an der Erde	13. November 2007
Vorbeiflug an Asteroid Steins	5. September 2008
Drittes Gravity Assist Manöver an der Erde	13. November 2009
Vorbeiflug an Asteroid Lutetia	10. Juli 2010
Beginn der Hibernation[13] Phase	Juli 2011
Ende der Hibernation Phase	Januar 2014

[13] Der Begriff Hibernation (dt. Winterschlaf) bezeichnet in der Raumfahrt einen Zustand eines Raumfahrtsystems in dem alle nicht existentiell notwendigen Systeme ausgeschaltet werden, ähnlich dem Winterschlaf der Tiere.

Rendezvous Manöver	Mai 2014
Beginn der globalen Bildaufnahmen	August 2014
Absetzten des Landers Philae	November 2014
Perihel Passage	August 2015
Ende der Mission	Dezember 2015

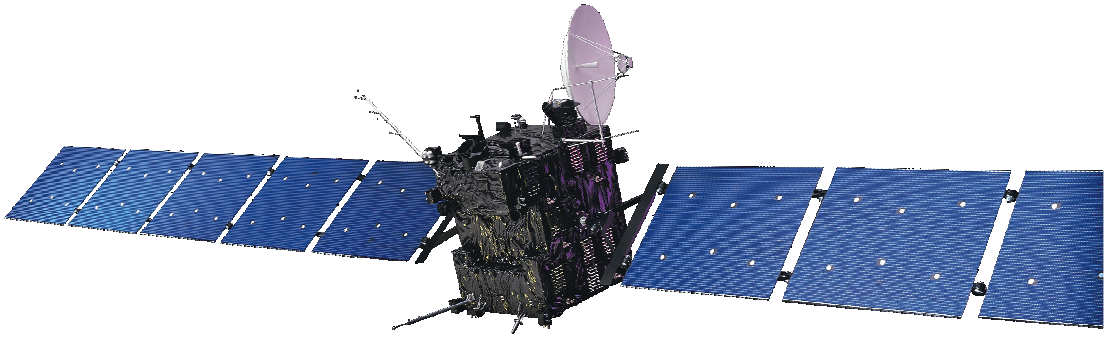


Abbildung 6-2 Rosetta Satellit (European Space Agency – ESA Science & Technology)

Rosetta selbst kann als ein großer Aluminium-Würfel bestehend aus zwei Hälften betrachtet werden. Die obere Hälfte – Payload Support Module – enthält alle wissenschaftlichen Experimente. Die untere Hälfte – Bus Support Module – enthält alle anderen Teilsysteme des Satelliten. Auf einer Seite des Satelliten ist eine 2,2 Meter große bewegliche Parabolantenne für die Kommunikation mit der Erde angebracht. Auf der gegenüberliegenden Seite befindet sich der Lander. Zwei riesige Solarkollektoren befinden sich auf zwei anderen Seiten des Satelliten (Abbildung 6-2). Um die maximale Sonneneinstrahlung ausnutzen zu können, sind die ausklappbaren Solarkollektoren um +/- 180 Grad entlang der Längsachse drehbar.

Rosetta hat ein Startgewicht von ca. drei Tonnen. Dabei sind die Tanks maximal mit 1,67 Tonnen Treibstoff aufgefüllt. Die wissenschaftliche Nutzlast auf dem Satelliten wiegt 165kg und der Lander selber wiegt noch mal 100 kg.

Alle Teilsysteme und wissenschaftlichen Nutzlasten des Satelliten befinden sich innerhalb eines 2,8 x 2,1 x 2,0 Metern Volumenwürfels. Die beiden Solarkollektoren lassen sich auf eine Länge von 14 Metern und einer Fläche von 64m^2 entfalten.

6.2 Rosetta Lander – Philae

Der ISM Ansatz wurde durch die Modellierung und Analyse von Philae verifiziert. Bei der Auslegung des Landers war ein wichtiger Aspekt das Aufsetzten auf dem Kometen. Abbildung 6-3 zeigt die Landekonfiguration von Philae mit den drei ausgeklappten Landebeinen.

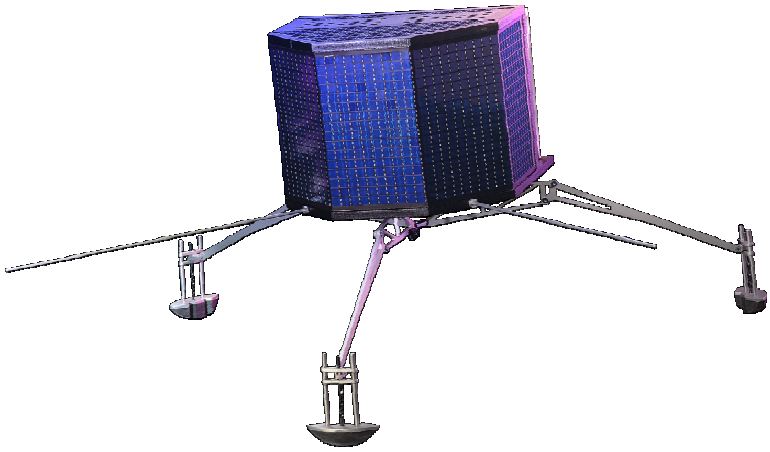


Abbildung 6-3 Rosetta Lander in Landekonfiguration [ESA 2005-10-05]

Bei der Landung auf dem Kometen kommt neben den Landebeinen, die eine Federcharakteristik aufweisen, auch einen Linearmotor als Dämpfungsglied zur Geltung. Ein Linearmotor dient nach der Landung als Höhenverstellung der Plattform mit den Experimenten (Kasten mit den Solarzellen in Abbildung 6-3). Neben diesem Feder-Dämpfer System besitzt der Lander Schrauben in den Füssen, eine Harpune an seiner Unterseite und Triebwerk auf seiner Oberseite, die zusammen dazu beitragen sollen, dass Philae nicht von der Oberfläche abprallt und sich wieder vom Kometen entfernt. Die Schrauben werden kurz nach dem Aufsetzten des jeweiligen Fußes in die Oberfläche des Kometen getrieben. Die Schrauben sollen sowohl das Rutschen über den Boden verhindern und bei einem Zurückfedern des Landers die Füße am Boden halten. Nachdem alle drei Füße Kontakt mit der Kometenoberfläche haben werden, wird die Harpune abgefeuert. Das an der Harpune befestigte Seil wird danach wieder eingeholt, so dass sich Philae damit auf dem Kometen festhalten kann. Zusätzlich werden die Triebwerke auf der Oberseite des Landers gezündet. Dies erzeugt eine weitere Kraft, mit der Philae auf dem Kometen gehalten werden soll.

Im Rahmen der Verifikation wird zu erst gezeigt, wie ein Modell mit ISM erzeugt wird. Anschließend wird gezeigt, wie man die Auswirkungen auf das System durch den neuen Kometen mit Hilfe dieses Modells untersuchen kann.

Mechanische Problemstellung

Vor der Modellierung des Systems werden zuerst die mechanischen Abhängigkeiten definiert und erklärt. Für die Berechnung des Aufsetzverhaltens wird in einem ersten Schritt die Geschwindigkeit im Bezugsystem des Kometen aus der Vektoraddition zwischen der Anfluggeschwindigkeit und der Kometenoberflächengeschwindigkeit berechnet. Die geometrischen Zusammenhänge sind in Abbildung 6-4 dargestellt.

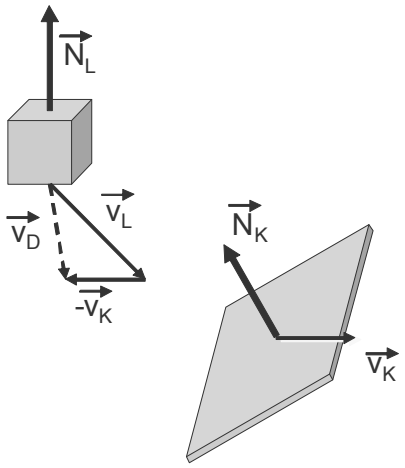


Abbildung 6-4 Abbildung der geometrischen Zusammenhänge zwischen Lander, Anfluggeschwindigkeit, Kometen und Geschwindigkeit der Kometenoberfläche

$\vec{N}_K$: Normalenvektor Kometenoberfläche
$\vec{N}_L$: Normalenvektor Lander
$\vec{v}_D$: Abstiegsgeschwindigkeit im Bezugsystem der Kometenoberfläche
$\vec{v}_K$: Geschwindigkeitsvektor Kometenoberfläche
$\vec{v}_L$: Geschwindigkeitsvektor Lander

Somit sei der Winkel α zwischen dem Normalenvektor des Landers und dem Normalenvektor der Kometenoberfläche der Angriffswinkel.

$$\cos(\alpha) = \frac{\vec{N}_L \bullet \vec{N}_K}{\left|\vec{N}_L\right|\left|\vec{N}_K\right|} \qquad 6.1$$

Des Weiteren seien der Winkel -β zwischen dem negativen Normalenvektor des Landers und $\vec{v}_D$ sowie der Winkel γ zwischen dem negativen Normalenvektor der Kometenoberfläche und $\vec{v}_D$ die beiden Aufsetzwinkel (siehe auch Abbildung 6-5). Somit gilt:

$$\alpha = \beta + \gamma \qquad 6.2$$

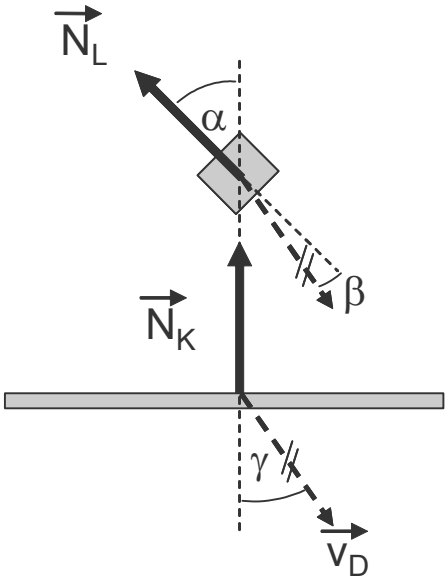


Abbildung 6-5 Definition der Winkel im 2-Dimensionalen Raum

Die nominalen Werte für die Simulation bei Hilchenbach sind (Hilchenbach; Impact on a comet, S. 364):

$$\alpha = 17°,\ \beta = 9°,\ \gamma = 8°$$

Die für das Aufsetzten relevante Geschwindigkeit (v_I) errechnet man durch:

$$\vec{v}_D = (\vec{v}_L - \vec{v}_K) \qquad 6.3$$

$$v_I = |v_D| \cos(\beta) \qquad 6.4$$

Die Geschwindigkeit des Landers lässt sich berechnen durch:

$$\vec{v}_L(t) = \vec{v}_0 + \int_0^{t_I} \vec{a}(\vec{r})\, dt$$

$$\vec{r} = \vec{r}_0 - \int_0^{t} \vec{v}_L(t) dt$$

$\vec{a}(\vec{r})$: Beschleunigung durch den Kometen
$\vec{r}$ Abstandsvektor zum Kometenmittelpunkt
$\vec{r}_o$ Abstandsvektor zum Kometenmittelpunkt zum Zeitpunkt der Abtrennung

von Rosetta

t_I: Zeit zwischen Abtrennen und dem Zeitpunkt an dem der Lander den Kometen berührt.

$\vec{v}_L$: Geschwindigkeitsvektor des Landers zum Zeitpunkt t

$\vec{v}_0$: Geschwindigkeit des Landers nach dem Abtrennen von Rosetta [kann bis zu 0,5*m/s* betragen (Hilchenbach; Impact on a comet, S. 362)].

Bei der Berechnung muss noch das Potentialfeld der Gravitation berücksichtigt werden. Da aber nur die Oberflächenbeschleunigung des Kometen bekannt ist, wird die folgende Formel verwendet:

$$F = ma = -\frac{\mu m}{r^2} \text{ mit } \mu = R_K^2 g_K$$

$$a(r) = -\frac{{R_K}^2 g_K}{r^2} \qquad 6.5$$

$a(r)$: Beschleunigung durch den Kometen

g_K: Oberflächenbeschleunigung des Kometen [für Wirtanen mit $7 \cdot 10^{-5} \frac{m}{s^2}$ und für 67P/Churyumov-Gerasimenko mit $10^{-3} \frac{m}{s^2}$ geschätzt (Hilchenbach; Impact on a comet, S. 364; Hilchenbach, Rosenbauer et al. 2004 – First Contact with a comet, S. 292)]

R_K: Radius des Kometen

Eine Abschätzung der Aufsetzgeschwindigkeit bei verschiedenen Startgeschwindigkeiten für beide Kometen zeigt Abbildung 6-6. Dabei wurde ein senkrechter Abstieg zur Kometenoberfläche angenommen. In der Abbildung wird zudem der durch die Gravitation bedingte Anteil bei der Aufsetzgeschwindigkeit angezeigt.

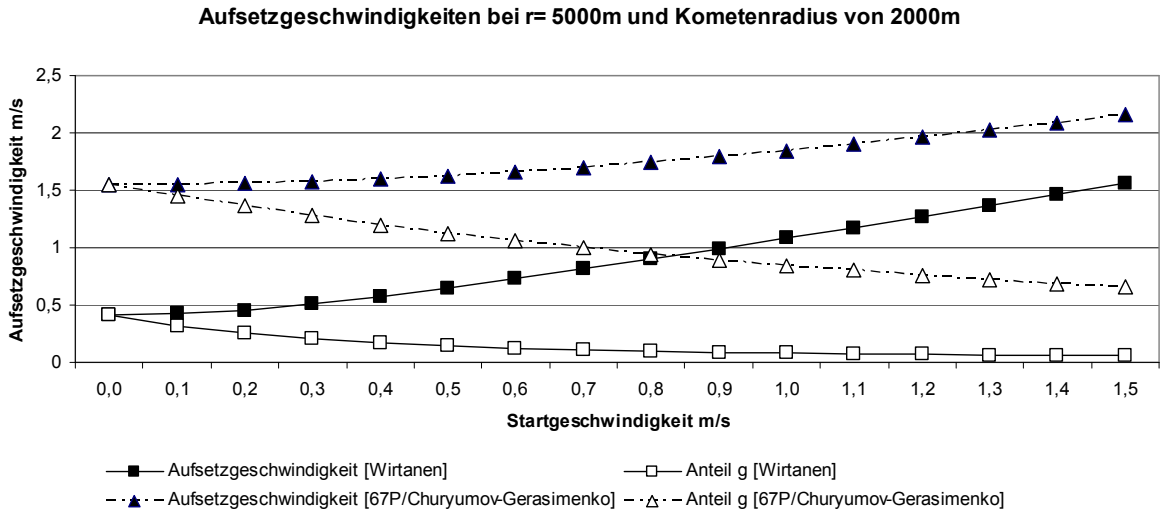


Abbildung 6-6: Aufsetzgeschwindigkeit für beide Kometen bei verschiedenen Startgeschwindigkeiten

Die dabei resultierende Aufsetzgeschwindigkeit wird bei Hilchenbach zwischen 0,1*m/s* und 1,8*m/s* für den Kometen Wirtanen angegeben (Hilchenbach; Impact on a comet, S. 364).

Nachdem die Aufsetzgeschwindigkeit bestimmt wurde, kann im nächsten Schritt die Bewegungsgleichung für das Aufsetzten dargestellt werden. Dabei wird das Verhalten für den Dämpfer genauer untersucht:

$$\sum_{i=1}^{n} F_i = 0 = F_G + F_D - F_T \qquad 6.6$$

F_D: Dämpfungskraft
F_G: Gravitationskraft
F_T: Trägheitskraft

mit:

$$F_G = m_L g_K \qquad 6.7$$

$$F_D = Dv \qquad 6.8$$

$$F_T = m_L \dot{v} \qquad 6.9$$

D: Dämpfungskoeffizient
m_L: Masse des Landers
v: Geschwindigkeit des Landers
$\dot{v}$: Geschwindigkeitsänderung des Landers über die Zeit

Setzt man die Formeln 6.8 bis 6.10 in Gleichung 6.7 ein, so bekommt man das Gleichungssystem:

$$m_L \dot{v} = -m_L g_K - Dv \tag{6.10}$$

Löst man die Gleichung nach der Beschleunigung auf, so bekommt man:

$$\dot{v} = -\frac{D}{m_L}\left(v + \frac{m_L}{D} g_K\right) \tag{6.11}$$

Definiert man:

$$\begin{aligned} x &= v + \frac{m_L}{D} g_K \\ \dot{x} &= \dot{v} \end{aligned} \tag{6.12}$$

und substituiert dies in Gleichung 6.11 so bekommt man:

$$\dot{x} = -\frac{D}{m_L} x \tag{6.13}$$

Diese Gleichung lässt sich lösen zu:

$$x(t) = x_0 e^{-\frac{D}{m_L} t} \tag{6.14}$$

Durch Resubstitution von Gleichung 6.13 in 6.15 bekommt man unter den Randbedingung, dass die Geschwindigkeit zum Zeitpunkt 0 gleich v_I ist:

$$v(t) = -\frac{m_L}{D} g_K + \left(v_I + \frac{m_L}{D} g_K\right) e^{-\frac{D}{m_L} t} \tag{6.15}$$

Für die Berechnung der Dauer für das Aufsetzten t_I auf der Kometenoberfläche muss in Gleichung 6.15 die Geschwindigkeit zu Null gesetzten werden. Durch Auflösen der Gleichung nach der Zeit t_I erhält man:

$$\begin{aligned} \frac{m_L}{D} g_K &= \left(v_I + \frac{m_L}{D} g_K\right) e^{-\frac{D}{m_L} t_I} \\ 1 + \frac{D}{m_L g_K} v_I &= e^{\frac{D}{m_L} t_I} \\ t_I &= \frac{m_L}{D} \ln\left(1 + \frac{D}{m_L g_K} v_I\right) \end{aligned} \tag{6.16}$$

Mit diesem Ergebnis kann man nun die Länge des Dämpfungsweges (l_I) durch Integration von Gleichung 6.15 berechnen.

$$
\begin{aligned}
l_I &= \int_0^{t_I} v(t)\,dt = \int_0^{t_I} \left[\left(v_I + \frac{m_L}{D} g_K \right) e^{-\frac{D}{m_L} t} - \frac{m_L}{D} g_K \right] dt \\
l_I &= \left(v_I + \frac{m_L}{D} g_K \right) \int_0^{t_I} e^{-\frac{D}{m_L} t}\,dt - \frac{m_L}{D} g_K (t_I - 0) \\
l_I &= \left(v_I + \frac{m_L}{D} g_K \right) \frac{-m_L}{D} e^{-\frac{D}{m_L} t} - \frac{m_L}{D} g_K t_I \\
l_I &= -\frac{m_L}{D} \left[\left(v_I + \frac{m_L}{D} g_K \right) e^{-\frac{D}{m_L} t} - g_K t_I \right]
\end{aligned}
\qquad 6.17
$$

Nachdem die für die Auslegung des Dämpfers notwendigen Gleichungen hergeleitet wurden, kann mit der Modellierung des Systems begonnen werden.

Die Modellierung

Für die Modellierung von Philae standen die Informationen aus den Publikationen von Hilchenbach (Hilchenbach; Impact on a comet, S. 361–369; Hilchenbach, Rosenbauer et al. 2004 – First Contact with a comet, S. 289–296) zur Verfügung. In diesen Veröffentlichungen wurden Ergebnisse von dynamischen Simulationen der Landung vorgestellt. Da ISM nicht für die dynamische Simulation sondern für die Auslegung von Systemen vorgesehen ist, wird für die Verifikation von ISM nicht der gesamte Landevorgang beschrieben, sondern nur die Auslegung der Verstellstrecke des Linearmotors bei der Modellierung betrachtet.

Bei der Modellierung müssen in einem ersten Schritt die verschiedenen Domänen, mit denen man das System beschreiben will, definiert werden. Dazu werden für dieses Beispiel die drei Bereiche Anforderungen, Produktbaum und ein Bereich für die Auslegungsrechnung (Berechnungen) definiert (siehe Abbildung 6-7). Die Anforderungen erlauben die Definition der Kometenspezifikationen (z.B. angenommener Radius und angenommene Gravitation) oder Missionsspezifikationen (z.B. Geschwindigkeit beim Aufsetzen). Die Struktur der Anforderungen ist in Abbildung 6-8 dargestellt. In dem Produktbaum wird die Konfiguration des Landers mit seiner Masse definiert. Werte aus diesem Bereich (z.B. gesamt Masse des Landers) werden mit Werten aus den Anforderungen an das Berechnungsmodul weitergeleitet. Dort wird die gesamte kinetische Energie berechnet, die beim Aufsetzten umgesetzt werden muss, und daraus der Anteil für die Dämpfung berechnet. Der Energieanteil der Dämpfung wird zusammen mit der Aufsetzgeschwindigkeit an den Linearmechanismus (Objekt „Zentra-

ler Dämpfer") weitergeleitet, wo der notwendige Verstellweg berechnet wird. Das gesamt Modell ist in Abbildung 6-10 dargestellt.

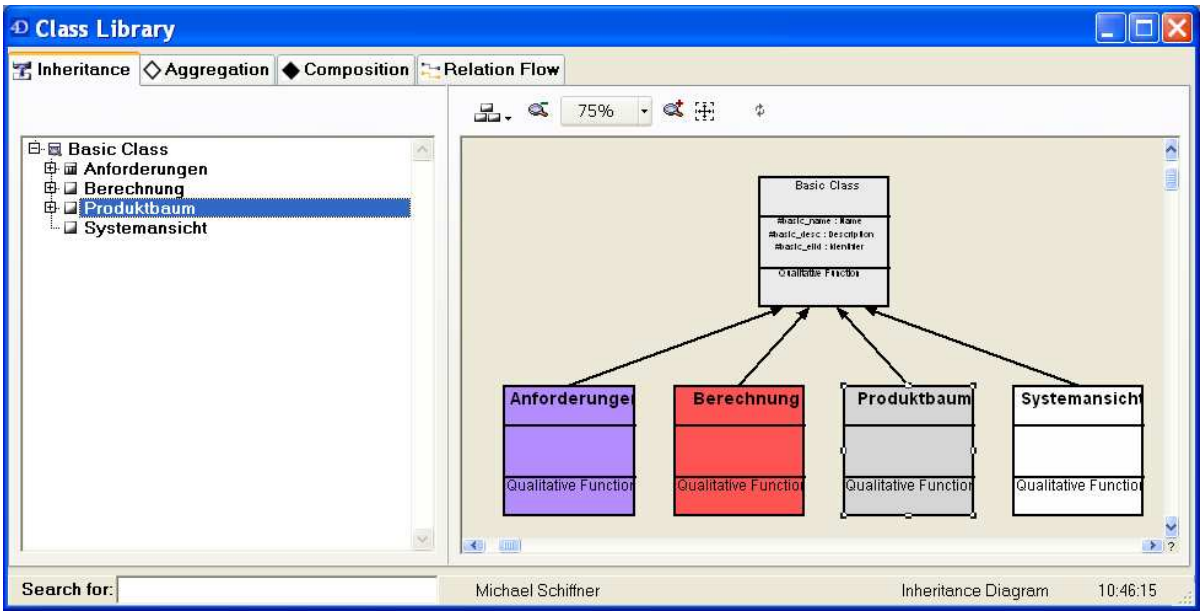


Abbildung 6-7 Klassen Diagramm: Erste Aufgliederung des Klassenbaums

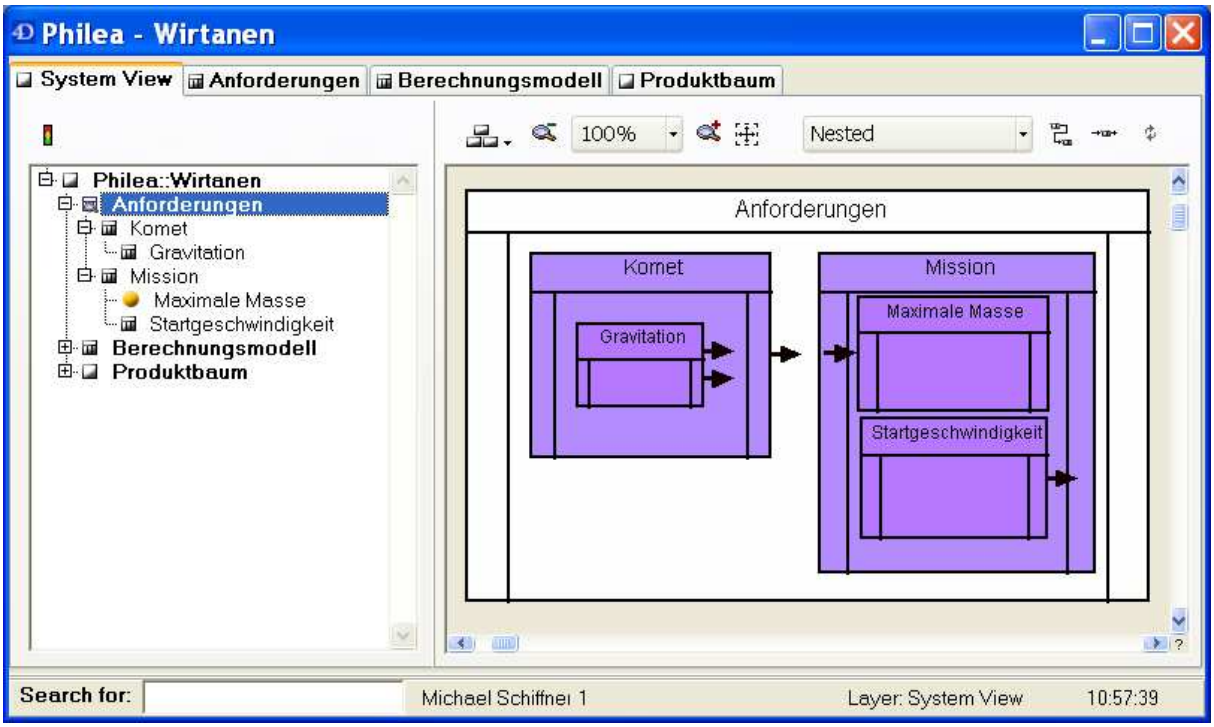


Abbildung 6-8 System Diagramm: Ausschnitt auf die Anforderungen

Die in Abbildung 6-8 dargestellte Anforderung „Maximale Masse" dient als Beispiel für die Integration einer Verifikation für eine Anforderung in das Modell. In dieser Anforderung wird

die Gesamtmasse des Landers aus dem Produktbaum ausgelesen und mit der maximal zulässigen Gesamtmasse verglichen. Für den Vergleich besitzt das Objekt eine Funktion, um zu bestimmen, ob die jeweilige Anforderung erfüllt ist oder nicht. Im diesem Fall reicht eine einfache Fallunterscheidung aus, die überprüft ob die Systemmasse kleiner oder gleich der erlaubten Gesamtmasse ist. Durch Setzen eines entsprechenden Parameters kann das Objekt anzeigen, ob die Anforderung erfüllt ist oder nicht. Dazu wurde der Datentyp „Requirement Fulfilled“ vom Typ Aufzählung mit den Aufzählungswerten „Fulfilled“ und „Not Fulfilled“ definiert.

Der in Abbildung 6-8 dargestellte gelbe Indikator im Systembaum vor dem Objekt „Maximale Masse“ zeigt dem Ingenieur eine Inkonsistenz zwischen dem verwendeten Inputwert und dem aktuell vorhanden Outputwert an. Ein roter Indikator signalisiert eine fehlende Inputrelation. Somit hat der Ingenieur eine schnelle Übersicht über die Objekte im Modell, die noch offene Schnittstellen oder Inkonsistenzen haben.

Zum Abschluss des Abschnittes über die Modellierung des Beispielsystems ist in Abbildung 6-9 der verwendete Klassenbaum dargestellt.

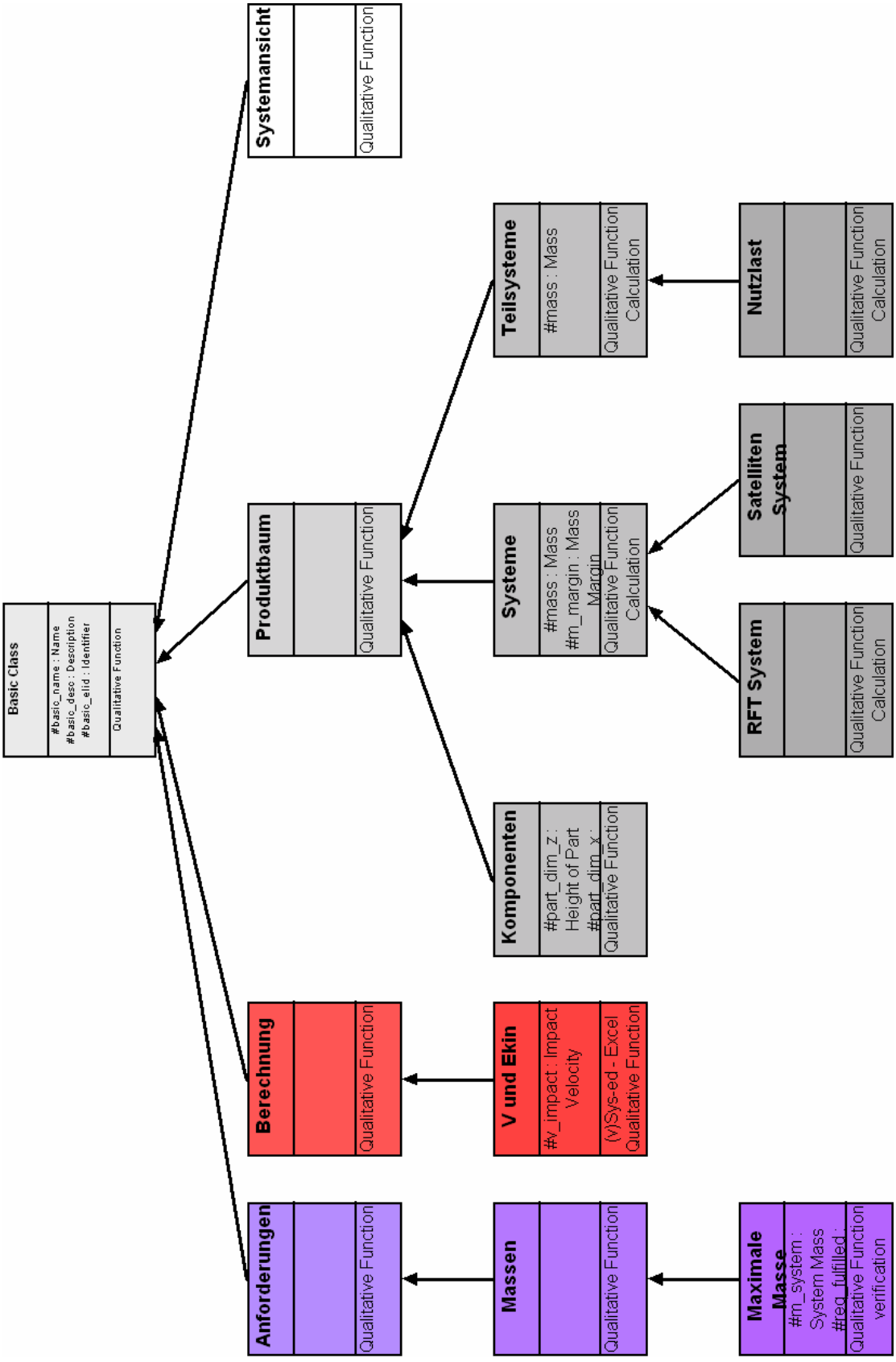


Abbildung 6-9: Verwendeter Klassenbaum für das Beispiel Philae

Analyse des Systemmodells

Abbildung 6-10 zeigt neben der Struktur des Modells bereits durch die gelb hervorgehobenen Objekte die Auswirkungen auf das System durch den neuen Kometen. Dazu wird in dem Objekt „Comet Gravity“ die Gravitationskonstante verändert. Die Zunahme der Gravitation durch den neuen Kometen hat eine erhöhte Geschwindigkeit beim Aufsetzten auf dem Kometen und somit eine höhere kinetische Energie zur Folge. Die höhere kinetische Energie bedingt einen veränderten Dämpfungsweg. Somit muss eine längere Dämpfungsstrecke in dem System berücksichtigt werden. Dies wiederum hat eine größere Masse mit den weiteren Implikationen zur Folge. Wie sich die Änderung der Anforderung durch das System fortpflanzt, kann in diesem Beispiel relativ einfach nachvollzogen werden. Bei komplexeren Systemen ist diese Ausbreitung von Änderungen durch das System nicht mehr so leicht zu identifizieren. Dazu bietet die ISM Methodik die Möglichkeit, diese Fortpflanzung von Änderungen im System zu analysieren und dem Entwickler zu verdeutlichen. Abbildung 6-11 zeigt dies für den dargestellten Fall.

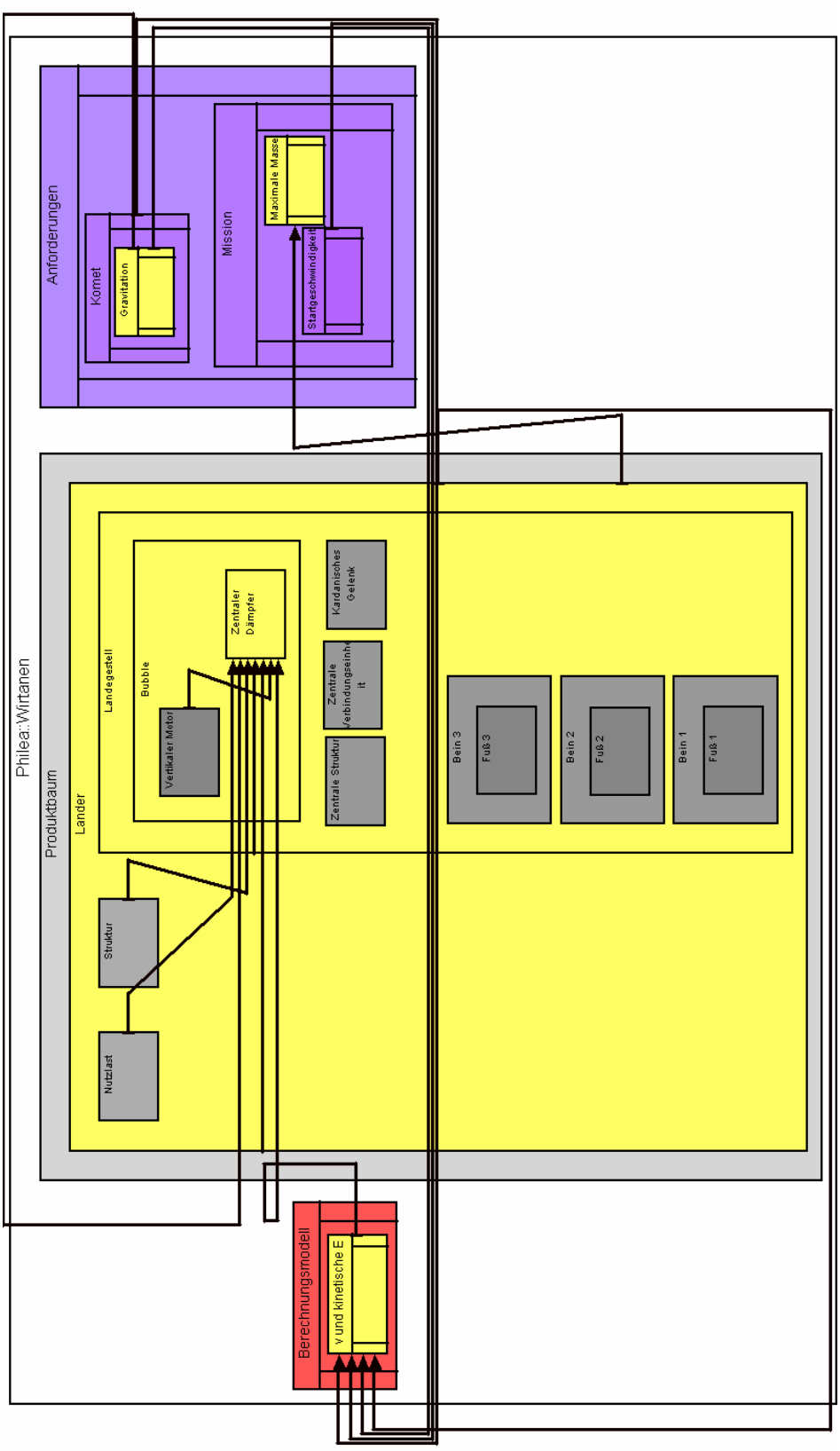


Abbildung 6-10 Analyse der Auswirkungen (gelbe Objekte) auf das System durch die veränderte Kometengravitation („Comet Gavity“)

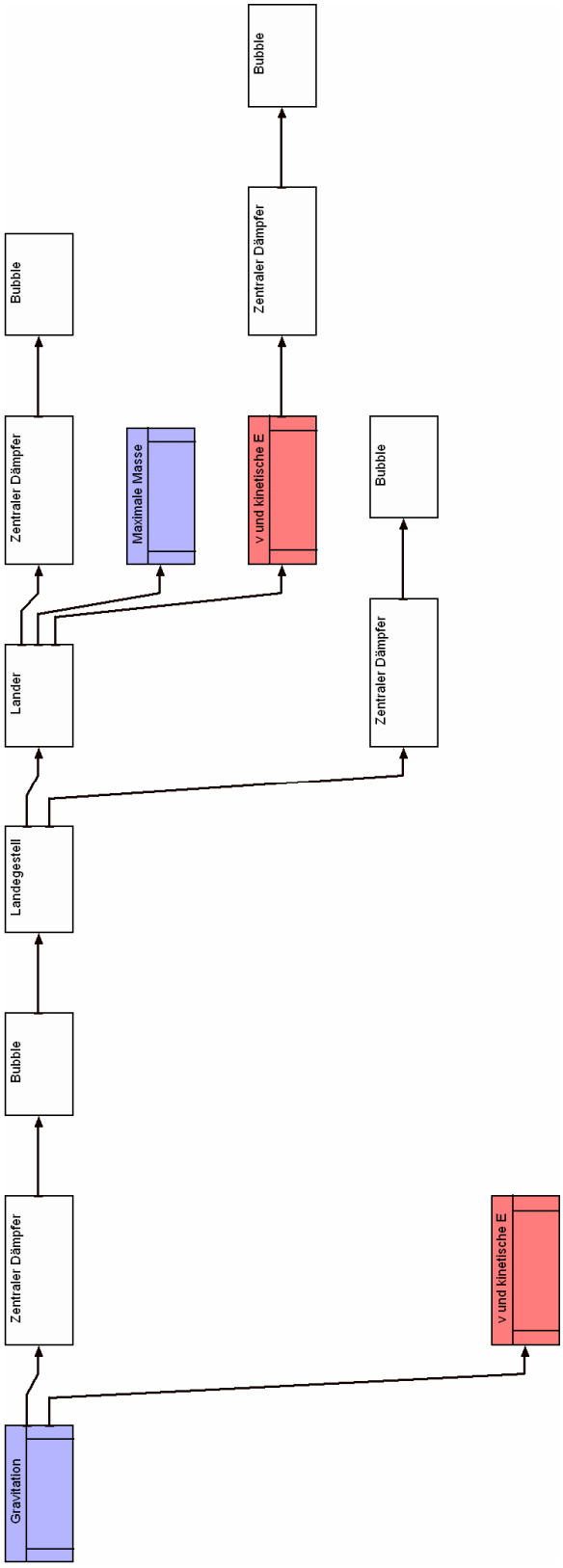


Abbildung 6-11 Darstellung der Fortpflanzungen von Änderungen in dem System durch geänderte Kometengravitation

Führt man eine Sensitivitätsanalyse durch eine Variation der Gravitation des Kometen durch, kann man die benötigte Dämpferlänge bei verschiedenen Kometen bei gleichbleibender Orbithöhe von Rosetta herausfinden. Dazu wurde im Modell eine Lookup-Tabelle (Abbildung 6-12) für verschiedene Kometengravitationen bei einer Startgeschwindigkeit des Landevorgangs von $0\frac{m}{s}$ angelegt. Bei der Berechnung wurden ein Startradius von 5000*m* und ein Kometenradius von 2000*m* angenommen.

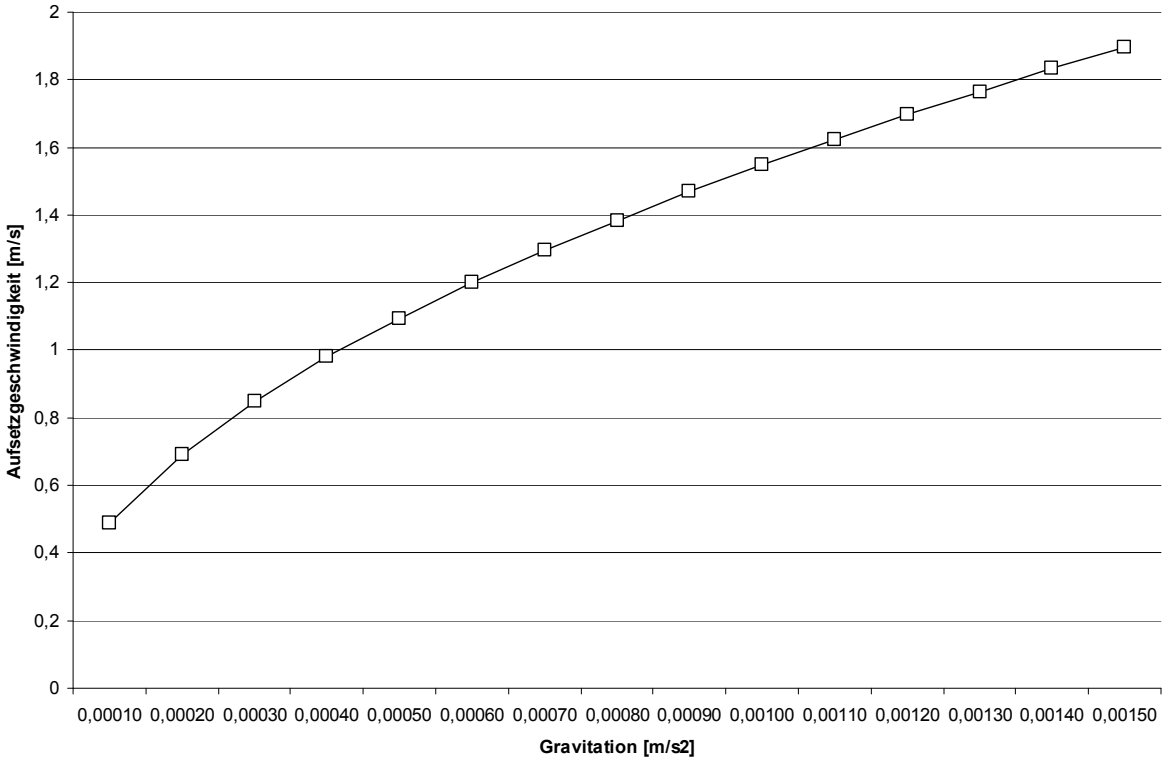


Abbildung 6-12 Aufsetzgeschwindigkeit in Abhängigkeit der Kometengravitation bei einer angenommenen Startgeschwindigkeit von $0\frac{m}{s}$, einer Orbithöhe von 5000*m*, einem Kometenradius von 2000*m* und einem senkrechten Abstieg

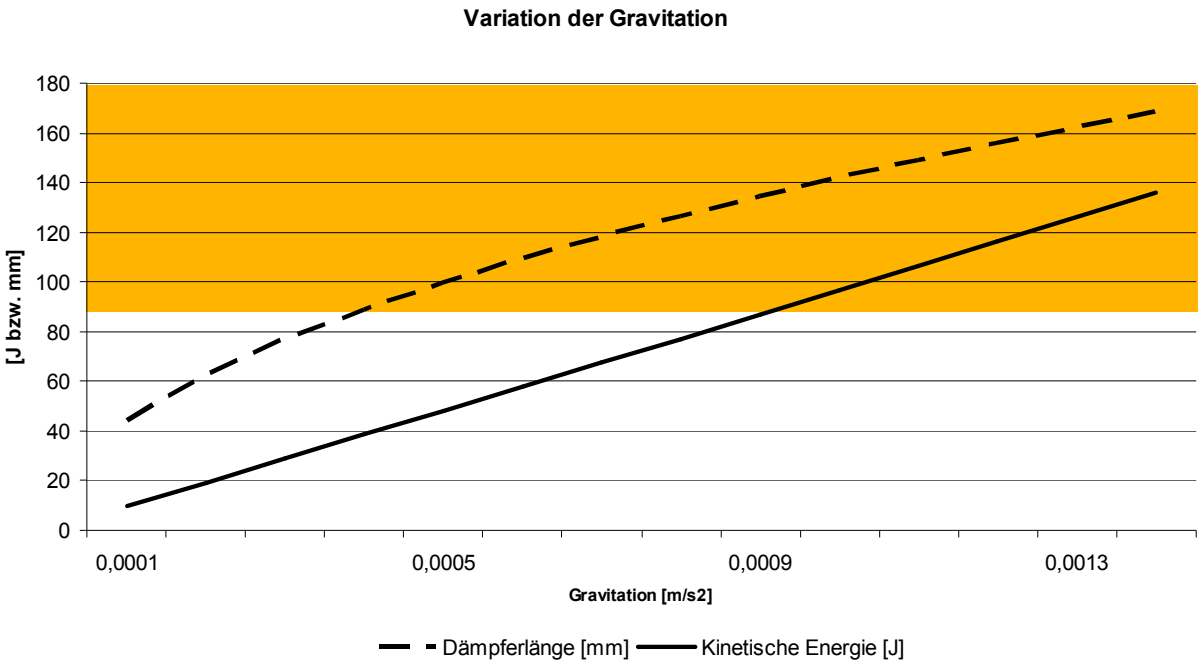


Abbildung 6-13 Veränderung der Dämpferlänge durch den neuen Kometen. Der orange Hintergrund kennzeichnet den Bereich des Dämpfungsversagens

Abbildung 6-13 zeigt die Auswirkung der veränderten Gravitation auf die notwendige Dämpferlänge. Die Linearisierung bei dem Verlauf der kinetischen Energie entsteht durch den Anstieg der Lander Masse durch die Zunahme der Dämpferlänge. Aus den zugrundeliegenden Informationen ist die maximale Dämpferlänge nicht absolut definiert sondern durch den maximal Fall $v_l = 1{,}0\frac{m}{s}$ und $\alpha = 30°$ definiert. Aus Abbildung 6-12 ist erkennbar, dass die maximale Aufsetzgeschwindigkeit bei dem neuen Zielkometen ($g = 0{,}001\frac{m}{s^2}$) ab einer Oberflächengravitation von $g = 0{,}0004\frac{m}{s^2}$ überschritten wird. Der Bereich des Versagens ist in Abbildung 6-13 durch den orangen Hintergrund hervorgehoben. Da die Dämpferlänge bei Philae nicht mehr geändert werden kann, stellt sich die Frage, wie man bei Beibehalten der bestehenden Linearverstellung auf Grund der veränderten Randbedingungen trotzdem sicher auf dem Kometen landen kann. Dazu sind alle Einflussfaktoren auf die Dämpfung zu identifizieren. Dies kann mit dem Modell recht einfach über eine Untersuchung der Abhängigkeiten aus der Matrix Darstellung des Funktionsblocks und den Relationen erfolgen. Abbildung 6-14 zeigt das Ergebnis dieser Suche nach den unabhängigen Attributen. Es wird nach Attributen

und nicht nur Parametern gesucht, da der Modellierer unter Umständen Inputs ohne Relationen oder Outputs ohne Berechnungsmethodik in dem Modell verwendet. In Abbildung 6-15 wird der Fluss der Änderungen durch das Modell dargestellt. Dieses Mal aber in umgekehrter Darstellung.

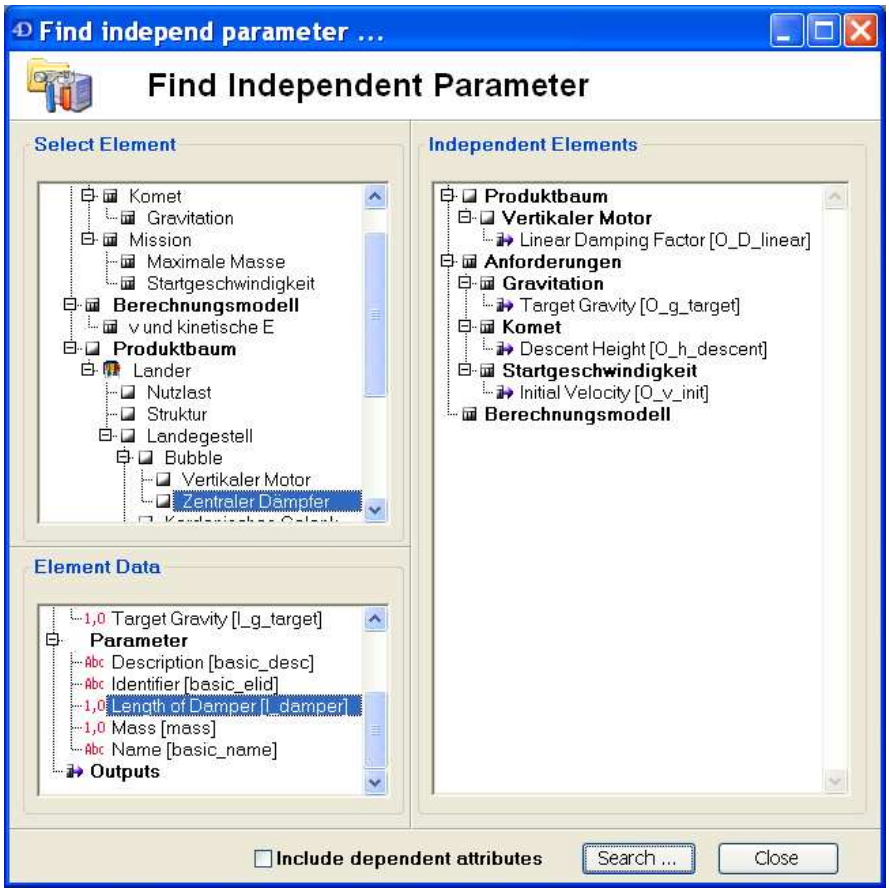


Abbildung 6-14 Suche nach unabhängigen Attributen im System für die Länge des Dämpfungsmechanismuses

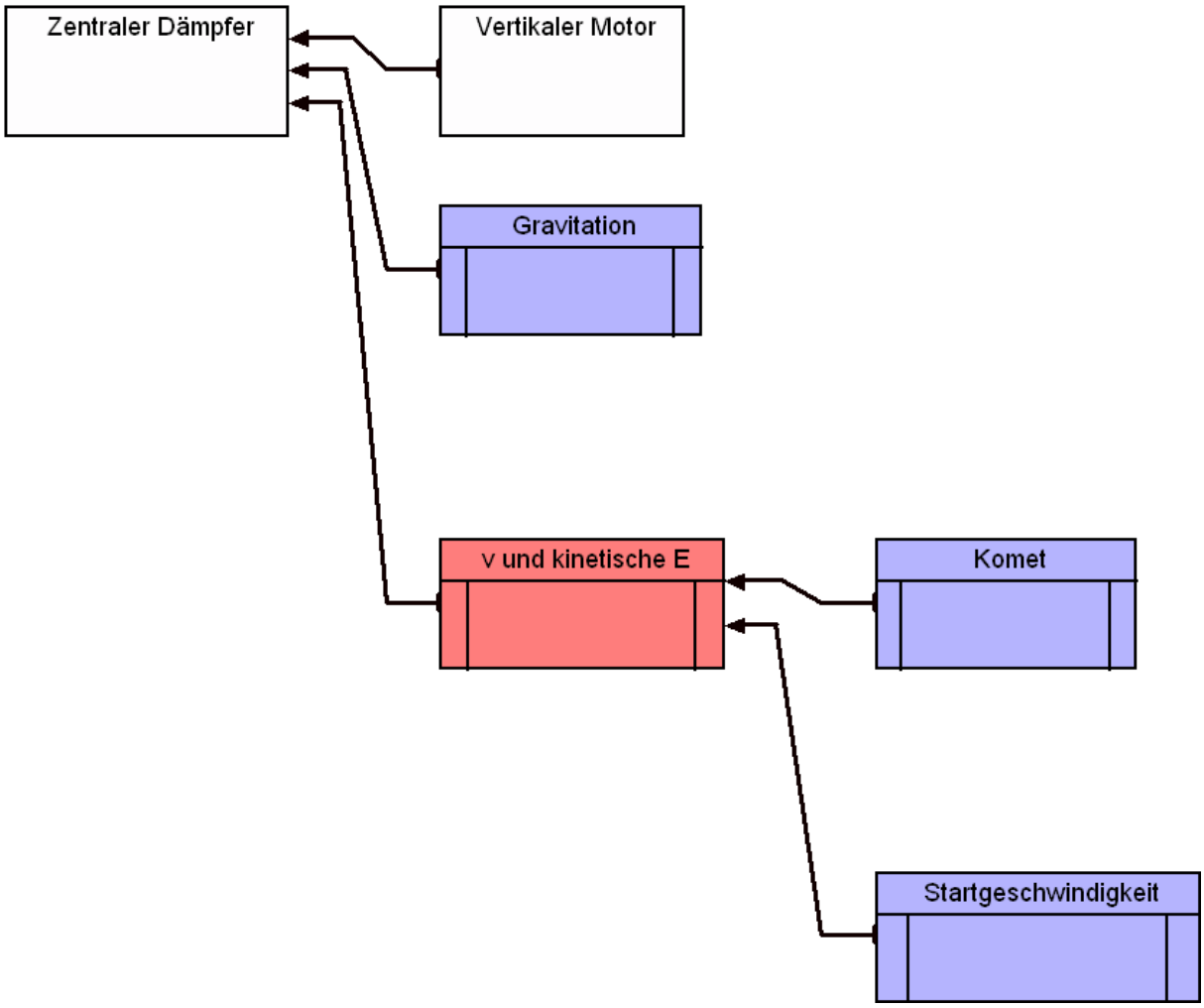


Abbildung 6-15 Graphische Darstellung der Abhängigkeit der Dämpferlänge von den unabhängigen Attributen

Es zeigt sich durch die Analyse, dass der Entwickler drei verschiedene Möglichkeiten für die Beeinflussung der Dämpferlänge hat:

1. Veränderung der Gravitationskonstante
2. Die Anfangsgeschwindigkeit für den Abstieg vermindern
3. Von einem niedrigeren Orbit starten
4. Variation des Dämpfungsfaktors

Die Gravitation des Kometen kann in diesem Fall natürlich nicht verändert werden, da sich gerade dieser Faktor durch den neuen Zielkometen geändert hat.

Eine Variation der Anfangsgeschwindigkeit ist auch nicht sinnvoll, da dafür die notwendigen Informationen wie zum Beispiel Starthöhe fehlen. Für die Analyse wird eine Aufsetzgeschwindigkeit von 1,2 m/s basierend auf den Daten aus den Veröffentlichungen angenommen.

Eine Variation der Orbithöhe ist nur bedingt sinnvoll, da die Orbithöhe von den Missionsanforderungen durch den Rosetta Satelliten definiert ist und von der Gravitation des Kometen abhängig ist.

Die letzte Möglichkeit der Variation liegt in der Veränderung des Dämpfungsfaktors, der sich durch den Motor ergibt. Die Auswirkung der Dämpferlänge durch Variation des Dämpfungsfaktors ist in Abbildung 6-16 gezeigt.

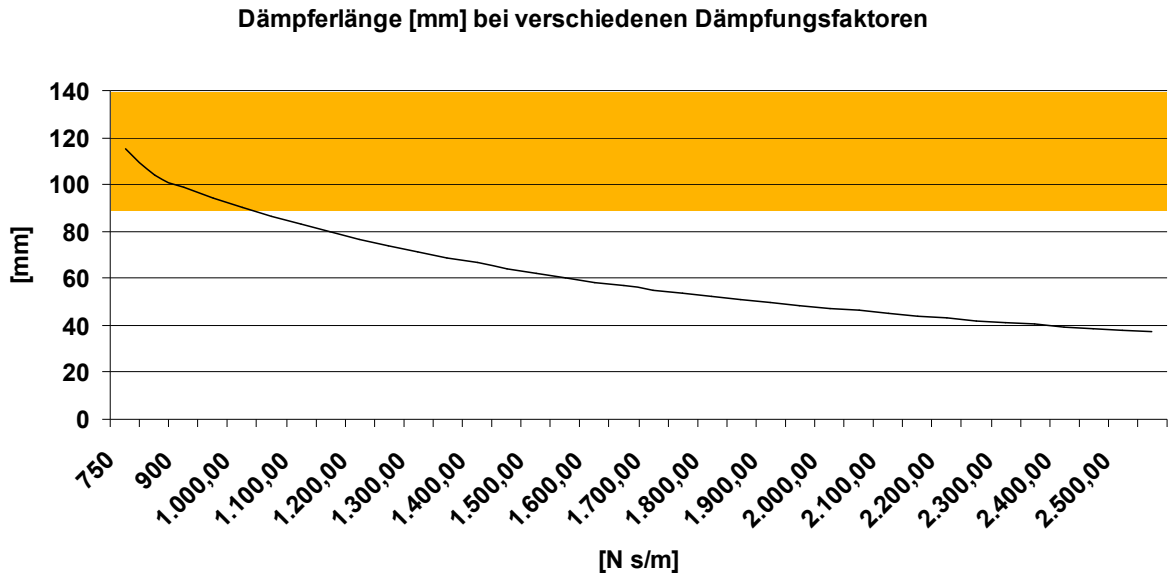


Abbildung 6-16 Veränderung der Dämpferlänge durch Variation der Dämpferkonstante. Der orange Hintergrund kennzeichnet den Bereich des Dämpfungsversagens.

Abbildung 6-16 zeigt die Abnahme der benötigten Dämpferlänge bei gleichzeitigem Anstieg des Dämpfungsfaktors. Der Dämpfungsfaktor wird durch den Elektromotor bedingt, so dass auch bei dieser Lösung eine Veränderung des Landers notwendig ist. Gleichzeitig ist bei einer Erhöhung des Dämpfungsfaktors auch eine Veränderung der Landebeine notwendig, da der Lander sonst von der Oberfläche abprallt. Bei einer Erhöhung der Dämpfungskonstante um den Faktor 1,4 muss die Federkonstante in den Beinen um den Faktor 2 erhöht werden (Hilchenbach, Rosenbauer et al. 2004 – First Contact with a comet, S. 293). Somit ist diese Lösung auch nicht realisierbar.

Da somit alle bisher identifizierten Möglichkeiten nicht geeignet sind, die neuen Anforderungen zu erfüllen, musste eine andere Lösung gefunden werden. Man hat sich dazu entschieden,

durch zusätzliche Freiheitsgrade in dem kardanischen Gelenk zwischen Dämpfer und Landebeinen die Energie abzubauen.

Ergebnis

Es wurde an dem Beispiel gezeigt wie man mit Hilfe der neuen Methodik ein Systemmodell generieren und verschiedene Designoptionen analysieren kann. Dazu wurde in einem ersten Schritt das mechanische Problem dargestellt. Nach der Analyse der Problemstellung wurde das Systemmodell beschrieben und der verwendete Klassenbaum aufgezeigt. In dem letzten Teil wurden anhand einer Designanalyse die Auswirkungen durch die neuen Missionsanforderungen aufgezeigt. Dabei konnten vier verschiedene Einflussfaktoren identifiziert werden. Da aber alle vier Möglichkeiten durch die Rahmenbedingungen der Mission nicht mehr umgesetzt werden konnten, wurde in Realität eine andere Lösung ausgewählt. Dadurch wurden aber auch die Einschränkungen durch eine modellbasierte Entwicklung aufgezeigt. Ein Modell kann nur die Lösungen anbieten, die auch im Modell abgebildet werden.

Die Einsatzfähigkeit der vorgestellten Lösung für mehrere unterschiedliche Fachdisziplinen für die simultane Satellitenentwicklung konnte an Hand des Rosetta Beispiels nicht gezeigt werden. Diese Verifikation fand unter anderem im Rahmen des Systemtechnischen Praktikums statt, bei dem mehrere verschiedene Gruppen verschiedene Systeme modelliert haben. Bei den Systemen handelte es sich um elektrische Zahnbürsten oder auch ein Mikrowellenherd.

7 Zusammenfassung und Ausblick

Ziel der Arbeit war es, eine neue Modellierungsmethode für die integrierte, modellbasierte Satellitenentwicklung in der frühen Systementwicklungsphase zu entwickeln. Die hierbei entstandene objektorientierte Modellierungsmethode fasst Elemente aus UML, SysML und dem Münchner Modellierungsansatz zusammen. Gleichzeitig wird eine Softwarearchitektur für die Implementierung der Methodik und ein entsprechendes Datenmodell definiert. Alle Aspekte der neuen Methode sind in der Modellierungssoftware (v)Sys-ed implementiert. Die Verifikation mittels dieser Software erfolgte im Rahmen dieser Arbeit primär anhand des Rosetta Landers Philae. Daneben wird diese neue Methode bereits seit einem Jahr erfolgreich in mehreren Praktika und Studienarbeiten am Lehrstuhl verwendet. Die Arbeit gliedert sich in die beiden Hauptbereiche Analyse und Umsetzung mit Verifikation.

Analyse

Die Analyse beginnt in Kapitel 2 mit einer Einführung in die Thematik der integrierten, modellbasierten Satellitenentwicklung. Dazu wird das gesamte Umfeld bestehend aus dem Entwicklungsprozess, -team, der notwendigen Infrastruktur und den Entwicklungswerkzeugen beschrieben. Bei dem Entwicklungsprozess wird zuerst das Stage-Gate-Modell von Cooper für einen erfolgreicher Entwicklungsprozess außerhalb der Raumfahrt vorgestellt. Diesem allgemeinen Prozess wird anschließend der in der europäischen Raumfahrt wichtige Entwick-

lungsprozess ECSS-M-30A der ESA gegenüber gestellt. Zusätzlich wird die frühe Entwicklungsphase, welche das Einsatzumfeld für die hier vorgestellte Lösung ist, näher charakterisiert. Nach dem Prozessumfeld werden die Entwicklungsteams im Rahmen der ISME behandelt. Als dritter Abschnitt wird das Concurrent Design Center als Entwicklungsumgebung vorgestellt und die Implementierungen dieser bei NASA, ESA, EADS-Astrium und Universitäten gezeigt. Das Kapitel endete mit der Analyse der für den Entwurf relevanten Werkzeuge und Modelle und der Quintessenz, dass ein zentrales disziplinunabhängiges Modell dringend erforderlich ist.

In dem darauffolgenden Kapitel 3 werden die aktuellen Werkzeuge der ESA und NASA für dieses zentrale disziplinunabhängige Modell beschrieben. Neben diesen beiden Ansätzen wird die Modellierungsmethode des Lehrstuhls für Raumfahrttechnik der TU München sowie die grafischen Modellierungsansätze UML und SysML dargestellt. Neben den möglichen Modellierungsansätzen wird auf die ideale Softwarearchitektur des Entwicklungswerkzeuges für die integrierte Zusammenarbeit eingegangen. Aus diesem Kapitel resultiert der Bedarf einer objektorientierten Modellierungsmethodik, die in einer anwender-unabhängigen Client-Server Software konsekutiv realisiert wird. Die Analyse über den Einsatz von Concurrent Engineering und Concurrent Design Centern und der modellbasierten Entwicklung sowie der eingesetzten Werkzeuge wird zusätzlich mit den Erfahrungen des Autors aus dem Bereich der integrierte, modellbasierte Satellitenentwicklung in Kapitel 4 ergänzt.

Umsetzung mit Verifikation

Die neue integrierte Systemmodellierung (ISM) bestehend aus einem Modell für die Objekte und Klassen und einer graphischen Darstellung dieser in einer UML und SysML kompatiblen Form wird in Kapitel 5 vorgestellt. Zusätzlich wird eine Sammlung von Funktionalitäten, die Softwarearchitektur und ein Datenmodell definiert. Abschließend wird die Umsetzung der Methode in (v)Sys-ed, einer Client-Server Software, gezeigt.

Die Vorteile für den Entwickler bei einer Systemauslegung unter Verwendung von ISM werden am Beispiel des Landers Philae der Rosetta Mission in Kapitel 6 aufgezeigt. Bei dieser Verifikation wird ISM für die Modellierung des Systems und der anschließenden Analyse des Designs aufgrund von Änderungen der Rahmenbedingungen verwendet. Es werden dabei auch die Einschränkungen, die eine modellbasierte Systemanalyse mit sich bringt, aufgezeigt.

Ausblick

Die Modellierungsmethode und deren Umsetzung in einer Anwendersoftware bieten noch weitere Entwicklungsmöglichkeiten. So ist es theoretisch möglich eine dreidimensionale Benutzerschnittstelle basierend auf X3D, einer Weiterentwicklung der Virtual Reality Modelling Language für eine dreidimensionale Darstellung im Webbrowser, zu generieren. Die dafür notwendigen Informationen können aus dem Systemmodell extrahiert und dem Benutzer dargestellt werden (siehe Abbildung 4-3 oder Abbildung 4-4 auf Seite 68). Durch die Auswahl eines der Objekte in dieser Ansicht kann der Benutzer die Eigenschaften ansehen und ggf. verändern[14]. Da der für (v)Sys-ed verwendete Datenbank Server die Möglichkeit eines parallelen Zugriffes via einen Webserver anbietet, wäre eine Integration dieses Ansatzes einfach zu realisieren. Darüber hinaus sind weitere Schnittstellen für andere Werkzeuge vorstellbar. Dazu wird bereits an einem Datenaustausch basieren auf XML gearbeitet.

[14] Beispiele sind auf der Webseite des Web 3D Konsortiums unter http://www.web3d.org/models/ zu finden.

8 Anhänge

8.1 Literaturverzeichnis

ACE – Konstruktion ohne Grenzen ONLINE, C. I.O.: *Konstruktion ohne Grenzen - Technik - neue Lösungen - CIO - CIO.de* URL http://www.cio.de/technik/804701/index1.html. – Überprüfungsdatum 2007-06-04– IDG BUSINESS VERLAG GMBH

Aguilar, Dawdy et al. 1998 – The Aerospace Corporation's Concept Design AGUILAR, J. A. ; DAWDY, A. B. ; LAW, G. W.: The Aerospace Corporation's Concept Design Center. Bd. 8. *Proceedings of the 8th Annual International Symposium of the International Council of Systems Engineering*. , 1998. (Proceedings of the Annual International Symposium of the International Council of Systems Engineering)

Andersson 1994 – Early Design Phases and Their ANDERSSON, Peder: Early Design Phases and Their Role in Designing for Quality. 1994, Vol. 5 Issue 4. *Journal of Engineering Design*. , 1994. (Vol. 5 Issue 4), S. 283–298

Ariane-5: Learning from Flight 501 15.05.1997 *Ariane-5: Learning from Flight 501 and Preparing for 502* URL http://www.esa.int/esapub/bulletin/bullet89/dalma89.htm. – Aktualisierungsdatum: 1997-05-15. – Überprüfungsdatum 2007-05-28

Bandecchi 20.06.2006 – Probleme bei dem Austausch BANDECCHI, M.: *Probleme bei dem Austausch von Teilmodellen mit IDM.* Persönliches Gespräch, 20.06.2006. SCHIFFNER, Michael (Adressat)

Bandecchi 2001 – The ESA Concurrent Design Facility @ Finnland BANDECCHI, M.: The ESA Concurrent Design Facility (CDF) applied to space mission assessments. *Nordic Systems Engineering Seminar*. , 2001. (The Second Nordic Systems Engineering Boat Seminar)

Bandecchi, Melton et al. 2000 – The ESA/ESEC Concurrent Design Facility BANDECCHI, M. ; MELTON, B. ; GARDINI B. ; ONGARO F.: The ESA/ESEC Concurrent Design Facility. Bd. 2. *Proceedings of 2nd European Systems Engienering Conference*. München, 2000. (Proceedings of European Systems Engineering Conference), S. 329–336

Bandecchi, Robin Biesbroek 2004 – The ESTEC Concurrent Design Facility BANDECCHI, Massimo ; ROBIN BIESBROEK, Robin: *The ESTEC Concurrent Design Facility.* In: *Proceedings of the Annual International Symposium of the International Council of Systems Engineering* 1 (2004)

Cooper 2002 – Top oder Flop COOPER, Robert G.: *Top oder Flop in der Produktentwicklung : Erfolgsstrategien; von der Idee zum Launch.* 1. Aufl Weinheim : Wiley-VCH Verl., 2002 – ISBN 3527500278

DEAN, E. ; UNAL, R.: *Elements of Designing for Cost.* AIAA: *Elements of Designing for Cost.* Irvine, CA : 1992

Department of Defence (Hg.) 23.02.2001 – DoD 5000 *DoD 5000 - Terminology Review* URL http://dod5000.dau.mil/TERMS/index.htm. – Aktualisierungsdatum: 2001-02-23. – Überprüfungsdatum 2007-04-30; Seite ist umgezogen zu https://akss.dau.mil/pv/Glossary.aspx – Überprüfungsdatum 2007-12-06

Eckart, Kesselmann et al. 1999 – LunarSat AIV ECKART, Peter ; KESSELMANN, Michael ; CANALES, Martin: *LunarSat AIV.* München, 1999. – Aktualisierungsdatum: 1999

ECSS M30-A NORM M-30A. *Projektphaseneinteilung und -planung*

Ehrlenspiel 2007 – Integrierte Produktentwicklung EHRLENSPIEL, Klaus: *Integrierte Produktentwicklung : Denkabläufe, Methodeneinsatz, Zusammenarbeit.* 4., aktualisierte Aufl. München : Hanser, 2007 – ISBN 3-446-40733-2

Ehrlenspiel, Kiewert et al. 2003 – Kostengünstig entwickeln und konstruieren EHRLENSPIEL, Klaus ; KIEWERT, Alfons ; LINDEMANN, Udo: *Kostengünstig entwickeln und konstruieren : Kostenmanagement bei der integrierten Produktentwicklung.* 4., bearb. Aufl Berlin : Springer, 2003 – ISBN 3-540-44214-6

ESA 27.06.1996 – The Cluster Mission *The Cluster Mission - ESA's Space Fleet to the Magnetosphere* URL http://www.esa.int/esapub/bulletin/bullet84/credl84.htm. – Aktualisierungsdatum: 1996-06-27. – Überprüfungsdatum 2007-06-11– ESA

European Space Agency – ESA Science & Technology *ESA Science & Technology: Rosetta 3D Model* URL http://rosetta.esa.int/science-e/www/object/index.cfm?fobjectid=31389. – Überprüfungsdatum 2007-04-28– European Space Agency

European Space Agency – ESA Science & Technology *ESA Science & Technology: Rosetta* URL http://rosetta.esa.int/science-e/www/area/index.cfm?fareaid=13. – Überprüfungsdatum 2007-04-28– European Space Agency

European Space Agency – ROSETTA at a Glance *ROSETTA at a Glance* URL http://www.esa.int/SPECIALS/Rosetta/. – Überprüfungsdatum 2005-10-05– European Space Agency

European Space Agency 2006 – Proceedings of the 2nd Concurrent EUROPEAN SPACE AGENCY: *Proceedings of the 2nd Concurrent Engineering for Space Applications Work-*

shop 2006. Noordwijk, Die Niederlande : 2006 (Proceedings of European Systems Engineering Conference 2)

Extensible Markup Language 06.07.2007 *Extensible Markup Language - Wikipedia* URL http://de.wikipedia.org/wiki/Xml. – Aktualisierungsdatum: 2007-07-06. – Überprüfungsdatum 2007-06-11

Faisst Wolfgang 1995 – Welche IV-Systeme sollte ein Virtuelles FAISST WOLFGANG: Welche IV-Systeme sollte ein Virtuelles Unternehmen haben? In: EHRENBERGER, Dieter Griese Joachim Mertens Peter (Hrsg.). *Informations- und Kommunikationssysteme als Gestaltungselement Virtueller Unternehmen*. , 1995. (1)

Ferris 2003 – Review of Papers Concerning Engineering FERRIS, Thimothy L. J.: Review of Papers Concerning Engineering Teams in INCOSE Symposia 1991 to 2002. In: INTERNATIONAL COUNCIL OF SYSTEMS ENGINEERING (HRSG.). *Proceedings of the 13th Annual International Symposium of the International Council of Systems Engineering*. Washington, USA, 2003, S. 502–516

Finkel S. 2002 – Design Centers –Transferring Experiences FINKEL S., Wilke M. Metzger H. Wahnfried M.: Design Centers –Transferring Experiences from Astronautics to Aeronautics. In: INTERNATIONAL COUNCIL OF SYSTEMS ENGINEERING (HRSG.). *Proceedings of the 10th International Symposium of the International Council of Systems Engineering*. Las Vegas, 2002. (Proceedings of the Annual International Symposium of the International Council of Systems Engineering)

Fricke 1998 – Der Änderungsprozess als Grundlage FRICKE, Ernst: *Der Änderungsprozess als Grundlage einer nutzerzentrierten Systementwicklung.* München, Technische Universität München, Fakultät für Maschinenwesen, Dissertation, 1998

Fricke, Negele et al. 06.02.1998 – Grundlagen der Prozeßmodellierung FRICKE, Ernst ; NEGELE, Herbert ; SCHREPFER, Negele: *Grundlagen der Prozeßmodellierung.* München, Technische Universität München, Fachgebiet Raumfahrttechnik, Technischer Bericht, 06.02.1998

Früh – Software Entwicklung SwE im IT-Studiengang FRÜH, P. T.: *Software Entwicklung (SwE) im IT-Studiengang* URL http://home.zhwin.ch/~frp/SwE/Doku/Ood.pdf. – Überprüfungsdatum 2007-05-02

Gaudenzi, Morelli 2006 – Basic concurrent design procedures GAUDENZI, Paolo ; MORELLI, Guido: Basic concurrent design procedures for satellite systems preliminary design and for educational purpose. *Proceedings of the 2nd Concurrent Engineering for Space Applications Workshop 2006*. Noordwijk, Die Niederlande, 2006. (Proceedings of European Systems Engineering Conference, 2), S. T1-05

Grady 1993 – System requirements analysis GRADY, Jeffrey O.: *System requirements analysis.* New York : McGraw-Hill, 1993 – ISBN 00702399404495

Graves, Morgan et al. 1992 – Project Planing and proposal development GRAVES, C. A. ; MORGAN, W. C. ; SCHEFFER, B. M. ; FLEMMING R. E.: Project Planing and proposal development in an empowered team environment. In: NATIONAL COUNCIL OF SYSTEMS ENGINEERING (HRSG.). *Proceedings of the Annual Symposium NCOSE*. , 1992, S. 155–162

Hall 1962 – A Methodology for Systems Engineering HALL, Arthur D.: *A Methodology for Systems Engineering.* Princeton, New Jersey : Van Nostrand and Company, 1962

Hilchenbach, Rosenbauer et al. 2004 – First Contact with a comet HILCHENBACH, M. ; ROSENBAUER, H. ; CHARES, B.: First Contact with a comet surface: Rosetta Lander Simulations. In: L. COLANGELI ET AL. (HRSG.). *The New ROSETTA Targets*. Die Niederlande : Kluwer Academic Publisher, 2004, S. 289–296

Hilchenbach; Impact on a comet HILCHENBACH, M. ; KÜCHEMANN, O. ; ROSENBAUER, H.: *Impact on a comet: Rosetta Lander simulations.* In: *planss* 48 (2000), S. 361–369

IEEE Standard for Application and Management of the Systems Engineering Process. IEEE-SA STANDARD BOARD (HRSG.): *IEEE Standard for Application and Management of the Systems Engineering Process.* 345 East 47th Street, New York, NY 10017-2394, USA : The Institute of Electrical and Electronics Engineers, Inc., 1998, S. 11

Igenbergs 1993-2007 – Grundlagen der Systemtechnik IGENBERGS, Eduard: *Grundlagen der Systemtechnik : Vorlesung.* Vorlesungsunterlagen, (Grundlagen der Systemtechnik). München, 1993-2007. – Aktualisierungsdatum: 1993-2007

International Council of Systems Engineering (Hg.) – INCOSE *INCOSE - SE Handbook* URL http://www.incose.org/ProductsPubs/products/sehandbook.aspx. – Überprüfungsdatum 2007-06-02– International Council of Systems Engineering

Jeckle 2004 – UML 2 glasklar JECKLE, Mario: *UML 2 glasklar : Unified modeling language.* München : Hanser, 2004 – ISBN 3-446-22575-7

Krottmaier 1995 – Leitfaden Simultaneous Engineering KROTTMAIER, Johannes: *Leitfaden Simultaneous Engineering : Kurze Entwicklungszeiten, niedrige Kosten, hohe Qualität.* Berlin : Springer, 1995 – ISBN 0387586369

Mager, Hartmann 2000 – The Satellite Design Office MAGER, Rolf ; HARTMANN, Ralf: The Satellite Design Office at Astrium — A Success Story of an Industrial Design Center Application. *Proceedings of 2nd European Systems Engienering Conference*. München, 2000. (Proceedings of European Systems Engineering Conference), S. 293–302

Mark G.: Extreme collaboration GLORIA MARK: *Extreme collaboration.* In: *Commun. ACM* 45 (2002), Nr. 6, S. 89–93

McInnes A. 2003 – Integrated Concurrent Design MCINNES A.: *Integrated Concurrent Design.* USA, 2003. – Aktualisierungsdatum: 2003

McInnes, Lang et al. 2003 – Integrated Data Exchange Architecture IDEA MCINNES, A. ; LANG, Jeff ; NUSSBAUMER, Eric: *Integrated Data Exchange Architecture (IDEA) : Software Users and Developers Guide.* THE AEROSPACE CORPORATION (HRSG.) 2003

Negele 1998 – Systemtechnische Methodik zur ganzeitlichen Modellierung NEGELE, Herbert: *Systemtechnische Methodik zur ganzeitlichen Modellierung am Beispiel der integrierten Produktentwicklung.* München, Technische Universität München, Fakultät für Maschinenwesen, Dissertation, 1998

Negele, Fricke et al. – Modelling of Integrated Product Development NEGELE, Herbert ; FRICKE, Ernst ; SCHREPFER, Lutz ; HÄRTLEIN, Nicole: Modelling of Integrated Product Development Processes. *Proceedings of the 9th Annual Symposium of INCOSE*

Oberg Dez. 1999 – Why the Mars probe [accident OBERG, J.: Why the Mars probe [accident investigation]. Bd. 12. *Spectrum*. Institute of Electrical and Electronics Engineers (IEEE), Dez. 1999. (36). – ISBN 0018-9235, S. 34–39

Object Management Group (Hg.) Februar 2007 – Unified Modeling Language OBJECT MANAGEMENT GROUP: *Unified Modeling Language: Superstructure : version 2.1.1 (non-change bar).* Februar 2007. – formal/07-02-05

Object Management Group (Hg.) Mai 2006 – OMG Systems Modeling Language OMG OBJECT MANAGEMENT GROUP: *OMG Systems Modeling Language (OMG SysMLTM) : Final Adopted Specification.* Mai 2006. – ptc/06-05-04

OMG SysML *OMG SysML* URL http://www.omgsysml.org/. – Überprüfungsdatum 2007-06-04

Otto, Wood 2001 – Product design OTTO, Kevin N. ; WOOD, Kristin L.: *Product design : Techniques in reverse engineering and new product development.* Upper Saddle River, NJ : Prentice Hall, 2001 – ISBN 0-13-021271-7

Parkin, Sercel et al. 2003 – Icemaker™: an Excel-based environment PARKIN, K.L.G. ; SERCEL, J.C. ; LIU, N.J. ; THUNNISSEN, D.P.: Icemaker™: an Excel-based environment for collaborative design. In: PROFET, Robert A. (Hrsg.). *2003 IEEE Aerospace Conference proceedings : Big Sky, Montana, March 8 - 15, 2003*. Piscataway, NJ : IEEE Operations Center, 2003. – ISBN 078037651, S. 8:3669 - 8:3679

Perez 2004 – Ariane 5 PEREZ, Edouard: *Ariane 5 : User's Manual.* ARIANESPACE (HRSG.); PEREZ, Edouard (Mitarb.): 2004

Programmierschnittstelle 27.04.2007 *Programmierschnittstelle - Wikipedia* URL http://de.wikipedia.org/wiki/Programmierschnittstelle. – Aktualisierungsdatum: 2007-04-27. – Überprüfungsdatum 2007-05-07

Quirmbach 2001 – Integration der System- und Kostenentwicklung QUIRMBACH, Oliver: *Integration der System- und Kostenentwicklung am Beispiel eines Satellitensystems.* München, Technische Universität München, Fakultät für Maschinenwesen, Dissertation, 2001

Schiffner, Kuss 2006 – Results of an international survey SCHIFFNER, Michael ; KUSS, Thimo: Results of an international Survey of the Implementation of Concurrent Design Centers. *Proceedings of the 2nd Concurrent Engineering for Space Applications Workshop 2006*. Noordwijk, Die Niederlande, 2006. (Proceedings of European Systems Engineering Conference, 2)

Schneider, Werner et al. 2001 – Taschenbuch der Informatik SCHNEIDER, Uwe ; WERNER, Dieter ; EBERT, Joachim: *Taschenbuch der Informatik : Mit 114 Tabellen.* 4., aktualisierte Aufl München : Fachbuchverl. Leipzig im Carl-Hanser-Verl., 2001 – ISBN 3-446-21753-3

Sercel 1998 – Introduction to Integrated Concurrent Engineering SERCEL, J.C.: *Introduction to Integrated Concurrent Engineering (ICE) : Developed by the Caltech Laboratory for Spacecraft and Mission Design.* Cambridge, MA, USA, 1998 URL http://www.lsmd.caltech.edu/faq/ICEpresentation.pdf. – Aktualisierungsdatum: 1998. – Überprüfungsdatum 2007-05-07

Sercel, Clymer et al. 1999 – The Product Attributes Database PAD SERCEL, J.C. ; CLYMER, T. F. ; HEIKICHS, W. M.: The Product Attributes Database (PAD) : First of a New Class of Productivity Tools for Product Development. In: PROFET, Robert A. (Hrsg.). *Proceedings*. Piscataway, NJ : IEEE Service Center, 1999. – ISBN 0780354265, S. 285–299

SHISHKO, Robert: *NASA Systems Engineering Handbook.* NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (HRSG.) NATIONAL AERONAUTICS AND SPACE ADMINISTRATION: *NASA Systems Engineering Handbook.* 1995 URL http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19960002194_1996102194.pdf

Snow, Miles et al. 1992/0 – Managing 21st century network organizations SNOW, Charles C. ; MILES, Raymond E. ; COLEMAN, Henry J.: *Managing 21st century network organizations.* In: *Organizational Dynamics* 20 (1992/0), Nr. 3, S. 5–20

Stagne 2003 – The Integrated Concurrent Enterprise STAGNE, David B.: *The Integrated Concurrent Enterprise.* Cambridge, MA, USA, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics and the Sloan School of Management, Master Thesis, 2003

Walther – Systemtechnisches Verfahren zur Bestimmung WALTHER, Christian: *Systemtechnisches Verfahren zur Bestimmung der Zusammenhänge zwischen Eigenschaften und Funktionsstruktur technischer Systeme.* München, Technische Universität München, Fakultät für Maschinenwesen, Dissertation

Wilke, Quirmbach et al. 1999 – Modell der integrierten System WILKE, Martin; QUIRMBACH, Oliver; SCHIFFNER, Michael: *Modell der integrierten System- und Kostenentwicklung in MuSSat : RT-TB 99/06.* München, 1999. – Technischer Bericht

Wilke 2002 – Integrierter modellbasierter Satellitenentwurf WILKE, Martin: *Integrierter modellbasierter Satellitenentwurf.* München, Technische Universität München, Fakultät für Maschinenwesen, Dissertation, 2002

Wilke, Finkel et al. 2003 – A video approach to knowledge WILKE, Martin ; FINKEL, Stephan ; ZENDER, Joe ; SCHWEHM, Gerhard: A video approach to knowledge management. In: INTERNATIONAL COUNCIL OF SYSTEMS ENGINEERING (HRSG.). *Proceedings of the 13th Annual International Symposium of the International Council of Systems Engineering*. Washington, USA, 2003

Winner 1988 – The role of concurrent engineering WINNER, R. I.: *The role of concurrent engineering in weapons system acquisition.* Alexandria, Va : Institute for Defense Analysis, 1988 – ISBN 19681506

Yazdi, Mussat et al. 2006 – Astrium's Satellite Design Office YAZDI, Kian ; MUSSAT, Philippe ; BLEIF, Joern ; BANTING, Jonathan ; TATARD, Christophe: Astrium's Satellite Design Office : Current State, Vision and Way Forward. *Proceedings of the 2nd Concurrent Engineering for Space Applications Workshop 2006*. Noordwijk, Die Niederlande, 2006. (Proceedings of European Systems Engineering Conference, 2)

Young-Keun, Ki-Lyong et al. 2005 – Development of System Engineering Design YOUNG-KEUN, Chang ; KI-LYONG, Hwang ; SUK-JIN, Kang ; BYUNG-YOUNG, Moon ; SU-JEONG, Kim ; JIN-SOO CHANG: Development of System Engineering Design Tool (SEDT) for small satellite conceptual design. *Proceedings of the 5th IAA Symposium on Small Satellites for Earth Observation*. , 2005. (Symposium on Small Satellites for Earth Observation)

8.2 Abbildungsverzeichnis

8.3 Tabellenverzeichnis