Technische Universität München
Ingenieurfakultät Bau Geo Umwelt
Lehrstuhl für Computergestützte Modellierung und Simulation

Entwicklung eines Prototyps für den Einsatz von Knowledge-based Engineering in frühen Phasen des Brückenentwurfs

Dominic Singer

Matrikelnummer: [redacted]

Masterthesis im Studiengang Bauingenieurwesen

Prüfer:	Prof. Dr.-Ing. André Borrmann
Betreuer:	Julian Amann M.Sc.
	Dipl.-Ing. Markus Hochmuth
Ausgegeben am:	01.07.2014
Eingereicht am:	28.11.2014

„Überall geht ein frühes Ahnen dem späteren Wissen voraus.“

[Earlier intuition always precedes later knowledge.]

Friedrich Wilhelm Heinrich Alexander Freiherr von Humboldt

Vorwort

Die vorliegende Arbeit ist am Lehrstuhl für Computergestützte Modellierung und Simulation an der Technischen Universität München in Kooperation mit der Firma Obermeyer Planen + Beraten GmbH entstanden.

Mein besonderer Dank gilt dem Inhaber des Lehrstuhls für Computergestützte Modellierung und Simulation, Herrn Prof. Dr.-Ing. André Borrmann für die Anregung des Themas Knowledge-based Engineering und seine Betreuung im Rahmen der Arbeit.

Mein weiterer Dank gilt den Betreuern Julian Amann M.Sc. und Dipl.-Ing. Markus Hochmuth. Durch ihre stetige Unterstützung haben sie maßgeblich zum Gelingen dieser Arbeit beigetragen.

Allen Mitarbeitern der Lehrstühle Computergestützte Modellierung und Simulation sowie Computation in Engineering, im Besonderen Nevena Perovic M.Sc., Dipl.-Math. Matthias Flurl, Dipl.-Ing. Javier Ramos Jubierre M.Sc. und Fabian Ritter M.Sc. möchte ich für die gute Arbeitsatmosphäre und wertvollen Diskussionen danken.

Abschließend bedanke ich mich bei meinen Korrekturlesern Brigitte-Katharina Zank und Ursula Wegener.

Abstract

In infrastructure construction, different interests of builders, population, authorities and contractors, difficult technical constraints and the growing number of stakeholders lead to more and more complex designs. In the course of this, many decisions made by civil engineers, ensure a balance of functionality, operability, maintainability, sustainability, aesthetics, time and costs of a structure. Therefore, engineers make use of their experiences and use estimations for decision-making.

For design and construction of civil engineering structures such as tunnels and bridges, engineers use already extensive (parametric) CAD-based tools. Nevertheless, it is not possible within these tools to map the expertise and experience of the engineers and reuse it for similar issues. By the adaption of knowledge-based engineering for infrastructure planning, repetitive processes can be automated in early design phases of bridge and tunnel planning. This is possible by capture, storage and reutilization of engineering knowledge.

This work investigates the application of knowledge-based engineering for the design of infrastructure in the example of a girder bridge. In addition to issues of knowledge representation and modeling in knowledge-based systems, methods for the development of KBE systems are presented. Furthermore, existing commercial KBE systems are examined for their applicability to the rule-based planning of infrastructure.

The necessary rules for the KBE prototype development for bridge design are identified and addressed. Finally, the implementation of a KBE system in Dynamo and Autodesk Revit is shown, which ensures the feasibility of KBE adaptation for the planning of infrastructure.

Zusammenfassung

Unterschiedlichste Interessen der Bauherren, Bevölkerung, Genehmigungsbehörden und Ausführenden, schwierige technische Randbedingungen und die wachsende Zahl der Projektbeteiligten haben eine in jeder Hinsicht komplexere Planung von Infrastrukturbauwerken zur Folge. Viele Entscheidungen die Bauingenieure treffen, stellen ein ausgewogenes Verhältnis von Funktionalität, Ausführbarkeit, Erhaltung, Ressourcenverbrauch, Ästhetik, Termine und Kosten eines Bauwerks sicher. Die Beteiligten machen dabei Gebrauch von ihren Erfahrungen und nutzen Abschätzungen zur Entscheidungsfindung.

Für den Entwurf beziehungsweise die Konstruktion von Ingenieurbauwerken wie Tunnel und Brücken stehen dem Bauingenieur bereits umfangreiche (parametrische) CAD-basierte Werkzeuge zur Verfügung. Mit diesen Planungstools ist es allerdings nicht möglich, die Expertise und Erfahrung der Ingenieure abzubilden und bei ähnlichen Fragestellungen wiederzuverwenden. Durch die Anwendung von Knowledge-based Engineering im Infrastrukturbau können repetitive Planungsprozesse in frühen Entwurfsphasen der Brücken- und Tunnelplanung automatisiert werden, indem dieses Ingenieurwissen erfasst, gespeichert und für ähnliche Aufgaben erneut herangezogen wird.

In dieser Arbeit wird am Beispiel einer Balkenbrücke in Massivbauweise die Anwendung von Knowledge-based Engineering für den Entwurf im Infrastrukturbau untersucht. Neben Fragestellungen der Wissensrepräsentation und -modellierung in Wissensbasierten Systemen, werden Methoden zur Entwicklung von KBE Systemen vorgestellt. Anschließend werden vorhandene kommerzielle KBE Systeme auf ihre Anwendbarkeit für die regelbasierte Planung von Infrastrukturbauwerken untersucht.

Die für die Entwicklung des KBE Prototypen nötigen Entwurfsregeln der Brückenkonstruktion werden identifiziert und aufbereitet. Davon ausgehend zeigt die vorgestellte Implementierung eines KBE Prototypen in Dynamo und Autodesk Revit schließlich die prinzipielle Machbarkeit der KBE Adaption für die Planung von Infrastrukturbauwerken auf.

Inhaltsverzeichnis

Seite

Abbildungsverzeichnis Seite

Abkürzungsverzeichnis

AI	Artificial Intelligence
AIT	Advanced Information Technology in Design and Manufacturing
AML	Adaptive Modeling Language
API	Application Programming Interface
BASt	Bundesanstalt für Straßenwesen
BIM	Building Information Modeling
BKM	Building Knowledge Model
BMVI	Bundesministerium für Verkehr und digitale Infrastruktur
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CATIA	Computer Aided Three-Dimensional Interactive Application
CFD	Computational Fluid Dynamics
DBMS	Database Management Systems
DS	Dassault Systemes
ES	Expert System
ETO	Engineer-To-Order
EUC	End-User Computing
FEM	Finite-Element-Methode
GDL	General-purpose Declarative Language
GUI	Graphical User Interface
GVP	Generative Virtual Prototype
HAO	Highway Alignment Optimization
HLCts	High Level CAD templates
ICAD	Intelligent Computer-Aided Design
ICARE	Illustration Constraint Activity Rule Entity

IDL	Intelligent Declarative Language ICAD Design Language
IFC	Industry Foundation Classes
KADS	Knowledge Acquisition Documentation and Structuring
KBE	Knowledge-based Engineering
KBS	Knowledge-based Systems
KF	Knowledge Fusion
KI	Künstliche Intelligenz
KM	Knowledge Management
KMU	Klein- und Mittelständische Unternehmen
KNOMAD	Knowledge Nurture for Optimal Multidisciplinary Analysis and Design
KTI	Knowledge Technologies International
LISP	List Processing Language
MDO	Multidisciplinary Design Optimization
MIT	Massachusetts Institute of Technology
MML	MOKA Modelling Language
MOKA	Methodology and tools Oriented to Knowledge-based Engineering Applications
OMG	Object Management Group
OSM	Open Street Map
PLM	Product Lifecycle Management
PTC	Parametric Technology Corporation
UML	Unified Modelling Language
WBS	Wissensbasierte Systeme
XML	Extensible Markup Language

1 Einführung

1.1 Motivation und Problemstellung

Bereits das Römische Reich verfügte über herausragende Infrastrukturbauwerke wie Brücken, Tunnel oder Aquädukte, um ein weit verzweigtes Straßennetz zu ermöglichen und eine fortschrittliche Wasserversorgung sicherzustellen. Diese Infrastruktur bildete die Grundlage des wirtschaftlichen Erfolgs des Römischen Reichs. Heutzutage nutzen wir unsere Infrastruktur ganz selbstverständlich und werden uns ihrem Stellenwert erst bei Straßen- oder Brückensperrungen oder dem Ausfall von Strom- oder Wasserversorgung bewusst. Die Infrastruktur als Rückgrat unseres täglichen Lebens und unserer Volkswirtschaften ist von enormer Bedeutung für unsere Gesellschaft. Auch zur Sicherung der Wettbewerbsfähigkeit des Technologiestandorts Deutschland ist eine funktionierende Infrastruktur unabdingbar. Durch Erhaltung sowie Aus- und Neubau von Infrastrukturbauwerken, stellen Bauingenieure den Standortfaktor Infrastruktur sicher.

Diese Aufgabe stellt die Planer vor große Herausforderungen. Insbesondere in den Großstädten und Megacitys unserer Welt wird die Planung von Infrastrukturbauwerken zunehmend komplizierter, wie das Bahnprojekt Stuttgart 21 und andere Projekte eindrucksvoll belegen. Unterschiedlichste Interessen der Bauherren, Bevölkerung, Genehmigungsbehörden und Ausführenden, schwierige technische Randbedingungen und die wachsende Zahl der Projektbeteiligten haben eine in jeder Hinsicht komplexere Planung von Infrastrukturbauwerken zur Folge. Die in der Planung solcher Projekte auftretenden Anforderungen, verlangen eine große Anzahl an Ingenieurs- und Managemententscheidungen.

Viele Entscheidungen die Bauingenieure dabei treffen, stellen ein ausgewogenes Verhältnis von Funktionalität, Ausführbarkeit, Erhaltung, Ressourcenverbrauch, Ästhetik, Termine und Kosten eines Bauwerks sicher. Bereits in frühen Entwurfsphasen haben nebensächlich anmutende Entscheidungen erhebliche, kaum zu kalkulierende Konsequenzen für weitere Planungsprozesse. Die Planungsbeteiligten machen dabei Gebrauch von ihren Erfahrungen und nutzen Abschätzungen zur Entscheidungsfindung. Für ein Ingenieur-

büro stellt das Wissen und die Erfahrung der Mitarbeiter, besonders in Zeiten immer besser werdender Planungstools und Arbeitsmethoden (z.B. die Parametrische Modellierung), das größte Kapital dar.

Es ist daher wünschenswert mit Hilfe von Softwaresystemen, dieses Know-how einzelner Experten innerhalb einer Firma zu erfassen, zentral zu speichern und bei ähnlichen Fragestellungen anderen Ingenieuren zur Verfügung zu stellen. Hierzu wurden bereits seit den 1980er Jahren so genannte Wissensbasierte Systeme erforscht (Hayes-Roth und Jacobstein 1994), um Expertenwissen digital abzulegen und wiederzuverwenden. Wissensbasierte Systeme (englisch Knowledge-based Systems, KBS) sind Softwaresysteme, die mittels einer Wissensdatenbank sowie durch Inferenz, analog zu menschlichen Experten, komplexe Aufgaben lösen. Deren Anwendung im Bereich des Computer-Aided Designs (kurz CAD) nennt man Knowledge-based Engineering (kurz KBE).

Während die Anwendung von KBE (deutsch Wissensbasierte Konstruktion) sowohl in der Automobilbranche und der Luft- und Raumfahrttechnik, als auch im Maschinen- und Anlagenbau bereits Verwendung findet, ist in der von Klein- und Mittelständischen Unternehmen (KMU) geprägten Bauindustrie, nicht einmal die 3D-Konstruktion von Bauwerken üblich. Mit der Einführung von Building Information Modelling (kurz BIM) in Deutschland werden jedoch in den kommenden Jahren rapide Fortschritte zu verzeichnen sein.

Die Anwendung von KBE im Infrastrukturbau birgt angesichts der großen Anzahl ähnlicher Planungsaufgaben großes Potential. Durch die Wiederverwendung von Ingenieurwissen können repetitive Planungsprozesse der Brücken- und Tunnelplanung (teil-) automatisiert werden, um bei gleichzeitig steigender Planungsqualität Zeit zu gewinnen und Kosten einzusparen. So ist beispielsweise die Konstruktion eines Brückenwiderlagers ein immer wiederkehrender Planungsprozess im Brückenentwurf, bei dem die Funktionen und Anforderungen stets dieselben sind. Auch die Widerlagergeometrie ist in Normen und Richtlinien standardisiert. Die Varianz der Konstruktion entsteht lediglich im Kontext weiterer Bauteile oder des Geländes. Somit bietet sich die Automatisierung dieses Konstruktionsprozesses mit Hilfe eines KBE Systems an. Selbiges gilt analog für andere Brückenbauteile wie Überbau, Pfeiler oder Gründung.

1.2 Ziele der Arbeit

In dieser Arbeit soll zunächst der aktuelle Stand der Technik von Knowledge-based Engineering aufgezeigt werden. Neben einer Einführung in Wissensmodellierung und -repräsentation sowie Wissensbasierter Systeme sollen vor allem die methodischen Grundlagen der KBE Entwicklung und der aktuelle Stand der Forschung zu KBE erläutert werden.

Um den Nutzen bestehender KBE Systeme für Fragen des Infrastrukturbaus zu prüfen, sollen zwei verbreitete Softwaresysteme, Siemens NX Knowledge Fusion und Autodesk Inventor Automation Professional, untersucht werden. Insbesondere soll herausgefunden werden wie gut die verwendeten KBE Sprachen geeignet sind, um die Wissensabbildung zu unterstützen. Außerdem sind diese Softwaresysteme für die Anwendung im Infrastrukturbau zu bewerten.

Hauptziel dieser Arbeit ist die Entwicklung eines KBE Prototyps für frühe Phasen des Brückenentwurfs. Dazu ist es zunächst nötig, Regeln innerhalb des Brückenentwurfs in gängigen Regelwerken und durch Experten zu identifizieren und aufzubereiten. Anschließend erfolgt die Implementierung einiger Regeln mit Dynamo und Autodesk Revit unter Verwendung der Graph-basierten Programmierung.

Nicht zuletzt soll diese Arbeit als Basis für weitere Entwicklungen hinsichtlich der Anwendung von KBE für die Konstruktion von Infrastrukturbauwerken dienen. Dazu werden im Ausblick verschiedene Forschungs- und Entwicklungsansätze für weitere Implementierungen ausgearbeitet.

1.3 Aufbau der Arbeit

Dieser Abschnitt gibt einen Überblick über den Aufbau dieser Arbeit geben.

In Kapitel 2 soll zunächst ein Einstieg in die zentrale Thematik dieser Arbeit - nämlich Wissen - gelingen. Neben einer Definition wird besonders auf verschiedene Formen von Wissen, die Fragestellung der Wissensrepräsentation und -modellierung, sowie Wissen im Kontext Wissensbasierter Systeme näher eingegangen.

Darauf aufbauend folgt in Kapitel 3 eine Beschreibung Wissensbasierter Systeme, was wiederum als Grundlage für die darauffolgenden Kapitel dienen soll. Neben einer Beschreibung der Softwarearchitektur Wissensbasierter Systeme und ihrer Komponenten Wissensdatenbank, Inferenzmaschine und Erklärungskomponente, wird zuletzt auf Probleme Wissensbasierter Systeme eingegangen.

Kapitel 4 zeigt Methoden zur Entwicklung von KBE Systemen auf. Dazu zählen zum einen KADS/CommonKADS, MOKA, und KNOMAD. Die zugrundenliegenden Modelle zu Wissensrepräsentation, sowie die Rollen einzelner Beteiligter während des Implementierungsprozesses, werden näher dargelegt.

Es folgt Kapitel 5, welches sich ausschließlich dem Thema Knowledge-based Engineering widmet. Hier finden sich neben einem Überblick der Entwicklung von KBE, einem Überblick über die unterschiedlichen Forschungsaktivitäten im Bereich KBE, ein Abschnitt zu KBE Programmiersprachen, ein Überblick über Typen von KBE System, eine Diskussion der Vor- und Nachteile von KBE Systemen und schließlich eine Beschreibung kommerzieller Softwaresysteme.

Kapitel 6 beinhaltet eine Beschreibung der Entwicklung des KBE Prototyps. Neben einem Überblick des Brückenentwurfs und der zugrunde liegenden Regelwerke, wird die Implementierung des Prototyps erläutert. Dazu zählen eine Einführung in die verwendeten Softwaresysteme, die Modellierung der Familien in Revit und schließlich die Umsetzung einiger Entwurfsregeln in Dynamo / Autodesk Revit.

Am Ende der Arbeit folgt in Kapitel 7 eine Zusammenfassung der Ergebnisse sowie in Kapitel 8 ein Ausblick auf mögliche Forschungs- und Entwicklungsansätze aus dieser Arbeit.

2 Wissen

Wissen ist der zentrale Themenkomplex aller Teile dieser Arbeit. Es ist daher sinnvoll diesen näher zu betrachten. Dieses Kapitel gibt, neben einer Definition von Wissen, Auskunft über wichtige Fragstellungen wie Unterteilung in Wissenstypen oder Wissen im Kontext Wissensbasierter Systeme.

2.1 Definition

Die Definition des Begriffs Wissen hat sich im Laufe der Zeit immer wieder verändert. Eine eindeutige Begriffsbestimmung ist kaum möglich. Daher werden im Folgenden einige Definitionsansätze zusammengestellt.

Im Altdeutschen bedeutet das Wort „wissan“ (oder „wizzan“), „ich habe gesehen“. Dies deutet auf eine Nutzung des Begriffs im Zusammenhang mit gemachten Erfahrungen und gewonnen Erkenntnissen hin (Cenker 2012).

Wohl einer der ältesten Definitionen von Wissen entstammt dem Werk Theaitetos des Philosophen Platon. In diesem Werk beschreibt Platon Wissen als

„wahre und gerechtfertigte Meinung“

(englisch *justified true belief*).

(Duden Online 2013) beschreibt Wissen als

„durch eigene Erfahrung oder Mitteilung von außen Kenntnis von etwas, jemandem haben, sodass zuverlässige Aussagen gemacht werden können.“

Eine moderne Definition von Wissen gibt (Cenker 2012):

„Wissen bezeichnet die Gesamtheit der Kenntnisse und Fähigkeiten, die Individuen zur Lösung von Problemen einsetzen.“

2.2 Die Wissenspyramide

Die Begriffe Daten, Information und Wissen werden in den Informations- und Kommunikationswissenschaften hierarchisch gegliedert. Die Wissenspyramide (Abbildung 2.1) verdeutlicht die Beziehung der einzelnen Komponenten.

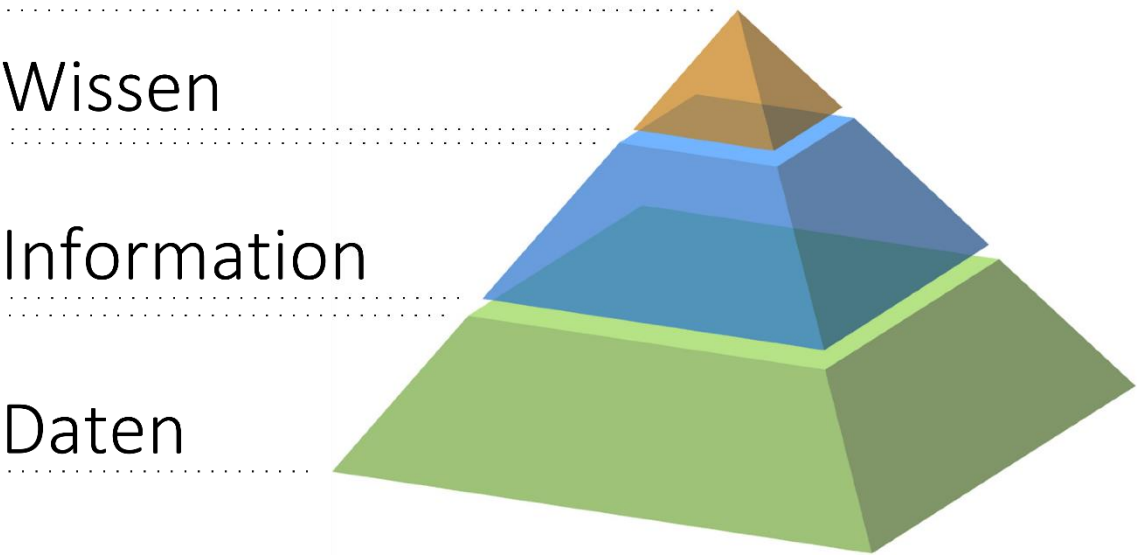


Abbildung 2.1: Wissenspyramide

Das Fundament der Wissenspyramide bilden die Daten. Daten (lateinisch: *datum* „gegeben“) sind zunächst einmal „dumme“ Ansammlungen von Zahlen und Zeichenketten die in keinerlei Relation zueinander stehen. Durch Beobachtung und Messungen können neue Daten gewonnen werden. Daten sind also das Rohmaterial für die weiteren höheren Ebenen. Durch Bewertung, Verarbeitung, Verknüpfung und Strukturierung der Daten können neue Informationen gewonnen werden. Wissen entsteht erst durch Vernetzung beziehungsweise Kombination von Informationen. Dieser Prozess der Wissensgenerierung ist nur mit menschlichen Verstand zu bewerkstelligen.

2.3 Wissenstypen

2.3.1 Explizites und implizites Wissen

Die Trennung von explizitem und implizitem Wissen ist in der Wissenstheorie von elementarer Bedeutung und betrifft viele andere Disziplinen ebenso. 1966 wurde diese Kategorisierung von Wissen von Michael Polanyi vorgestellt. Im Folgenden sind die wesentlichen Merkmale der beiden Wissenstypen aufgeführt.

Explizites Wissen

- ist in Sprache oder Schrift formulierbar
- kann mit anderen Menschen kommuniziert werden

Implizites Wissen

- ist nicht in Worte fassbar
- ist an eine Personen gebunden
- entsteht durch Erfahrungen

Die Abbildung von implizitem Wissen in Wissensbasierten Systemen stellt eine große Herausforderung dar. Da Experten ihre Erfahrungen kaum artikulieren können, ist es sehr schwer dieses Wissen in Softwaresystemen abzubilden (semantische Lücke).

2.3.2 Deklaratives und prozedurales Wissen

Die Unterscheidung von Wissen in deklaratives und prozedurales Wissen, nimmt bei der Entwicklung Wissensbasierter Systeme eine wichtige Rolle ein. Die deklarative Abbildung ist nämlich wesentliches Merkmal eines Wissensbasierten Systems und unterscheidet es von anderen Softwaresystemen. In (Liebig 2008) wird deklaratives und prozedurales Wissen folgendermaßen unterschieden:

Prozedural Wissen ist in einem Programm codiert. Die Bedeutung ergibt sich aus dem Verhalten des Programms. Das Wissen ist „versteckt" und damit nur schwer analysier- und kommunizierbar.

Deklarativ Abbildung der symbolischen Ausdrücke (Symbolebene) auf Abstraktionsebene der zu repräsentierenden Sachverhalte (Wissensebene). Wissen ist logisch analysierbar, kommunizierbar und verifizierbar.

2.4 Wissen im Kontext Wissensbasierter Systeme

Die wichtigste und gleichzeitig zeitintensivste Aufgabe während der Entwicklung eines Kowledge-based Engineering Systems, stellt der Aufbau einer Wissensdatenbank, durch Analyse, Wissenserfassung und -repräsentation des Problems dar (Rasdorf 1985). In vielen Anwendungsbereichen treten durch den Prozess der Wissensakquirierung Lücken zwischen den verfügbaren Wissensquellen und des tatsächlich innerhalb der Domäne benutzten Wissens zu Tage. Diese Schwierigkeiten sind typisch für die Entwicklung Wissensbasierter Systeme im Ingenieurwesen, wo Wissen nicht starr festgelegt ist und die Informationsflüsse fragmentiert sind.

Trotz allem lässt sich die Entwicklung einer Wissensdatenbank stark vereinfachen, indem man das Wissen entsprechend ihrer Funktion klassifiziert. Die im Bereich des Ingenieurwesens im Wesentlichen nach (Rasdorf 1985) auftretenden Typen sind:

- Fakten
- Prozeduren
- Beurteilungswissen
- Kontrollwissen

Im Folgenden werden diese von (Rasdorf 1985) identifizierten Wissenstypen näher beschrieben:

2.4.1 Fakten

Faktenwissen ist der einfachste, beschreibende Wissenstyp. Dieser Typ besteht aus formalisiertem Wissen, dass typischerweise in Handbüchern zu finden ist. Faktenwissen ist beispielsweise Wissen über bestimmte Materialeigenschaften, physikalische Gesetze, technische Datenblätter, Prüf-, Überwachungs- und Zertifizierungsdokumente. Faktenwissen kann sehr einfach mit Hilfe von Datenbanken digital repräsentiert werden.

2.4.2 Prozeduren

Prozedurales Wissen besitzt algorithmische und operative Eigenschaften. Algorithmen enthalten zur Lösung eines Problems oder zur Herbeiführung eines Zustands numerische oder nicht-numerische Schritt-für-Schritt Anweisungen. Im Allgemeinen überführen Al-

gorithmen Fakten von einem Ausgangs- in einen Zielzustand. Prozedurales Wissen bearbeitet also Faktenwissen, indem es Fakten bildet, löscht, modifiziert und transportiert. Beispiele für Prozedurales Wissen sind unter anderem die Finite-Element-Methode (FEM), Database Management Systeme (DBMS), Optimierungen und graphische Benutzeroberflächen.

2.4.3 Beurteilungswissen

Beurteilungswissen kann nur sehr schwer in herkömmlichen Softwaresystemen implementiert werden. Diese Art von Wissen besteht aus Komponenten, die individuelle Überzeugungen und Meinungen verkörpern und Heuristiken, die Beobachtungen, Beurteilungen, Erfahrungen, Best-Practice, Plausibilität und Wissen über anerkannte Prozeduren und Methoden beinhalten.

Obwohl Ingenieure diesen Wissenstyp traditionell nicht in Programmen implementieren, ist diese Art von Wissen dennoch in allen Phasen des Planungsprozesses essentiell. Ingenieure nutzen Abschätzungen und ihre Erfahrung, um Entscheidungen abzuwägen und die Lösungssuche zu beschleunigen. Häufig sind diese Erfahrungen in Regularien wie Verordnungen, Spezifikationen, Normen, Richtlinien, Empfehlungen und weiteren regulatorischen Dokumenten eingebettet.

2.4.4 Kontrollwissen

Kontrollwissen ist Metawissen, also Wissen über Wissen, was die Verarbeitung der bereits erwähnten Wissenstypen in einem wissensbasierten System steuert. Kontrollwissen kann prozeduraler, heuristischer oder beider Natur sein. Wichtige Merkmale von Kontrollwissen sind:

- Musterbestimmte Handlung
- Flexibles und teilweise selbstmodifizierendes Verhalten
- Voraussehen von unerwarteten Entwicklungen
- Nutzung eines ergebnisoffenen Lösungsraums
- Behandlung von Unsicherheiten

Kontrollwissen kontrolliert und koordiniert die Entstehung der anderen Wissenstypen, indem es geeignete Wissensquellen für ein bestimmtes Problem auswählt.

2.4.5 Klassifizierung von Wissen nach Chandrasekaran

Das im Ingenieurwesen genutzte Wissen ist nicht einheitlich in Art und Anwendbarkeit. Während das Wissen an Stellen, an denen allgemeine Entscheidungen getroffen werden, semantisch sehr hochwertig ausgeprägt ist, kann es, einzelne Komponenten oder einfache Aktivitäten betreffend, sehr einfach sein oder aber verschiedenste Komplexitätsgrade zwischen beiden Extremen annehmen.

In ähnlicher Weise kategorisiert (Chandrasekaran 1983) Planungsaufgaben. Er identifiziert drei Klassen von Planungsaufgaben und ihr zugrunde liegendes Wissen, nämlich Kreatives Wissen, Innovatives Wissen und Routine Wissen:

Kreatives Wissen Wird vom Ingenieur verwendet um unterschiedlichste, oft sogar widersprüchliche Möglichkeiten abzuwägen. Die Nutzung dieses Typs erfolgt speziell in frühen Planungsphasen, wo Konflikte zwischen konträren Projektzielen gelöst werden und mit verfügbaren Ressourcen sparsam umgegangen werden muss. Oder aber innerhalb kritischer Planungsphasen in denen unvorhergesehene Probleme auftreten und Lösungsalternativen ausgearbeitet werden. Ingenieure nutzen kreatives Wissen um beispielsweise Planungsvarianten zu bewerten. Ein Problem aus dem Brückenentwurf wäre zum Beispiel die Bewertung verschiedener Tragsysteme unter denselben äußeren Entwurfsbedingungen.

Innovatives Wissen Ist im Vergleich zu kreativem Wissen weniger allgemein definiert. Es trägt aber immer noch wesentlich zur Entscheidungsfindung bei und wird mehr oder weniger selbstverständlich während der Planung angewandt. Ingenieure nutzen Innovatives Wissen um die richtige Software für eine Planungsaufgabe auszuwählen, Projekte zu managen und zu terminieren und ein Bauwerk entsprechend der gegebenen Lasten, Materialien und anderen Randbedingungen zu planen.

Routine Wissen Ist zur Erbringung der täglich anfallenden Aufgaben erforderlich. Die Anwendung dieses Wissenstyps zur Unterstützung der Entscheidungsfindung erfolgt bereits. Alternativen können aus einer großen aber klar dezidierten Wissensdatenbank gewählt werden. Ingenieure nutzen Routine Wissen für die Analyse technischer Störungen, die Prozessoptimierung, das Controlling und die Dokumentation. Beispielsweise wird im Bauingenieurwesen zur Planableitung oder Dokumentation der statischen Berechnungen Routine Wissen verwendet.

Diese Klassifizierung des für Ingenieursaufgaben nötigen Wissens kann nicht starr festgelegt werden. Die Wissenstypen können sich durchaus überschneiden oder weiter unterteilt werden. Dennoch ist diese Klassifizierung nach (Chandrasekaran 1983) ausreichend, um die Bandbreite an Ingenieurwissen adäquat abzubilden (Rasdorf 1985).

3 Wissensbasierte Systeme

Dieses Kapitel bietet eine Einführung zu Wissensbasierten Systemen. Nach Beschreibung der Charakteristik und Architektur eines Wissensbasierten Systems, werden die Themengebiete Wissensmodellierung und -repräsentation behandelt. Zum Schluss folgt eine Diskussion der Probleme Wissensbasierter Systeme.

Ein Wissensbasiertes System nutzt menschliches Wissen um Probleme zu lösen, deren Lösung normalerweise menschliche Intelligenz erfordert (Tripathi 2011). Wissensbasierte Systeme werden oft auch synonym als Expertensysteme oder Regelbasierte Systeme bezeichnet. Expertensysteme fokussieren dabei mehr auf die Ersetzung eines menschlichen Experten und die Lösung eines Problems durch ein Softwaresystem, während Wissensbasierte Systeme sich mehr auf die Softwarearchitektur beziehen. Ausgehend von ersten Erfolgen in den 1980er Jahren, werden heutzutage Wissensbasierte Systeme in unterschiedlichsten Bereichen und Wissensgebieten erfolgreich eingesetzt.

3.1 Charakteristik Wissensbasierter Systeme

(Tripathi 2011) beschreibt die Charakteristik Wissensbasierter Systeme folgendermaßen:

Grundlage jedes Wissensbasierten Systems ist die Ressource Wissen. Die Mächtigkeit eines Wissensbasierten Systems resultiert aus dem speziellen, qualitativ hochwertigen Wissen der Domäne. Im Gegensatz zu konventionellen Programmen, wird in Wissensbasierten Systemen das Wissen von der Verarbeitung desselbigen getrennt. So sind Wissensdatenbank und Inferenzmaschine zwei separierte Komponenten.

Wissensbasierte Systeme

1. lösen schwierige Probleme einer Domäne so gut oder besser wie die menschlichen Experten
2. können enorme Mengen an domänenspezifischen Wissen besitzen
3. verwenden Heuristiken, um den Inferenzmechanismus zu leiten und dadurch den Suchraum einzugrenzen
4. verwenden Symbole, um unterschiedliche Wissenstypen abzubilden
5. können mit unsicheren und irrelevanten Daten umgehen

3.2 Architektur Wissensbasierter Systeme

Ein Wissensbasiertes System besteht wie in Abbildung 3.1 gezeigt, im Wesentlichen aus den Komponenten Benutzerschnittstelle, Wissensdatenbank, Inferenzmaschine, Wissensakquise und Erklärungskomponente.

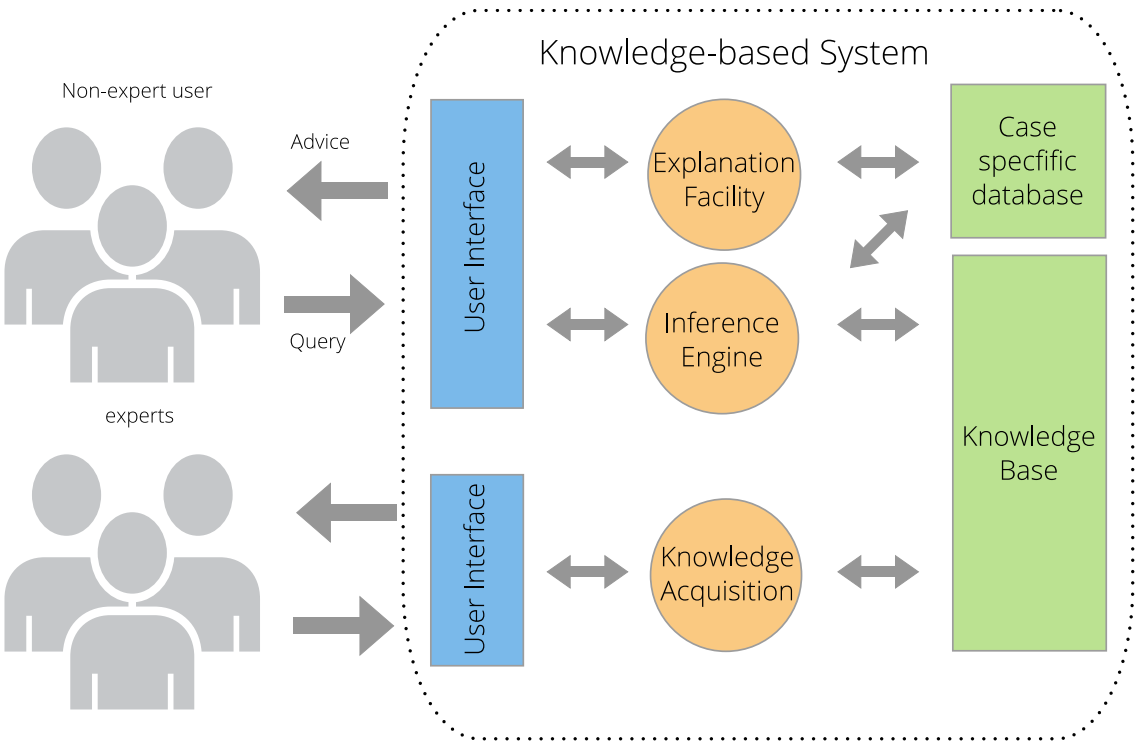


Abbildung 3.1: Aufbau eines wissensbasierten Systems

In (Tripathi 2011) ist folgende Beschreibung der Komponenten zu finden:

Wissensdatenbank Die Wissensdatenbank sammelt und speichert das Wissen beziehungsweise die Regeln einer Domäne, die durch die Experten mit Hilfe der Wissensakquirierungskomponente bereitgestellt wurden, um die Probleme zu lösen. In dieser Datenbank werden alle Wissensformen (siehe 2.4) in expliziter deklarativer Form gespeichert.

Inferenzmaschine Eine Inferenzmaschine ist das Gehirn eines Wissensbasierten Systems. Sie interpretiert die Regeln und stellt Methoden zum

Schließen zur Verfügung. Die in der Wissensdatenbank gespeicherten Regeln werden durch die Inferenzmaschine analysiert und abgearbeitet, um ausgehend von der Abfrage des Anwenders nach Schlussfolgerung eine Entscheidung zu treffen. Dafür werden zwei Herangehensweisen, nämlich Vorwärts- und Rückwärtsschließen verwendet.

Wissensakquise

Wissensakquise bezeichnet die Sammlung, Übertragung und Umwandlung der Problem-lösenden Expertise von einem Experten oder anderen Quellen in ein Softwaresystem, um eine Wissensdatenbank aufzubauen oder zu erweitern.

Erklärungskomponente

Das Merkmal *„Black-Box“* ist häufiger Kritikpunkt bei der Anwendung Wissensbasierter Systeme. Anwender äußern Bedenken hinsichtlich eines blinden Vertrauens sowie ein Vertrauen in eine „falsche Sicherheit“ des Systems. Sie fürchten einen Kontrollverlust durch die Anwendung Wissensbasierter Systeme. Mit Hilfe einer Erklärungskomponente kann dem Benutzer erklärt werden, WIE das System zu einer Lösung kam. Durch die explizite Wissensrepräsentation ist das System in der Lage eine Schlussfolgerung zu begründen und dem Anwender über die Erklärungskomponente mitzuteilen.

Benutzerschnittstellen

Innerhalb eines Wissensbasierten Systems existieren zwei getrennte Benutzerschnittstellen: Eine für die Anwender, die andere für die Experten. Ein Anwender kann ausgehend von einer konkreten Problemstellung über eine Benutzerschnittstelle die Wissensdatenbank abfragen und erhält vom System eine entsprechende Handlungsanweisung. Ein Experte wiederrum definiert, modifiziert oder löscht Regeln mit Hilfe dieser Schnittstelle. Damit kann der Experte also Regeln bearbeiten, ohne dabei selbst über spezielle Programmierkenntnisse zu verfügen. Die Gestalt der Schnittstellen ist für den Aufbau

der Wissensbasierten Systeme nicht relevant. Sie können beispielsweise als graphische Benutzeroberflächen (Graphical User Interface, kurz GUI), Kommandozeilen oder sprachbasiert gestaltet sein. Grundsätzlich ist jedoch, dass sie abhängig von der jeweiligen Anwendung so einfach wie möglich gehalten sind. In Hinblick auf KBE Systeme können diese Schnittstellen auch in eine CAD Anwendung integriert werden.

3.3 Wissensmodellierung

Die Wissensmodellierung (englisch Knowledge Engineering) ist eine Methodik zur Entwicklung Wissensbasierter Systeme. Wie andere Methoden in anderen Bereichen der Softwareentwicklung, stellt sie geeignete Werkzeuge zur methodischen Entwicklung Wissensbasierter Systeme bereit. Die Wissensmodellierung ist ein Spezialgebiet der Künstlichen Intelligenz und des Wissensmanagements (englisch Knowledge Management).

Die vier erstrangigen Aufgaben zur Modellierung von Wissen innerhalb Wissensbasierter Systeme sind, Wissen

a) zu erfassen,
b) zu formalisieren,
c) zu verarbeiten,
d) zu visualisieren.

Weitere Aspekte der Wissensmodellierung werden unter anderem in Kapitel 4 vorgestellt.

3.4 Wissensrepräsentation

Zur Formalisierung und Repräsentation von Wissen in Wissensbasierten Systemen, existieren verschiedene Ansätze. Grund dieser Diversität ist die Entwicklung der Repräsentationsformen in unterschiedlichen Wissensgebieten. So entstammt die Logik der Mathematik, während Neuronale Netze in der Biologie entwickelt wurden. (Liese 2003) gibt einen ausführlichen Überblick über die verschiedenen Formen der Wissensrepräsentation:

Logik

Die Logik ist eine bedeutende und wichtige Form der Wissensrepräsentation, schließlich bauen alle anderen Formen auf dieser auf. Mathematische Begriffe und die formale Logik werden in der logischen Wissensrepräsentation verwendet, um Wissen abzubilden. Weit verbreitet ist die Anwendung der *Prädikatenlogik* als logisches Repräsentationsschema. Die Syntax der Logik legt Objekte, wie *Variablensymbole*, *Funktionssymbole*, *Konstantensymbole* und *Prädikatsymbole*, fest und definiert eine Sprache in der Aussagen formuliert werden können. Aus diesen primitiven Symbolen werden größere Objekte aufgebaut. Außerdem ist es mit Hilfe von *Allquantors*, *Existenzquantors* und der *Implikation* möglich Regeln abzubilden.

Constraint-basierter Ansatz

Um einen durch Parameter definierten Lösungsraum einzugrenzen, können im Constraint-basieren Ansatz *Constraints* verwendet werden. Der Constraint-basierte Ansatz wird üblicherweise in parametrischen CAD-Systemen eingesetzt. Die Anwendung von *Constraints* bietet gegenüber Regeln den Vorteil, dass sie Abhängigkeiten unter geometrischen Objekten rein deklarativ und ungerichtet beschreiben. Durch einen *Constraint Solver* kann das zugrundeliegende Gleichungssystem nach jeder Variable gelöst werden. Es stellt damit einen mächtigen Ansatz zur Beschreibung konstruktiver Abhängigkeiten und Regeln dar.

Graphen

Graphen dienen als Basisrepräsentation und können beispielsweise zur Darstellung von Strukturen oder als Basis für *Semantische Netze* dienen. Ein Graph besteht aus einer nichtleeren endlichen Menge von Knoten und einer Menge ungeordneter Knotenpaare,

die Kanten genannt werden. In CAD-Systemen werden Graphen unter anderem zur Abbildung der Produktstruktur verwendet.

Semantische Netze

Ein *Semantisches Netz* besteht aus Knoten, welche Begriffe symbolisieren, und aus Kanten, welche die Beziehungen der Begriffe repräsentieren. Ein *Semantisches Netz* ist ein gerichteter Graph mit bewerteten Knoten und Kanten. Dadurch sind *Semantische Netze* sehr gut geeignet Abhängigkeiten zwischen den Objekten zu formalisieren. Die Freiheit der Kantentypen erlaubt die Abbildung komplexer Zusammenhänge.

Produktionssysteme und regelbasierter Ansatz

Eine *Produktionsregel* ist eine mit einer Vorbedingung versehene Aktion. Die Aktion gilt als ausführbar, wenn die Vorbedingung erfüllt ist. In dieser Form wird Wissen in der prozeduralen Wissensrepräsentation als Sammlung von Anweisungen repräsentiert. *Produktionssysteme* bestehen aus einer Faktenbasis, den Produktionsregeln und einem Interpretationsprozess. Um Fakten von einem Ausgangs- in einen Zielzustand zu überführen wird auf den Regeln ein Inferenzmechanismus zum Beispiel die Vorwärtsverkettung – falls nötig auch iterativ - ausgeführt. In KBE Systemen ist diese regelbasierte Wissensrepräsentation am meisten verbreitet.

Vorteile der regelbasierten Wissensrepräsentation sind:

- Experten formulieren Wissen gerne in Regeln, dadurch einfach formalisierbar
- Abbildung von Regelwissen wird in kommerziellen KBE Systemen unterstützt

Nachteile der regelbasierten Wissensrepräsentation sind:

- Komplexe Probleme nicht abbildbar
- Regeln sind im Gegensatz zu *Constraints* gerichtet (aus A folgt immer B)
- Regelbasis tendiert zur Unübersichtlichkeit
- Zur Ausführung der Regeln nötige Anwendungslogik liegt außerhalb des Systems

Entscheidungsbäume

Entscheidungsbäume dienen der Visualisierung von *Produktionsregeln*. Die graphische Darstellung zeigt den hierarchischen Aufbau der Regeln.

4 Methoden zur Entwicklung Wissensbasierter Systeme

Die Entwicklung eines KBE Systems folgt häufig einem immer gleichen Muster. Zunächst muss das Problem auf konzeptioneller Ebene erfasst und verstanden werden. Anschließend wird das Problem in kleinere Instanzen unterteilt, welche eindeutig verstanden werden. Daran schließt sich ein iterativer Prozess an, der so lange anhält, bis das Ergebnis als zufriedenstellend bewertet werden kann. Dieses Muster wurde bereits in verschiedenen Methoden verständlich beschrieben. Dazu gehören CommonKADS (Schreiber et al. 2000) und MOKA (Stokes 2001). Allerdings beschäftigen sich die meisten existierenden Methoden mit der Entwicklung von Wissensbasierten Systemen, nicht mit der Entwicklung von KBE Systemen. Zudem sind die Methoden speziell für die Anwendung in Firmen der großen Industriebranchen wie Maschinenbau, Automobilbau oder Luft- und Raumfahrttechnik gedacht (Lovett et al. 2000). MOKA jedoch zielt besonders auf die Unterstützung der Entwicklung von KBE Systemen ab (Sandberg 2003).

4.1 KADS/CommonKADS

CommonKADS ist eine weit verbreitet Methode zur Entwicklung von Wissensbasierten Systemen. Der Begriff KADS ist ein Akronym und steht für **K**nowledge **A**cquisition **D**ocumentation and **S**tructuring. CommonKADS ist das Ergebnis einer Reihe von internationalen Forschungsaktivitäten und Anwendungsprojekten, welche bis auf das Jahr 1983 zurückgehen. Es wurde von Universitäten und Firmen im Rahmen des European ESPRIT Programms entwickelt. Heutzutage ist es der europäische de-facto Standard für die Entwicklung von Wissensbasierten Systemen und findet in vielen Firmen in Europa, USA und Japan Anwendung. Außerdem bietet CommonKADS Methoden für detaillierte Wissenserfassung und -verarbeitung. Abbildung 4.1 zeigt die Entwicklung von Methoden für Wissensbasierte Systeme seit 1965.

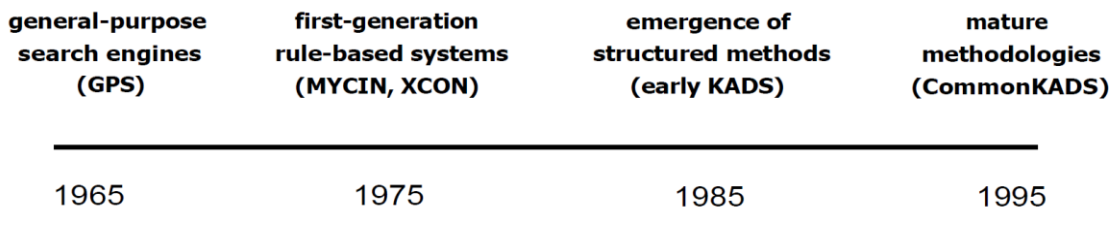


Abbildung 4.1: Die Entwicklung von Methoden für wissensbasierte Systeme seit 1965 (Schreiber et al. 2000)

Historisch wurden Wissensbasierte Systeme durch simples Ausprobieren entwickelt, den methodischen Aspekten wurde dabei wenig Beachtung geschenkt. Schließlich formulierten Anwender in der Industrie die Notwendigkeit von Richtlinien und Techniken, um den Entwicklungsprozess zu strukturieren und besser zu kontrollieren. Folglich begrüßten Entwickler die ersten Schritte von CommonKADS, um diese Lücke zu schließen.

Ausgehend vom KADS-I-Projekt (Schreiber et al. 1993), begann im Winter 1990 die Entwicklung einer neuen Methode, welche kommerziell nutzbar sein und den gesamten Lebenszyklus Wissensbasierter Systeme abdecken sollte, genannt CommonKADS (KADS-II). Über die Jahre wurden die Methoden, basierend auf Feedback der Anwender und Wissenschaftler, immer weiter verbessert. Die praktische Anwendung von CommonKADS zeigte, dass viele Projekte aufgrund einer Technologie getriebenen Herangehensweise scheiterten. Eine Firma kann Informations- und Wissenstechnologien nur dann erfolgreich implementieren, wenn die Rolle des Systems und der potenzielle Effekt innerhalb der Firma explizit vorher, sowie während des Entwicklungsprozesses definiert werden (Schreiber et al. 2000). Aus diesem Grund stellt die Einführung von Wissensorientierten Methoden und Techniken für den Zweck der Analyse einen wesentlichen Vorteil dar. KADS beinhaltet eine Vielzahl von Funktionen und ist daher sehr schwierig zu erlernen und nicht für die Anwendung in Klein- und Mittelständischen Unternehmen (KMU) geeignet (Lovett et al. 2000). Im Folgenden soll das Konzept und die Methoden hinter CommonKADS näher beschrieben werden.

4.1.1 CommonKADS Model Suite

Die *Model Suite* stellt den Kern der CommonKADS Methodik dar und beschreibt die methodische Umsetzung der Entwicklung eines Wissensbasierten Systems. Wie in Abbildung 4.2 erkennbar, besteht die *Model Suite* aus drei Gruppen von Modellen: *Context*, *Concept* und *Artefact*. Jede Gruppe gibt dabei nach (Schreiber et al. 2000) jeweils Antwort auf eine wesentliche Frage:

Warum? Warum ist das Wissensbasierte System von potentiellem Nutzen? Für welche Probleme eignet sich der Einsatz? Welche Auswirkungen hat das System hinsichtlich Kosten? Welche Vorteile bieten sich dem Unternehmen? Das Verständnis über die Projektumgebung steht hier im Vordergrund (*Context*).

Was? Was sind die Eigenschaften und Strukturen des vorhandenen Wissens? Was sind die Eigenschaften und Strukturen der zu diesem Wissen korrespondierenden Kommunikation? Die konzeptuelle Beschreibung des für die Aufgabe verwendeten Wissens steht im Vordergrund (*Concept*).

Wie? Wie muss das Wissen in einem Softwaresystem implementiert sein? Wie sieht die Softwarearchitektur aus? Die technischen Aspekte der Realisierung des Wissensbasierten Systems stehen hier im Vordergrund (*Artefact*).

Diese Fragen werden durch die Entwicklung von Modellen beantwortet. CommonKADS stellt dafür eine Reihe vorgefertigter Modelle (Abbildung 4.2) zur Verfügung. Jedes Modell betrachtet dabei nur einen Teilausschnitt, zusammen jedoch ergibt sich ein umfassender Überblick. In (Weih et al. 1994) und (Schreiber et al. 2000) sind diese Modelle genauer erläutert:

Organization Model Beschreibt die Organisation/Domäne für das zu entwickelnde System. Es bildet die Funktionen, Strukturen, Prozesse, Probleme, Autoritäten und Ressourcen der Domäne ab.

Task Model Beschreibt die Aufgaben in einer Domäne, welche zukünftig durch ein Wissensbasiertes System übernommen werden sollen. Das Task Model wird durch eine strukturierte Liste von Aufgaben repräsentiert. Eine Aufgabe wird durch Eingang, Ausgang, Ziel, Steuerung, Eigenschaften, verknüpfte Bestandteile, Beschränkungen und die geforderten Eigenschaften abgebildet.

Agent Model Beschreibt alle relevanten Eigenschaften von Akteuren, die in der Lage sind, die Aufgaben des *Task Model* zu lösen. Ein Akteur kann das Wissensbasierte System, ein Benutzer, ein Softwaresystem, eine Datenbank oder diverse andere Instanzen sein. Ein Akteur wird durch seine generellen Fähigkeiten, seine Einschränkungen und sein Urteilsvermögen beschrieben.

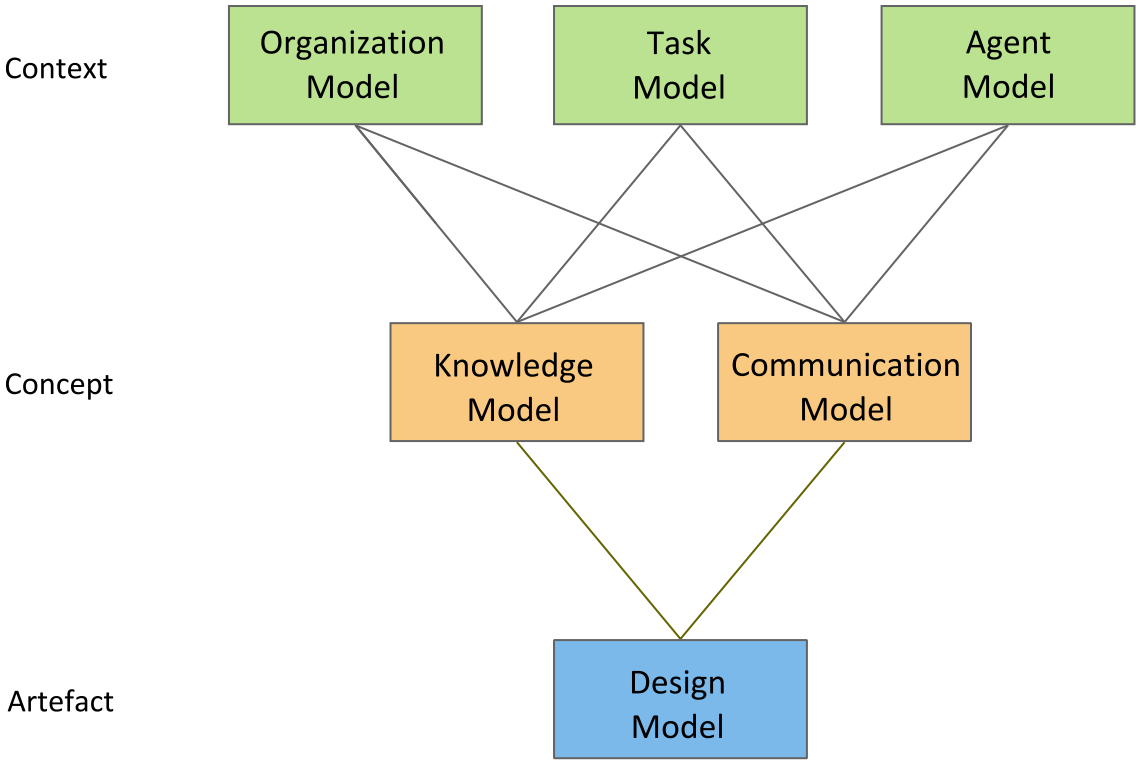


Abbildung 4.2: CommonKADS Model Suite (Schreiber et al. 2000)

Communication Model Beschreibt alle Vorgänge zwischen den einzelnen Akteuren. Es bildet jede Kommunikation zwischen Akteuren hinsichtlich Arbeitsvorgängen, Vorgangsplanung, Initiativen, sowie die Fähigkeit zur Teilnahme an einem Vorgang ab.

Knowledge Model Beschreibt das Potential eines Akteurs hinsichtlich wissensintensiver Problemlösung. Das *Knowledge Model* wird weiter in *Domain Knowledge*, *Inference Knowledge* und *Task Knowledge* unterteilt. Im Gegensatz zu KADS, ist diese Beschreibung ist sehr viel detaillierter.

Design Model Ausgehend von den vorangegangen Modellen beschreibt dieses Modell die technischen Spezifikationen. Diese sind unter anderem die Softwarearchitektur, Implementierungsplattform, Software Module, um die im *Communication Model* und *Knowledge Model* beschriebenen Funktionalitäten umzusetzen.

4.1.2 Das Knowledge Model

In (Speel et al. 2001) werden die Aufgaben und die Strukturen des *Knowledge Model* in KADS genauer erläutert:

Zusammen mit dem *Communication Model* bildet das *Knowledge Model* die *Concept* Ebene innerhalb der *Model Suite* von CommonKADS (siehe Abbildung 4.2). Es beschäftigt sich also mit der konzeptionellen Wissensmodellierung innerhalb des Systems. Das *Knowledge Model* spezifiziert dabei die Anforderungen des zu entwickelnden Systems hinsichtlich Wissensabbildung und Inferenzmechanismen, während das *Communication Model* die Bedürfnisse bezüglich der Schnittstellen mit anderen Agenten (Benutzer oder andere Softwaresystemen) berücksichtigt.

Zusammen dienen beide Modelle als Grundlage für das Design und die Implementierung des Systems (*Artefact*). Eine identifizierte Aufgabe, die von dem Wissensbasierten System übernommen werden soll, ist dabei Ausgangspunkt der konzeptionellen Modellierung. Die Annahme geht von einer wissensintensiven Aufgabe aus, deren Abbildung in technischer und wirtschaftlicher Hinsicht realisierbar ist. Die sogenannte konzeptionelle Modellierung ist eine Methode, die dazu beiträgt, die Struktur wissensintensiver Aufgaben besser zu verstehen und zu verdeutlichen. Das *Knowledge Model* stellt also eine für die Implementierung notwendige Beschreibung von Daten und Wissen zur Verfügung. Das Modell wird innerhalb des Prozesses der Wissensanalyse erstellt.

Das *Knowledge Model* besteht aus drei *Knowledge Categories*: *Domain Knowledge*, *Task Knowledge* und *Inference Knowledge*. Sie werden in (Schreiber et al. 2000) erläutert:

Domain Knowledge — Spezifiziert das domänenspezifische Wissen und die Informationstypen, welche in einer Anwendung genutzt werden sollen. Eine Beschreibung des Domänenwissens ist vergleichbar mit einem Daten- oder Klassenmodel in der Softwareentwicklung.

Task Knowledge — *Task Knowledge* beschreibt die Ziele einer Anwendung und die Methoden, um diese Ziele durch eine Aufteilung in Subaufgaben und Inferenzen zu erreichen.

Inference Knowledge *Inference Knowledge* bildet die grundlegenden Inferenz-Schritte ab, die auf dem *Domain Knowledge* ausgeführt werden.

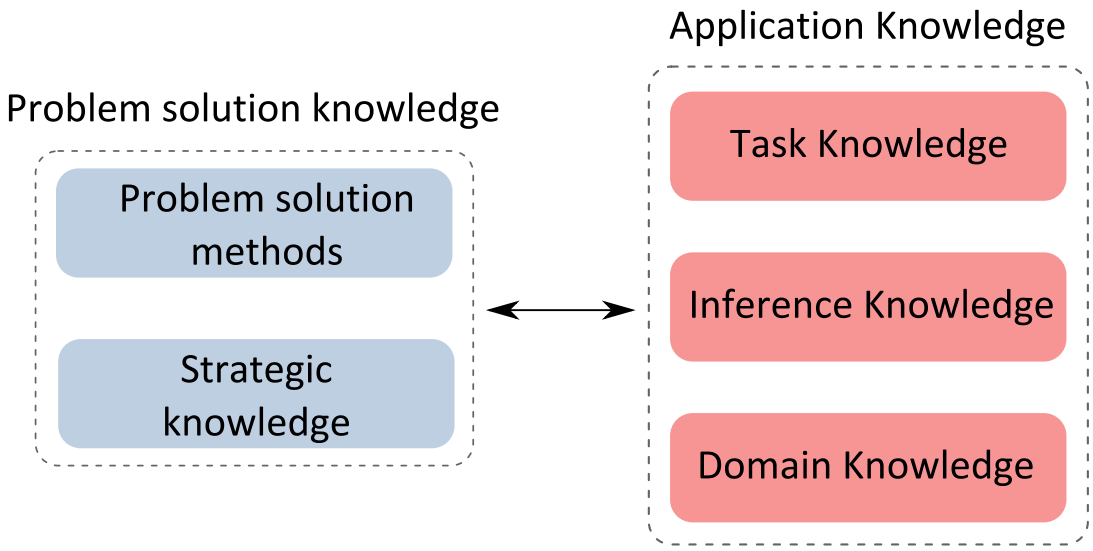


Abbildung 4.3: Kategorien des Knowledge Model in CommonKADS (Schreiber et al. 2000)

In Abbildung 4.3 sind die drei *Knowledge Categories* des *Knowledge Model* dargestellt. Das *Knowledge Model* enthält keine die Implementierung betreffenden Informationen. Es ist essentiell, während der konzeptionellen Modellierung des Systems implementierungsspezifische Überlegungen bewusst auszuklammern. Dies befreit den Wissensanalyst von etwaigen Umsetzungsbedenken seines erfassten Wissens.

4.1.3 Rollen

Innerhalb der Prozesse Wissenserfassung und -abbildung können verschiedene Akteure mit unterschiedlichen Rollen identifiziert werden. KADS unterscheidet sechs verschiedene Rollen, die in Abbildung 4.4 graphisch aufbereitet wurden. Die Übersicht zeigt das Zusammenwirken der verschiedenen Rollen während der Entwicklung eines Wissensbasierten Systems. In (Schreiber et al. 2000) sind die Funktionen der einzelnen Akteure folgendermaßen erläutert:

Knowledge Provider / Specialist

Er stellt das menschliche Wissen zur Verfügung. Er ist typischerweise ein Experte der Domäne, kann aber genauso eine andere Person sein die selbst kein Expertenstatus hat.

Knowledge Engineer / Analyst

Eine wesentliche Herausforderung des *Knowledge Engineers* ist es, die wahren Experten einer Domäne aufzudecken. Der Begriff "Knowledge Engineer" ist grundsätzlich für alle Rollen innerhalb der Entwicklung Wissensbasierter Systeme verwendbar, während der *Knowledge Analyst* ausschließlich für die Analyse des Systems zuständig ist. CommonKADS bietet dem *Knowledge Analyst* eine Reihe von Methoden und Werkzeuge, die die Analyse einer wissensintensiven Aufgabe relativ einfach machen.

Knowledge System Developer

Sie ist verantwortlich für die Softwarearchitektur und Implementierung des Systems. Der Entwickler muss die grundlegenden Analysemethoden beherrschen, um die Auflagen und Modelle des *Knowledge Analysts* zu verstehen.

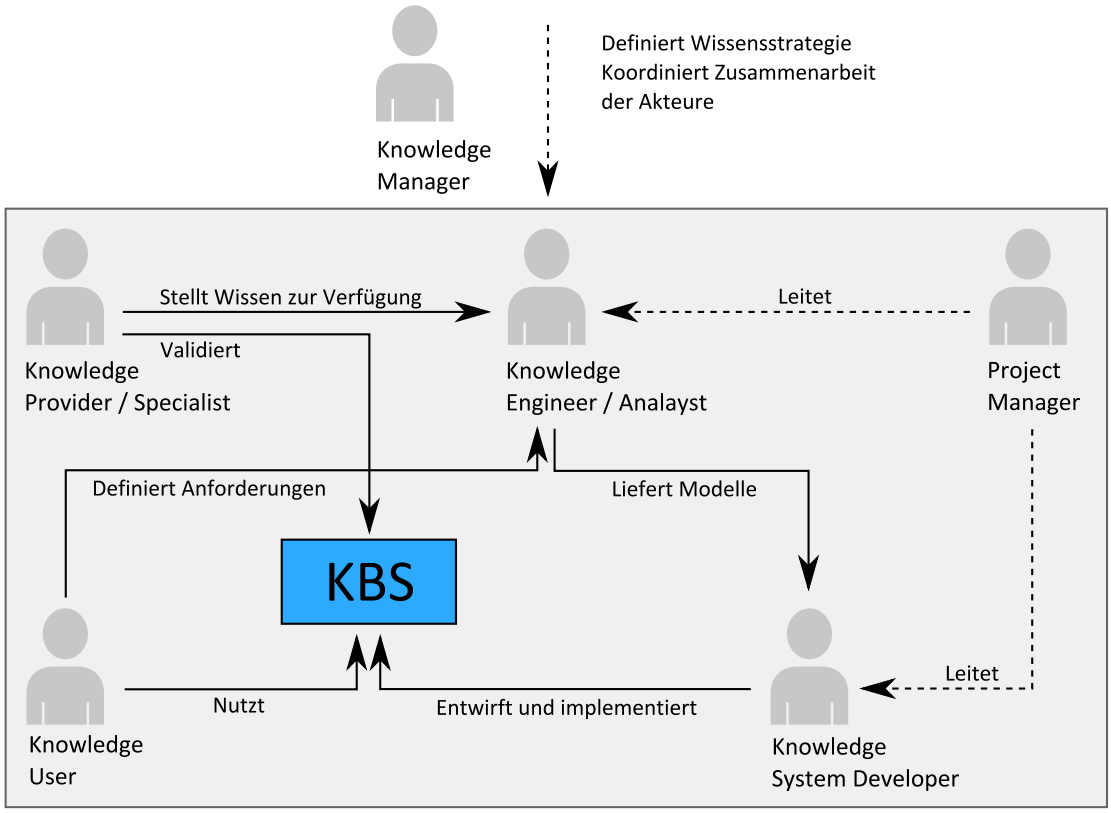


Abbildung 4.4: Darstellung der sechs Rollen in Knowledge Engineering und Management (Schreiber et al. 2000)

Knowledge User

Er nutzt direkt oder indirekt ein Wissensbasiertes System. Seine Anforderungen an das Wissensbasierte System sind entscheidend für die Projektentwicklung und -validierung.

Project Manager

Er leitet das Projekt. Er führt die *Knowledge Engineers*, *Knowledge Analysts* und die *Knowledge System Developers*.

Knowledge Manager

Der *Knowledge Manager* agiert auf höchster Ebene innerhalb des Projektteams. Er definiert beispielsweise die Wissensstrategie oder koordiniert die Zusammenarbeit aller anderen Rollen, ist aber nicht aktiv an der Entwicklung des Systems beteiligt.

4.2 MOKA

MOKA (**M**ethodology and tools **O**riented to **K**nowledge-based Engineering **A**pplications) (Stokes 2001) war ursprünglich ein von der Europäische Kommission gefördertes Projekt. Die Methode wurde innerhalb des AIT (**A**dvanced **I**nformation **T**echnology in Design and Manufacturing) Programms mit dem Ziel entwickelt, einen international gültigen, neutralen Standard für die methodische Implementierung von KBE Systemen zu bieten. Das MOKA Projekt begann im Januar 1998 und dauerte insgesamt 30 Monate. Projektbeteiligte waren unter anderem Aerospatiale-Matra, BAE Systems, Daimler-Chrysler, PSA Peugeot Citröen und Knowledge Technologies International (KTI).

Die Entwicklung von MOKA unterlag stark den Anforderungen der europäischen Automobil- und Raumfahrtbranche. Die grundlegenden Ziele eines solchen Standards waren:

- ein konsistenter Weg zur Erfassung und Repräsentation von Produkt- und Designprozesswissen
- ein Softwaretool zu Erfassung, Repräsentation und Erhaltung dieses Wissens
- langfristig die Möglichkeit der automatisierten Generierung von KBE Code mit Hilfe dieses Softwaretools

MOKA stellt dazu sowohl zur Wissenserfassung als auch zur Wissensrepräsentation ein Framework zur Verfügung. Dieses Framework arbeitet auf zwei Ebenen: *Informal Model* und *Formal Model*. Das *Informal Model* ist sehr einfach aufgebaut, um auch von Projektbeteiligten ohne technischen Hintergrund verstanden werden zu können. Das *Formal Model* hingegen ist wesentlich formaler, um einen höheren Abstraktionsgrad bei der Speicherung von Wissen zu ermöglichen. Das Ziel ist es, dieses Wissens so vorzuhalten, dass eine einfache Anbindung der Repräsentation an ein KBE System möglich wird. Weiter zeigt MOKA eine Prozesskette, ausgehend von der Wissenserfassung, über die Überprüfung und Darstellung bis hin zu Archivierung dieses Wissens, auf (MOKA Lebenszyklus).

Wesentliches Merkmal von MOKA ist also die Bereitstellung einer objekt-orientierten Methode, welche - anders als KADS - eigens auf die Anforderungen der Entwicklung von KBE Systemen zugschnitten ist. MOKA ist unabhängig von kommerziellen, proprietären KBE Systemen.

4.2.1 Der MOKA Lebenszyklus

Sowohl die Implementierung als auch die Erweiterung eines KBE Systems, unterscheidet sich nicht wesentlich von der Entwicklung anderer Software. Ausgehend von einer Idee oder einem konkreten Problem, zum Beispiel ein zu optimierender Teilprozess innerhalb des Entwurfs, beginnt die Entwicklung. Weitere Gründe für Innovationen können die strategischen Ziele eines Unternehmens, wie etwa die Ambition Technologieführer in einem dezidierten Segment der Branche zu werden, sein. Am Ende des iterativen Entwicklungsprozesses steht ein Softwaresystem, das den Entwurfsprozess verbessert oder besser unterstützt und so zu einem betriebswirtschaftlichen Nutzen führt.

Diesen Lebenszyklus unterteilt die MOKA Methodik in sechs Schritte (siehe Abbildung 4.5), nämlich *Identify*, *Justify*, *Capture*, *Formalize*, *Package*, *Activate*. MOKA fokussiert dabei auf die Schritte *Capture* und *Formalize*. Im Folgenden werden diese sechs Schritte wie in (Stokes 2001) beschrieben, näher erläutert:

Identify Der erste Schritt ist die konzeptionelle Spezifikation der neuen KBE Anwendung. Hier werden die Anforderungen an das neu zu implementierende System untersucht und KBE Typen ermittelt, die diese Anforderungen erfüllen könnten. Weiter wird geprüft, wer hinsichtlich Wissen, Personen, Werkzeugen und Techniken, Hardware und Software in der Lage ist, ein solches System aufzubauen. Es wird ebenfalls berücksichtigt, ob mit Hilfe des zu entwickelnden KBE Systems die angestrebten Ziele überhaupt erreicht werden können oder andere Lösungen möglicherweise sinnvoller sind. Außerdem wird die technische Machbarkeit des Vorhabens geprüft und umfangreiche Informationen über Ziele und Umfang der Anwendung zusammengetragen. Bei der Umsetzung dieses Schrittes sollten alle Projektbeteiligten also Manager, Experten, Entwickler, Anwender und andere, ihre jeweilige Sichtweise erklären um frühzeitig Konflikte auszuschließen. Manchmal wird der *Identify*-Schritt erneut von *Activate* angestoßen, weil beispielsweise weitere Informationen der Implementierungsumgebung benötigt werden.

Justify Dieser Schritt sieht die Erstellung eines Projektplans für die Entwicklung des KBE Systems vor, welcher zusammen mit dem *Business* Case der Einholung der Managementfreigabe für alle weiteren Schritte dient. Die Erstellung des Projektplans ist die wesentliche Aufgabe der Wissensingenieure und Entwickler, die die Information von allen Beteiligten zusammentragen und in einem Projektplan aufbereiten. Die bereits im *Identify*-Schritt ausgearbeiteten Ziele, werden nun in Akzeptanzkriterien überführt. Außerdem enthält der Projektplan eine detaillierte Beschreibung der neuen KBE Anwendung. Der erfolgreiche Abschluss dieses Schrittes ist die Projektfreigabe seitens des Managements. Dies führt zu einer weiter differenzierten Ausarbeitung des Projektplanes, der die technischen, kulturellen und finanziellen Risiken, die nötigen Ressourcen und Kosten, sowie einen Zeitplan des Gesamtprojekts enthält.

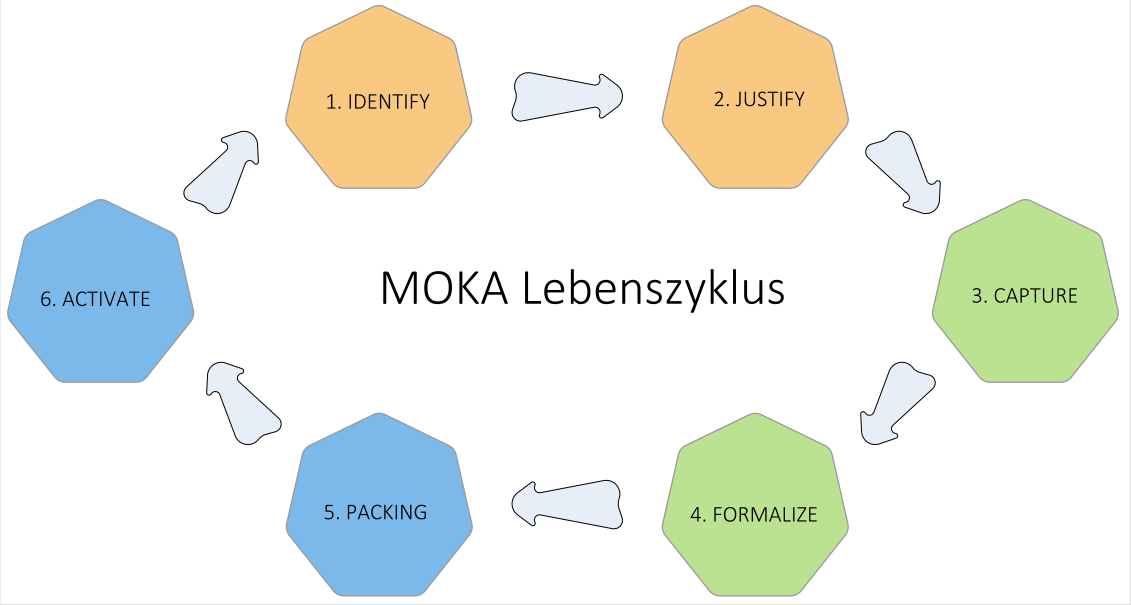


Abbildung 4.5: Der MOKA Lebenszyklus (Stokes 2001)

Capture Dieser Schritt besteht aus zwei wesentlichen Aktivitäten, nämlich die Sammlung des Wissens und die Strukturierung des Wissens mit Hilfe von ICARE in das *Informal Model* (siehe 4.2.2).

Formalize In diesem Schritt wird aus dem *Formal Model* das *Informal Model* erstellt. Dafür wird Modellierungssprache MOKA Modelling Language (MML) verwendet (siehe 4.2.3 und 4.2.4).

Package Dieser Schritt beinhaltet die Implementierung des *Formal Model*s zu einem funktionierenden KBE System, durch die Softwareentwickler. Dies erfordert die Übersetzung des *Formal Models* in eine fertiges, direkt nutzbares KBE System. Da das *Formal Model* in einem neutralen Datenformat erstellt wird, existiert gegebenenfalls kein geeignetes Mapping zwischen diesem Modell und einer existierenden, kommerziellen KBE Plattform. Daher wird dieser Schritt entweder manuell oder mit Hilfe eines eigens zu entwickelnden Tools durchgeführt.

Activate Der letzte Schritt im MOKA Lebenszyklus ist die Verteilung der KBE Anwendung auf die Computersysteme der Anwender, die Einführung des Systems bei den Benutzern und letztlich die Nutzung des Systems zur Entwicklung eines Produkts. Die Ausprägung dieses Prozesses hängt stark von den Umständen der KBE Entwicklung ab. Ist das KBE System nur eine kleine Erweiterung eines bestehenden Systems, wird der Aufwand sehr viel kleiner sein als bei Einführung eines komplett neuen Systems. In einem Unternehmen das bereits KBE nutzt, besteht dieser Schritt lediglich aus der Einführung des entwickelten Moduls in das bestehendes KBE Framework.

4.2.2 Informal Model

Um Wissen innerhalb des *Capture*-Schrittes abhängig von einem Produktmodell zu sammeln und zu strukturieren, werden sogenannte *Forms* verwendet. Das *Informal Modell* beinhaltet fünf verschiedene *Forms*, sogenannte *ICARE Forms* (Illustration Constraint Activity Rule Entity Forms). Das Wissen ihres Typs entsprechend diesen Formen zugeordnet werden:

Illustration Die *Illustration Form* speichert allgemeine Informationen, Case-Studies, Kommentare, Hinweise et cetera. Es kann jeder anderen Form innerhalb des *ICARE* Systems angehängt werden.

Constraint Die *Constraint Forms* enthalten Wissen über Einschränkungen für Produktobjekte und können einer oder mehreren *Entities* angehängt werden. Sie können für die ganze Entity oder nur einzelne Attribute gelten. *Constraints* können sowohl globale das ganze Produkt betreffende Einschränkungen definieren oder nur lokal beispielsweise ein *Part* einschränken. *Constraints* können die Existenz, Form und Größe, Material, Technologie oder andere Aspekte eines Produkts limitieren.

Activity Diese Form speichert Wissen des Entwurfsprozesses, es stellt also das "was soll geschehen" eines Prozesses zur Verfügung. Es speichert unter anderem Berechnung, Selektierung, Filterung, Positionierung, Dimensionierung einer Struktur. Es beinhaltet Wissen aller Detailgrade, von *Top-Level-Ebene* bis in kleinste Details einzelner Basiselemente.

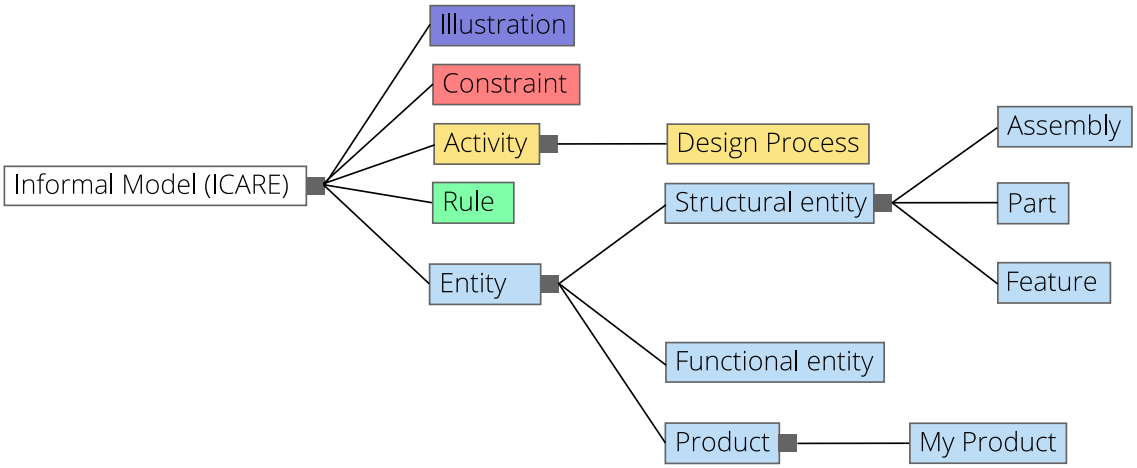


Abbildung 4.6: Informal Model der MOKA Methodik (Stokes 2001)

Rule Jede *Rule Form* ist einem Entwurfsprozess zugeordnet. Sie speichert das Wissen, welches die *Activity Forms* steuert, es stellt also das „wie soll es geschehen" eines Prozesses zur Verfügung. Wissen, das in *Rule Forms* abgelegt ist, beinhaltet die Algorithmen und Operationen die in den *Activitiy Forms* zur Durchführung eines Prozesses benötigt werden.

Entity Diese *Form* beinhaltet Wissen über das Produkt selbst. Es wird weiter unterteilt in *Structural*, *Functional* und *Bahavioural Entities*. In der

Structural Entity wird das Produkt hinsichtlich seines Aufbaus aus *Parts*, *Assemblys* und *Features* beschrieben. Um die Funktionalität eines Produkts, mögliche Designlösungen, konzeptionelle Entwürfe zu speichern, steht die *Functional Entity* zur Verfügung. Diese Form bietet Platz für Wissen, das Informationen zu Geometrie, Material und Herstellung speichert. Diese *Form* ist zusammen mit den *Activity Forms,* die wichtigste bezüglich der Wissenserfassung.

In (Ammar-Khodja et al. 2008) wurde eine Erweiterung der *ICARE Forms* von MOKA um die *Entities Resource* und *Function* vorgeschlagen (ICARREF). Damit sind die Autoren in der Lage, die Integration von Prozessplanungswissen in einem KBE System besser abzubilden. (Barreiro et al. 2009) wiederum verwenden nur die zusätzliche *Entitity Resource* (ICARER) zur Modellierung Wissensbasierter Systeme für die Inspektionsplanung. Sie sind der Meinung, dass die *Entity Function* mit den bereits vorhandenen *Entities Activity* und *Rule* gleichwertig abgebildet werden kann. (Skarka 2007) zeigt die Anwendung des *Informal Model* aus MOKA im CAD-System CATIA.

4.2.3 Formal Model

Für den *Formalize*-Schritt des MOKA Lebenszyklus wird ein gänzlich anderes Modell verwendet, das *Formal Model*. Es wird mit Hilfe der MOKA Modelling Language (MML) (siehe 4.2.4) beschrieben. Um den Nutzen des *Formal Model* zu verstehen ist es sinnvoll zunächst einen Schritt vorauszuschauen: Die Implementierung des Formal Models in einer KBE Plattform. Die meisten KBE Plattformen benutzen eine Reihe von Befehlen, die Anweisungen in formalen Programmiersprachen sehr ähneln. Die Wissensrepräsentation einer implementierten KBE Plattform ist also viel detaillierter als die des *Informal Model*. Das *Informal Model* bietet nur eine eingeschränkte Unterteilung des Wissens in Objekte. Es klassifiziert diese lediglich in fünf Kategorien. Zudem sind die Beziehungen zwischen diesen Objekten in Textform beschrieben.

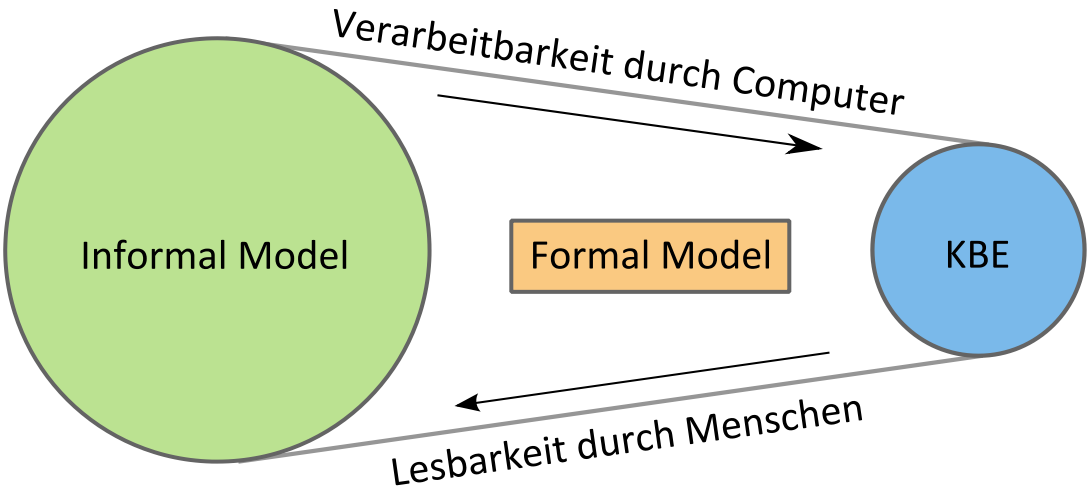


Abbildung 4.7: Vom Informal Model zu KBE (Stokes 2001)

Um die Repräsentationslücke zwischen *Informal Model* und KBE System zu schließen, wurde das *Formal Model* eingeführt. Mit Hilfe des *Formal Model* ist es möglich das Wissen ohne Programmierkenntnisse genauer abzubilden. Das Wissen wird im Vergleich zum *Informal Model* in weit mehr und kleinere Objekte und Kategorien unterteilt, mit genauerer Beschreibung der Beziehungen zwischen unter den Objekten. Dadurch beschreibt das *Formal Model* die Struktur einer Wissensrepräsentation auf eine Weise, die sowohl für den Menschen als auch den Computer interpretierbar ist. Abbildung 4.7 veranschaulicht diese Einordnung des *Formal Models* zwischen *Informal Model* und implementiertem System.

Innerhalb des *Formal Model* existieren zwei eigenständige Teile, das *Design Process Model* und das *Product Model*. Abbildung 4.8 zeigt die Beziehungen der beiden Modelle untereinander. Durch die Aufteilung von *Design Process* und *Product Model* ist eine Verwendung des *Product Models* in einer weiteren Anwendung möglich. Das Design Process Model definiert die Reihenfolge der einzelnen Entwurfsprozesse und die Abfolge der zu treffenden Entscheidungen. Das *Product Model* speichert Wissen über den strukturellen Aufbau und die Anordnung von Teilen, Geometrie, das Verhalten unter variierenden Bedingungen, Material und Herstellung und die Einschränkungen des Produktmodells. Das *Product Model* von MOKA korrespondiert mit der *Domain Knowledge* Ebene des *Knowledge Model* aus KADS (siehe 4.1.2).

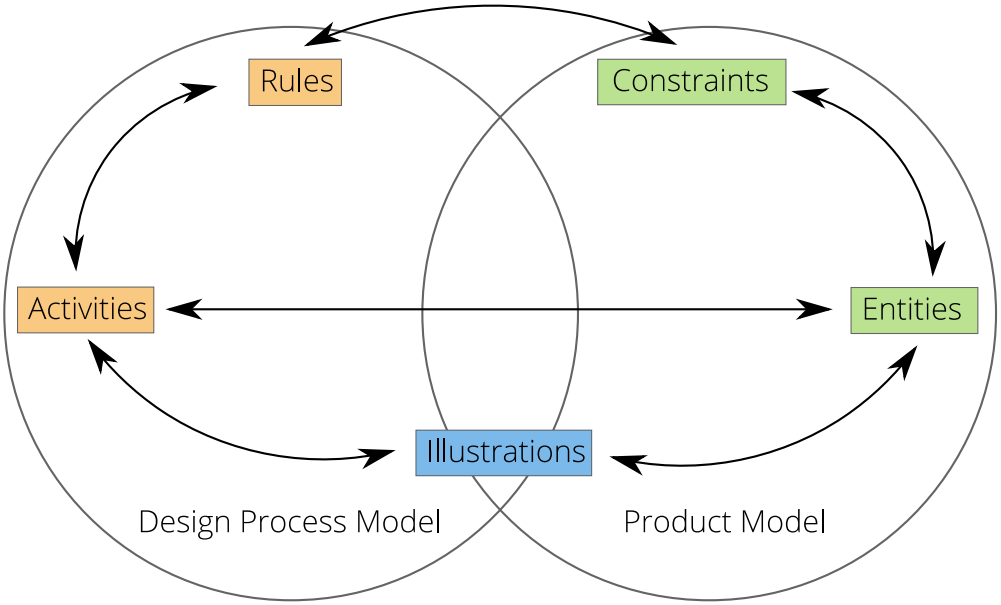


Abbildung 4.8: Beziehungen zwischen Design Process Model und Product Model (Stokes 2001)

Da selbst bei einfachen Produkten die Zahl der Wissensobjekte sehr groß sein kann, besitzt das *Product Model* fünf vordefinierte Views. Dabei werden die Wissensobjekte in *Structure*, *Function*, *Behaviour*, *Technology* und *Representation View* (siehe Abbildung 4.9) gegliedert (Stokes 2001):

Structure View Die *Structure View* definiert den Aufbau der Produktstruktur mit Hilfe von *Assembly*, *Part* und *Feature*. Sie kann physikalische, logische und konzeptionelle Strukturen in jeder Entwurfsphase abbilden.

Function View Sie speichert den funktionellen Aufbau des Produkts und wie dieser Aufbau mit Hilfe verfügbarer Technologie realisiert werden kann.

Behaviour View Die *Behaviour View* speichert das Produktmodell in verschiedenen Zuständen, zum Beispiel „zusammengebaut“, „einzeln“ und die Übergange von einem Ausgangszustand in einen Zielzustand.

Technology View Speichert Material und Herstellungsprozesse des Produkts.

Representation View Beinhaltet ebenfalls Geometrie, welche aber auf jede Größe und Form erweitert werden kann, zum Beispiel Finite Elemente.

4.2.4 MOKA Modelling Language

Während der Entwicklung des MOKA Projekts stellte sich heraus, dass kein brauchbares Modellierungsformat existierte um Ingenieurwissen abzubilden (Stokes 2001). Die vorhandenen Formate waren entweder zu allgemein oder zu speziell für eine unmittelbare Anwendung in MOKA. Daher wurde zur Abbildung des *Formalen Modells*, eine eigene für MOKA passende, aber von existierenden Formaten abgeleitete Modellierungssprache entwickelt (Brimble und Sellini 2000). Die Sprache heißt MOKA Modelling Language (MML). Die Anforderungen an die Modellierungssprache sind:

- Formalisierung des gesamten Wissens der Domäne
- Vermittelnde Sprache zwischen Experten, Wissensingenieuren und Entwicklern
- Graph-basiert
- Ausreichend formal für automatisierte Code Generierung
- Erweiter- sowie anpassbar durch Benutzer

Die Unified Modelling Language (UML) erfüllt bereits die meistesn Anforderungen (Graph-basiert, objekt-orientiert, erweiterbar) und wurde ergänzt, um formal das Ingenieurwissen innerhalb von KBE Systemen zu erfassen und zu speichern. Durch die Bereitstellung von vorgefertigten *Klassen*, *Assoziationen* und *Attributen* stellt sie dem MML Benutzer Best-Practice Methoden in strukturierter und logischer Art und Weise zur Verfügung.

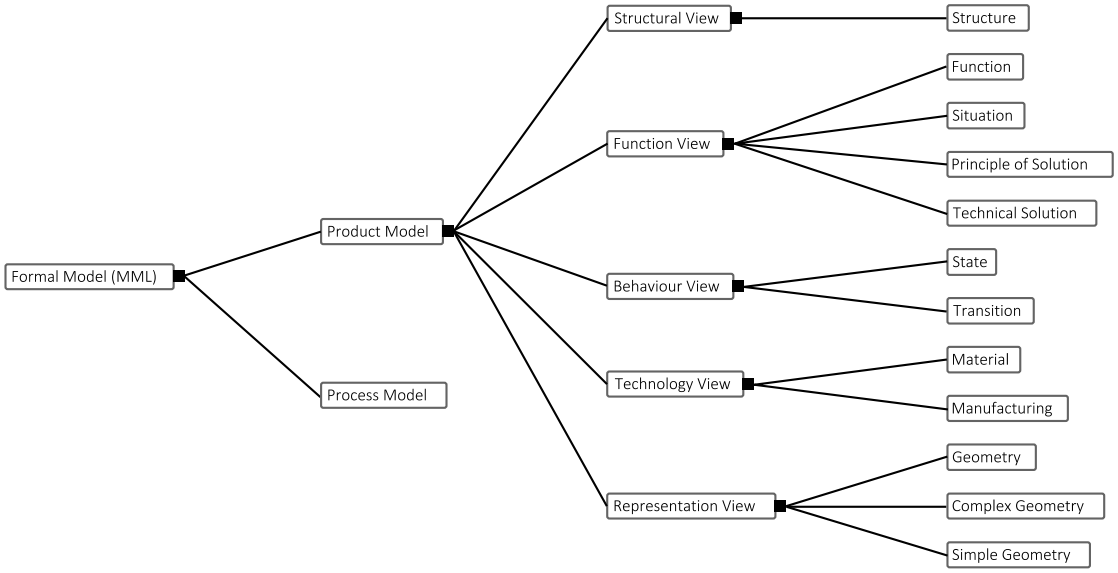


Abbildung 4.9: Formal Model der MOKA Methodik abgebildet als MML (Stokes 2001)

MML stellt folgende vorgefertigte Objekte zur Verfügung:

View Stellt unterschiedliche Perspektiven auf das betreffende Modell zur Verfügung und bestimmt damit den erwarteten Inhalt einer Ansicht. Die Kerndefinitionen werden durch *Structural*, *Functional* und *Behaviour* Views abgebildet.

Klassen Stellt dem Benutzer Klassen zur Verfügung, um diese für die Modellierung zu nutzen. Zum Beispiel die Klasse *<Part>*.

Klassen-Attribute Erfasst häufig genutzte Attribute für einen bestimmten Klassentyp. Zum Beispiel besitzt jede Klasse *Assembly* das vordefinierte Attribut *Assembly_Operation*.

(Abdullah et al. 2005) kritisiert MML als informelle Erweiterung von UML, die nicht den Anforderungen der Object Management Group (OMG) an einen Erweiterungsmechanismus erfüllt. Die OMG hat zwei Mechanismen zur Erweiterung von UML definiert: *profiles* und *metamodel extensions*.

4.3 KNOMAD

KNOMAD steht für **K**nowledge **N**urture for **O**ptimal **M**ultidisciplinary **A**nalysis and **D**esign und ist eine Methode zur analytischen Nutzung, Entwicklung und Abbildung von multidisziplinären Ingenieurwissen innerhalb der Produktentwicklung und Produktion (Curran et al. 2010):

Der Schritt der Wissenserfassung von KNOMAD ähnelt sehr den Schritten *Identify*, *Justify* und *Capture* von MOKA. Der Schritt *Organisation* hat einige Ähnlichkeiten mit dem *Formalize*-Schritt von MOKA und kann daher ebenfalls MOKAs *Formalen Modell* zugeordnet werden. Trotz dieser Gemeinsamkeiten, kommt auch hier das bekannte Manko von MOKA zum Tragen: Der Mangel an Implementierungsbeispielen. So geht der *Formalize*-Schritt nicht mit Tools, Beispielen oder implementierten Case-Studies aus der Literatur einher. Beispiele würden aber die Adaption der MOKA Methode erheblich bereichern. KNOMAD geht über MOKA hinaus, in dem es eine Herangehensweise für multidisziplinäres Modellieren und Analysieren in den Schritten *Modeling* und *Analysis* bereitstellt. Außerdem bietet der *Normalisation*-Schritt für die Wissensakquirierung einen Qualitätsprozess, welcher in MOKA nicht explizit vorhanden ist. Die Trennung von nützlichem, glaubwürdigem und nachweisbarem Wissen von Irrelevantem ist ein entscheidender Teil der Wissensmodellierung.

5 Knowledge-based Engineering

Nach Einstieg in Wissen, Wissensbasierte Systeme sowie methodischer Grundlagen, widmet sich dieses Kapitel nun speziell dem Themengebiet des Knowledge-based Engineerings. Darin sind zunächst verschiedene Definitionen von und eine Einführung in Knowledge-based Engineering enthalten. Anschließend ist die Entwicklung von KBE seit 1980 beschrieben. Es folgt eine umfangreiche Literaturübersicht zu KBE. Ferner werden einige KBE Programmiersprachen und Typen von KBE Systemen vorgestellt. Eine Diskussion der Vor- und Nachteile einer KBE Adaption ist ebenfalls aufgeführt. Zum Schluss werden kommerzielle KBE Software Systeme vorgestellt.

5.1 Definition und Einführung

Die Literaturrecherche zur Begriffsdefinition KBE, ergab überwiegend unklare Definitionen. Daher sind nachstehend die aussagekräftigsten und bekanntesten Definitionen verschiedener Autoren zusammengestellt:

Knowledge-based Engineering wird durch (Stokes 2001) definiert als:

> *„The use of advanced software techniques to capture and re-use product and process knowledge in an integrated way."*

Eine weitere Definition von KBE ist in (Chapman und Pinfold 1999) zu finden:

> *„[…] an engineering method that represents a merging of object-oriented programming, Artificial Intelligence techniques, and computer-aided design technologies, giving benefit to customized or variant design automation solutions".*

In seiner Dissertation zum Thema „Knowledge based engineering techniques to support aircraft design and optimization" (La Rocca 2011) definiert La Rocca KBE wie folgt:

> *„Knowledge Based Engineering is a technology based on the use of dedicated software tools called KBE system, which are able to capture and systematically reuse product and process engineering knowledge, with the final goal of reducing time and costs of product development by means of the following:*
>
> - *Automation of repetitive and non-creative design tasks*
> - *Support of multidisciplinary design optimization in all the phases of the design process."*

(Bermell-Garcia und Ip-Shing 2002) definieren KBE wiederum als:

> *„[…] a special type of Knowledge Based Systems with a particular focus on product engineering design and downstream activities such as analysis, manufacturing, production planning cost estimation and even sales."*

Aus diesen Definitionen geht klar hervor, dass in Abhängigkeit der jeweiligen Standpunkte der Autoren, Definitionen zu KBE variieren. Wesentlich ist aber der Einsatz von Wissensbasierten Systemen im Ingenieurwesen beziehungsweise zur Erfassung von Ingenieurwissen und dessen Wiederverwendung in der Produktentwicklung. Abbildung 5.1 veranschaulicht die Herkunft von KBE aus der Künstlichen Intelligenz (KI).

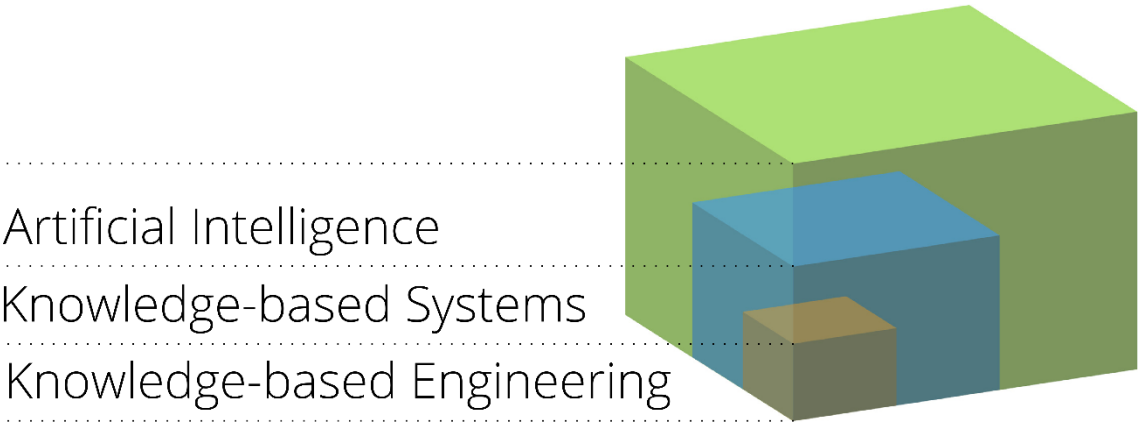


Abbildung 5.1: KBE - ein spezieller Typ eines KBS

5.2 Entwicklung von KBE

Die Entwicklung von Knowledge-based Engineering ist ein typischer Vertreter des Gartner Hype Cycle (Abbildung 5.2). Mit den ersten Anwendungen in den 1980er Jahren versprach die Anwendung von KBE großes Potential. Doch trotz aller aufgezeigter Vorteile, hat KBE, abgesehen von der Anwendung bei wenigen Automobil- und Flugzeugherstellern, bis heute keinen überzeugenden Durchbruch geschafft. Als Gründe für diesen schleppenden Erfolg von KBE hat (La Rocca 2012) folgende Punkte identifiziert:

1. Teure Software und Hardware
2. Mangel an Literatur, Fallstudien und Erfolgsgeschichten
3. Schwieriger Einstieg in die komplexe Thematik
4. Mangel an Methoden zur Entwicklung von KBE Systemen
5. Fragwürdige Marketingstrategie der Softwarehersteller

Aufgrund der technologischen Weiterentwicklungen in den letzten Jahren sind diese Punkte kaum mehr relevant. Die Kosten für Hardware sind gesunken, KBE Tools werden mit bestehenden Softwaresystemen kostenlos ausgeliefert und Methoden wie MOKA wurden entwickelt. Damit steht einer positiven Entwicklung von KBE in den nächsten Jahren nichts mehr im Wege (Plateau of Productivity).

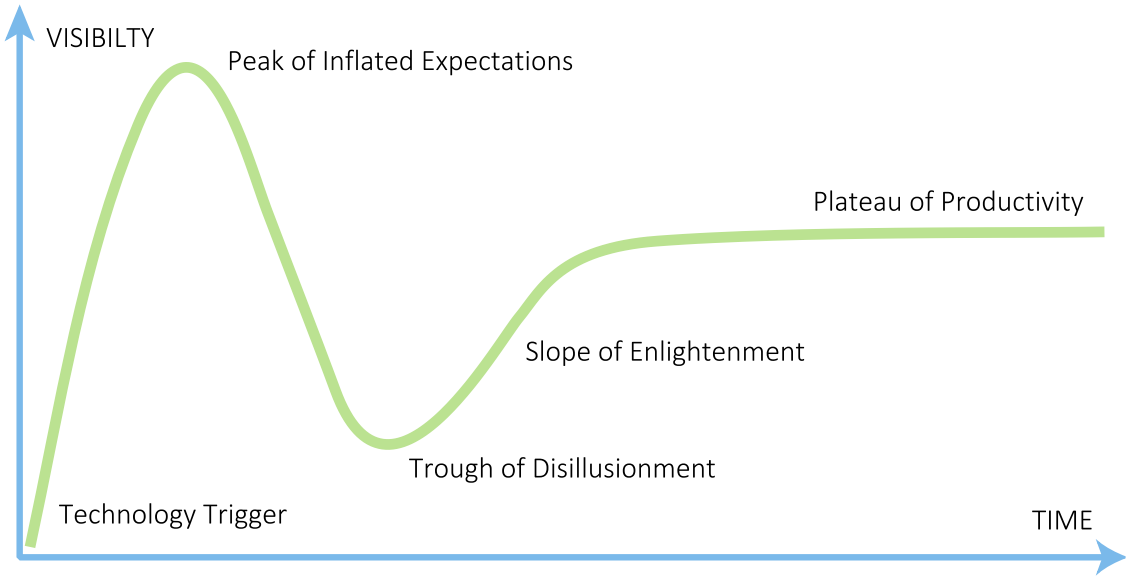


Abbildung 5.2: Gartner Hype Cycle (Gartner 2014)

In (McGoey 2011) wird die Entwicklung der KBE Anwendung in den verschieden Industriezweigen seit 1980 bis heute treffend beschrieben:

Zeitraum	KBE Anwendung in der Industrie
1980 - 1985	Erkundungsphase – Schiffsegel – Komplexes Tragflächendesign von Flugzeugen – Düsentriebwerkdesign von Flugzeugen
1985 - 1990	Erste Anwendungen - Aerospace: Ganzheitliches konzeptionelles Design, Rumpf - Detailentwurf Strukturen, Motoren - ACE: Rapid Design: HLK Systeme, Kühltürme, Brücken - Automotive: Räumliche Integration Konzeptauto - Sondermaschinenbau: Pumpen, Turbinen, Ventile
1990 - 2000	Globale Anwendung - Aerospace: Detailentwurf der Rumpf- und Flügelstruktur - Integration Systemstrukturen, Entwurf Wiederverwendung - Automotive: Antriebsstrang, Plattformkomponenten - Werkzeugbau - ACE: Gebäudestruktur, HLK Systeme, Anlagenbau Chemie
2000 - 2010	Weiterentwicklung durch CAD / CAM Hersteller - Spezialisierung auf die Märkte der CAD Hersteller: Aerospace, Automotive, ACE - Expansion nach Asien / Globale Liefer- und Entwurfsketten - Nivellierung des Nutzenwachstums in Aerospace & Automotive
ab 2010	Entwicklung von standardisierten KBE Anwendungen

5.3 Literatur zu KBE

Der Einsatz von KBE Methoden beziehungsweise Wissensbasierten Systeme wurde in anderen Ingenieurgebieten bereits wesentlich tiefer erforscht als im Bauingenieurwesen. Dieses Kapitel soll einer strukturierten Übersicht über verschiedene Forschungsschwerpunkte der einzelnen Branchen dienen. Es bietet also einen schnellen Einstieg in den Stand der Technik hinsichtlich Knowledge-based Engineering. Zunächst wird Literatur zu allgemeineren Themen bezüglich KBE aufgeführt. Anschließend folgen Literaturbeispiele im Bereich Offshore und Aerospace, Multidisciplinary Design Optimization (MDO) und Automobilbau. Zum Schluss wird Literatur auf Literatur hingewiesen, die sich mit der Anwendung von KBE im Bauingenieurwesen und KBE in Verbindung mit Building Information Modelling beschäftigt.

Allgemein

Einen guten Überblick über die bisherige Entwicklung, den aktuellen Stand und zukünftige Trends des KBEs fasst (La Rocca 2012) zusammen. Der Bericht möchte bisherige wissenschaftliche Informationslücken rund um das technologische Fundament von KBE schließen, da sich die bis dato vorhandene wissenschaftliche Literatur ausschließlich der Anwendung von KBE, nicht aber den technologischen Hintergrund von KBE widmet.

Ein weiteres, häufig rezitiertes Werk ist (Chapman und Pinfold 1999). Sie diskutieren die Einschränkungen von traditionellen CAD-System und berichten über den Einsatz von KBE für einen besseren Organisationsfluss und schnellere Designlösungen. Zum Zeitpunkt der Veröffentlichung war es eines der wenigen wissenschaftlichen Publikationen zu KBE.

Der Best-Practice Guide (Cooper et al. 1999) möchte ebenfalls einen Einstieg in das Thema KBE aus Perspektive der Industrie geben. Das Handbuch zeigt die Auswirkungen der KBE Anwendung für eine Firma auf und gibt Empfehlungen für die Implementierung und Nutzung von KBE. Der Bericht basiert auf den Erfahrungen der beteiligten Firmen, unter anderem British Aerospace, British Steel und Jaguar sowie weiteren Fallstudien.

(Sainter et al. 2000) beschäftigt sich mit der Frage, wie Wissen innerhalb eines KBE Systems verwaltet werden kann, damit das gespeicherte Wissen nicht an Wert und Nutzen

über den gesamten Produktlebenszyklus verliert. Dafür wurde ein Schema für die Entwicklung und Verwaltung von Produktwissen innerhalb eines KBE Systems vorgestellt, um die Vorteile eines KBE Systems auch langfristig sicherzustellen.

Eine Methode zur Anwendung von KBE in Klein- und Mittelständischen Unternehmen (KMU) stellt (Lovett et al. 2000) vor. Die bekannten und auch im Rahmen dieser Arbeit vorgestellten Methoden CommonKADS und MOKA (siehe Kapitel 4) sind aufgrund ihrer Komplexität für die Anwendungen in KMUs ungeeignet.

In (Verhagen et al. 2012) wurde eine umfangreiche Literaturstudie zu KBE durchgeführt. Insgesamt konnten 50 Arbeiten gesammelt und analysiert werden. Wesentliche Ziele waren die Identifizierung von theoretischen Grundlagen, Forschungsbedarf sowie die Untersuchung von Herausforderungen und Schwierigkeiten, die einer verbreiteteren Anwendung von KBE im Wege stehen.

Beispiele Offshore

Kurze Planungszeiten und die Fähigkeit den Stakeholdern bereits in frühen Entwurfsphasen eine zufriedenstellende Detailtiefe der Planung zu präsentieren sind wichtige Wettbewerbsvorteile in der heutigen Planung von Offshore Plattformen. (Kalavrytinos und Sievertsen 2014) untersuchen den Nutzen von KBE Systemen für die Planung dieser Anlagen.

Beispiele Flugzeugbau

Der Entwurf und Bau von 1:n Testmodellen eines Flugzeugs für die Untersuchung in Windkanälen ist ein wichtiger, aber gleichzeitig zeitintensiver Prozess während der Entwicklung eines Flugzeugs. (Bermell-Garcia und Ip-Shing 2002) untersucht die Anwendung von KBE zur Automatisierung dieses Entwicklungsschrittes.

Amadori untersucht in (Amadori 2012) und (Amadori et al. 2012) Methoden zur multidisziplinären Optimierung (Multidisciplinary Design Optimization (MDO)) in frühen Phasen des Flugzeugentwurfs. Dafür ist ein geometrisches Modell notwendig, welches automatisch generiert oder verbessert werden kann. Um diese Optimierung zu erreichen, schlägt er ein KBE System zur Designwiederverwendung und -automatisierung sowie die Anwendung so genannter High Level CAD templates (HLCts) vor.

Designer wünschen sich eine anwenderfreundliche und benutzerspezifische Designkonfiguration, um auch kreative und innovative Entwürfe einfach und schnell beurteilen zu können. Der enorme Arbeitsaufwand zur Erstellung proprietärer Modelle wie beispielswiese für FEM- und/oder CFD-Programmen, hinderte die Designer bisher daran, viele Designalternativen zu entwerfen. In (La Rocca und van Tooren 2007) wird die Anwendung von KBE für diesen Prozess aufgezeigt. Auch die Arbeiten (La Rocca et al. 2005) (La Rocca et al. 2009) beschäftigen sich mit dieser Fragestellung.

Das Potential der Anwendung von KBE Methoden in der Produktentwicklung arbeitet (Corallo et al. 2009) aus. Es wird der betriebswirtschaftliche Mehrwehrt der KBE Anwendung in einer italienischen Luft- und Raumfahrtfirma mit Hilfe einer dreijährigen Studie bewertet.

Um in kurzer Zeit Designalternativen für Rotorblätter einer Windkraftanlage zu entwerfen, diskutiert (Chiciudean et al. 2008) den Einsatz von KBE. Das System soll in der Konzept- und Vorentwurfsphase der Anlagenentwicklung zum Einsatz kommen. Dabei wird im Besonderen die Verteilung der Arbeit in Speziallistenteams an verschieden geographischen Standorten berücksichtigt.

Beispiele Automobilbau

Die Anwendung von KBE für das Design einer Fahrgastzelle behandelt (Chapman und Pinfold 2001). Es wird ein KBE System vorgestellt, welches während der Produktentwicklung sehr flexibel auf neue Anforderungen innerhalb vorgegebener Randbedingungen reagieren kann und dabei die Auswirkungen auf andere Bauteile ebenfalls berücksichtigt. (Pinfold und Chapman 2001) erweitert dieses System um die wissensbasierte Generierung von FEM-Netzen einer Fahrgastzelle.

Wissensbasierte Systeme im Bauwesen

Bereits 1988 setzte sich (Adeli und Balasubramanyam 1988) mit dem wissensbasierten Entwurf von Stahlfachwerkbrücken auseinander. Dabei wurde prototypisch ein wissensbasiertes System zur Optimierung vier verschiedener Brückentypen entwickelt. Das System konnte als „intelligenter Assistent“ während des Brückenentwurfs verwendet werden, um ein optimales Design des Brückenüberbaus zu finden.

(Chassiakos et al. 2005) zeigt ein Wissensbasiertes System zur Instandhaltungsplanung von Autobahnbrücken aus Stahlbeton auf. Das System sieht Funktionen zur Priorisierung einzelner Brücken, Wahl des Beurteilungsverfahrens je Fall und Instandhaltungsplanung für die Brückenlager vor. Die Priorisierung wird mit Hilfe einer Rangliste mit gewichteten Entscheidungsparametern realisiert.

Zur computergestützten zerstörungsfreien Analyse und Simulation von Straßenoberbauten für Instandhaltungszwecke stellt (Evdorides 1993) ein wissensbasiertes System vor. Da die bis dato vorgestellten Systeme keine sinnvollen Lösungen ohne Interpretation der Analyseergebnisse mit Hilfe von Expertenwissen liefern konnten, wurde für diese Aufgabe ein Prototyp eines Wissensbasierten Systems entwickelt. Das System soll die bestehenden Methoden mit dem Ingenieurwissen verknüpfen, um so zuverlässige Ergebnisse zu liefern.

Um die Wahl von Bauverfahren wissensbasiert zu unterstützen, untersucht (Ferrada und Serpell 2013, 2014) entsprechende Methoden. Dafür wurde der Prozess der Bauverfahrenswahl mit Hilfe von Fallstudien erfasst und festgestellt, dass Ingenieurwissen diesbezüglich häufig nur implizit vorliegt und keine standardisierten Verfahren existieren.

Strategien zur Sammlung und Wiederverwendung von impliziten Expertenwissen innerhalb der Baubranche stellt (Woo et al. 2004) vor. Außerdem wird ein Konzept eines Prototypen, genannt Dynamic Knowledge Map, aufgezeigt, was die Wiederverwendung von Expertenwissen unterstützt. Dynamic Knowledge Map ist ein webbasiertes Tool, um Experten zu einem bestimmten Wissensgebiet zu finden.

Die Entwicklung und der Stand von Knowledge Management (KM) in der Baubranche untersucht (Rezgui et al. 2010). Weiter wird ein KM Framework vorgestellt, welches a) das Potential Einzelner, von Teams und Organisationen in der Baubranche, b) die Struktur der Informations- und Kommunikationstechnologie (ICT) und c) die Philosophie des Baumanagements berücksichtigt.

Wissensmodellierung, Wissensbasierte Systeme und KBE mit BIM

Mit dem Thema Wissensmodellierung im Rahmen von BIM beschäftigen sich (Aksamija und Ali 2008) und (Liu et al. 2013). Sie führen das Building Knowledge Modeling (BKM) ein und zeigen die Möglichkeiten und Grenzen der Integration der Wissensmodellierung in BIM auf.

Entscheidungen im Facility Management verlangen die Berücksichtigung vieler verschiedener Informations- und Wissenstypen. Werden sie in der Entscheidungsfindung nicht berücksichtigt hat dies Ineffektivität in der Gebäudeunterhaltung zur Folge. (Motawa und Almarshad 2013) stellt ein System auf Basis von BIM vor, welches es zulässt, dieses Wissen zu erfassen und wiederzugeben.

Ein visuelles BIM- und wissensbasiertes Analysewerkzeug für die Fehlersuche im Facility Management zeigt (Motamedi et al. 2014). Aufgrund der komplexen Interaktion und Abhängigkeiten einzelner Gebäudekomponenten ist es für Techniker schwierig, aus den aufgezeichneten Daten die Hauptursache einer Störung zu finden. Um die Ursachen schneller zu lokalisieren werden auf Expertenwissen basierende Modelle benutzt und anschließend mit Hilfe von gängiger BIM Software visualisiert.

(Voss und Overend 2012) stellt ein Tool vor, welches Building Information Modeling und Knowledge-based Engineering kombiniert, um die Planung von Fassaden zu unterstützen. Während der Herstellung und Installation einer Fassade sind viele Randbedingungen einzuhalten, welche in der Planung der Fassadenelemente berücksichtigt werden müssen. Das vorgestellte Werkzeug erfasst Expertenwissen zu diesen geometrischen Randbedingungen im Hinblick auf den Herstellungsprozess und verknüpft dies mit einem BIM Modell.

5.4 KBE Programmiersprachen

Dieser Abschnitt gibt einen Überblick über die KBE Programmiersprachen. KBE Systeme verfügen allesamt über proprietäre Programmiersprachen, welche dem objekt-orientierten Paradigma folgen. Da KBE Systeme ihren Ursprung in der Künstlichen Intelligenz haben, verwundert es nicht, dass KBE Programmiersprachen häufig auf einem objekt-orientierten Dialekt von LISP (List Processing) basieren. (La Rocca 2011) fasst die verbreitetsten KBE Programmiersprachen wie folgt zusammen:

IDL Die ICAD Design Language, basiert auf der Programmiersprache Common LISP, welche ein Dialekt von LISP ist, inklusive der objekt-orientierten Erweiterung Common LISP Object System (CLOS).

GDL Die General-purpose Declarative Language von Genwork (Genworks 2013) basiert auf der ANSI Standardversion von Common LISP.

AML Die Adaptive Modeling Language von Technosoft (TechnoSoft 2014), wurde ursprünglich in Common LISP geschrieben und nach und nach proprietär neu programmiert, besitzt aber immer noch LISP-ähnliche Merkmale.

Intent! Ist eine proprietäre KBE Programmiersprache von Heide Corporation. Sie ist heutzutage in Knowledge Fusion, ein Modul von Siemens NX integriert. Sie gehört ebenfalls zur Familie der LISP verwandten Programmiersprachen.

Abbildung 5.3 verdeutlicht KBE spezifische und von LISP geerbte Charakteristiken von KBE Programmiersprachen.

Coding features	KBE specific	LISP inherited
Object oriented paradigm	√	√
Declarative coding		√
Dynamic typing		√
Runtime value caching & dependency tracking	√	
Interpreted/compiled mode		√
Automatic Memory management		√
CAD capabilities	√	

Abbildung 5.3: Charakteristik von KBE Sprachen (La Rocca 2011)

5.5 Typen von KBE Systemen

In (Calkins 2000) werden drei grundsätzliche Arten von KBE Systemen unterschieden. Diese sind:

1. Diagnostische Systeme (Expertensysteme)
2. Kreative Systeme (Decision Support Systeme)
3. Generative Systeme (Virtueller Prototyp)

Expertensysteme waren die ersten für die Anwendung im Ingenieurwesen entwickelnden Systeme, zum Beispiel Werkstatt Diagnosesysteme für Autos. Eine Weiterentwicklung sind Werkzeuge zur Unterstützung der Entscheidungsfindung, sogenannte Decision Support Systems. Sie unterstützen den Anwender, indem sie ausgehend von einer konkreten Problemstellung Lösungen anbieten oder Alternativen bewerten. Der Anwender agiert dann anhand der Empfehlung dieses Systems. Generative Systeme beinhalten zusätzlich die regelbasierte Generierung eines Modells. Dieser Virtuelle Prototyp reagiert dann auf geometrische und nicht-geometrische Attributänderungen und regeneriert eine neue Instanz des Modells. Abbildung 5.4 zeigt die Unterschiede und Gemeinsamkeiten von Expertensystem (auch synonym KBS) und KBE Systemen auf.

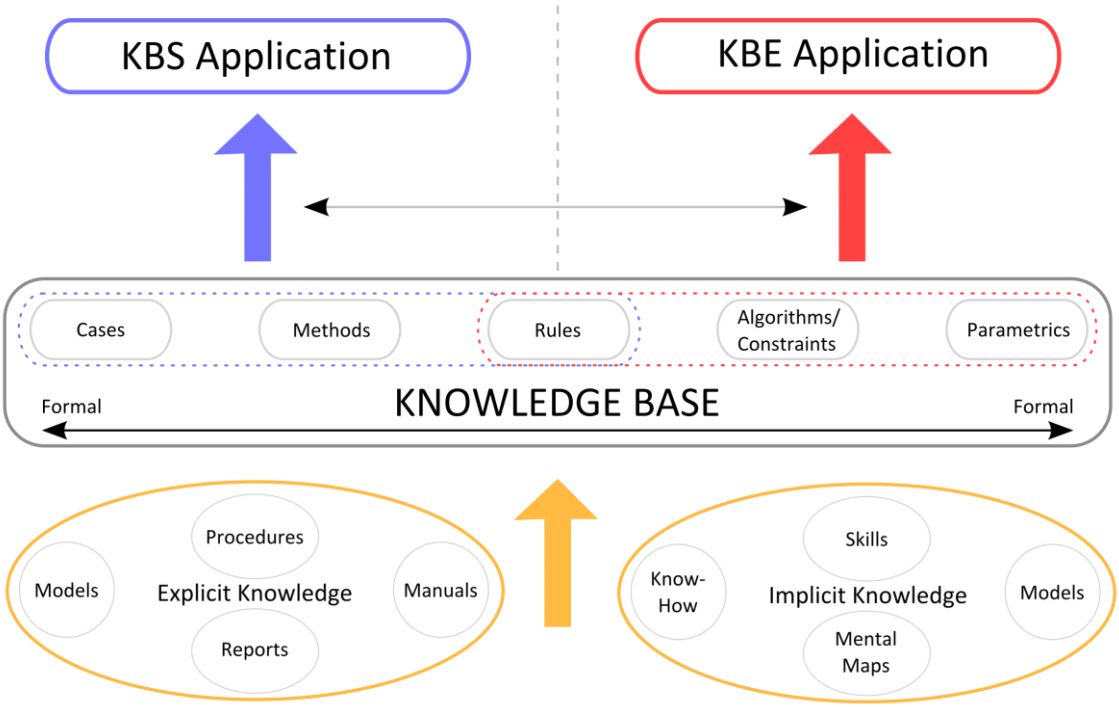


Abbildung 5.4: KBS versus KBE

5.6 Vor- und Nachteile der KBE Anwendung

Es existieren einige Vorteile durch die Implementierung eines KBE Systems im Vergleich zu traditionellem CAD. Der größte Vorteil ist die reduzierte Produktentwicklungszeit durch Rationalisierung und Automatisierung von repetitiven, nicht kreativen Planungsaufgaben. Besonders Produktentwicklungen mit sich stark ähnelnden Aufgaben, einer großen Anzahl an Anpassungsmöglichkeiten und einer großen Anzahl an Designprozessen profitieren von der Anwendung von KBE. KBE ermöglicht einen Planungsprozess in schneller Trial-and-Error Manier. Der Benutzer kann viele „wenn-dann“ Szenarien ausprobieren und so in wesentlich kürzerer Zeit zu einem Ergebnis gelangen. Dieser Effekt schafft außerdem Freiräume für die kreative Lösung weiterer technischer Problemstellungen. In Abbildung 5.5 ist ein Vergleich der Produktentwicklungszeit zwischen KBE und traditionellem CAD dargestellt. Mit Hilfe dieser Grafik werden die Einflüsse einer KBE Adaption auf den Planungsprozess sehr deutlich.

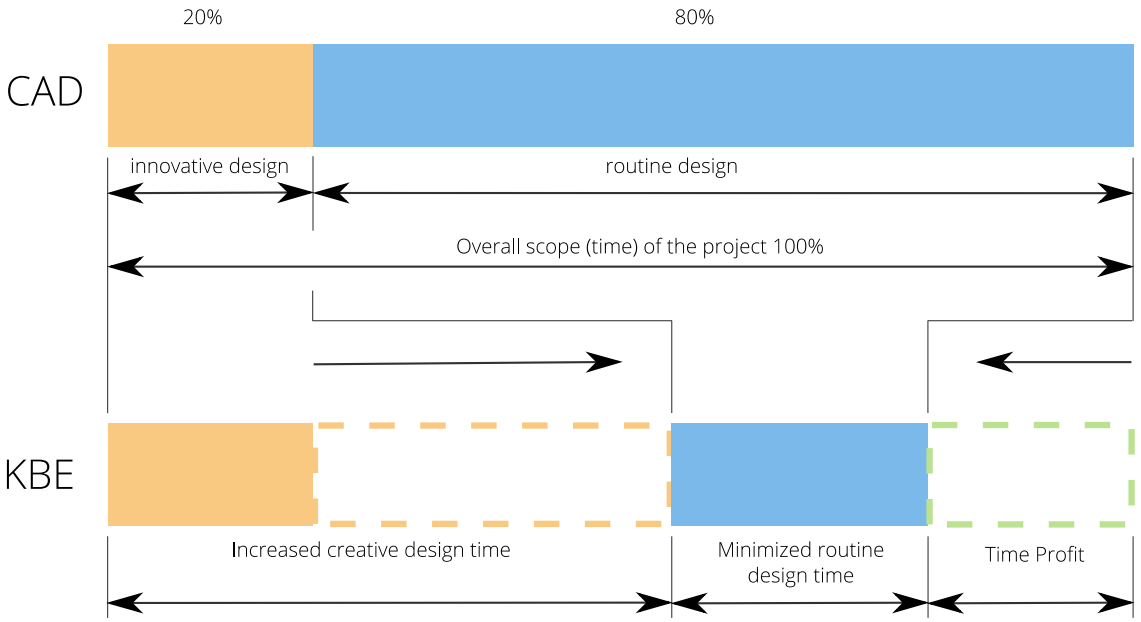


Abbildung 5.5: Vergleich von KBE und traditionellem CAD nach (Stokes 2001)

Die Wissenserfassung und Wiederverwendung birgt bereits in sich Vorteile. Durch die zentrale Speicherung des Firmenwissens in einer Wissensdatenbank wächst auf lange

Sicht der Wert des KBE Systems und nicht das Wissens eines Einzelnen Ingenieurs. Außerdem erfordert die Implementierung Reflektion und Analyse der Ingenieursaktivitäten. Das Sammeln von Wissen verstärkt den interdisziplinären Austausch von Informationen und macht die Einrichtung von Kollaborationsnetzwerken notwendig.

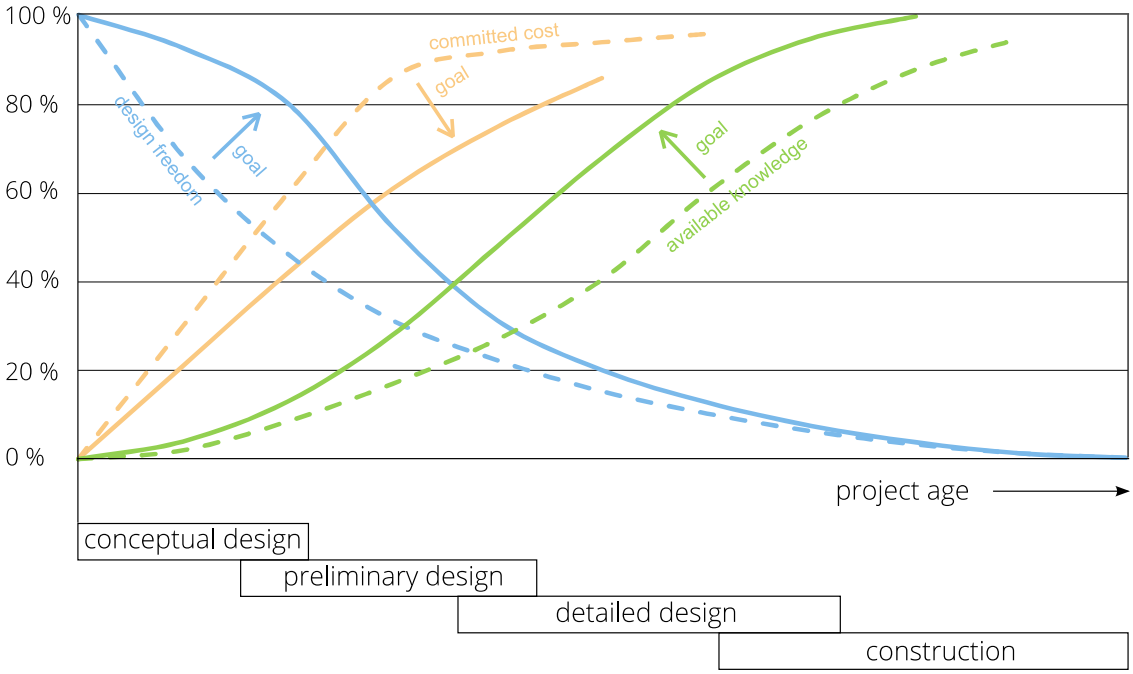


Abbildung 5.6: Einfluss der KBE Anwendung auf die Produktplanung (Verhagen et al. 2012)

Ein weiterer Vorteil ist die Einbeziehung von Erhaltungs-, Herstellungs- und Sicherheitsaspekten in frühen Entwurfsphasen. Bereits in dieser Entwurfsphase werden Festlegungen getroffen, die von großer Tragweite für den gesamten Lebenszyklus eines Produkts sind. Mit Hilfe eines KBE Systems können Entscheidungen mit negativen Konsequenzen auf die spätere Erhaltbarkeit, Herstellbarkeit, Sicherheit und die weiteren Lebenszykluskosten ausgeschlossen werden. Abbildung 5.6 zeigt die Auswirkungen der KBE Anwendung hinsichtlich Planungskosten, verfügbaren Wissen und Entwurfsfreiheit über den Projektzeitraum im Vergleich zu traditionellem CAD.

Trotz der genannten Vorteile, existieren ebenso Gründe Knowledge-based Engineering nicht anzuwenden. Ein großer Nachteil ist der zeitaufwendige Implementierungsprozess. Abbildung 5.7 zeigt die praktische Entwicklungszeit eines KBE Systems im Vergleich zu traditionellen CAD Designzyklen auf. Ein weiterer Grund ist, dass ein KBE System von Natur aus ein in sich geschlossenes System darstellt, ähnlich einer black-box, was zu einer

falschen Sicherheit beim Anwender führen könnte. Zusätzlich fehlt eine Quantifizierung der Zeit- und Kostenersparnisse durch die Anwendung von KBE.

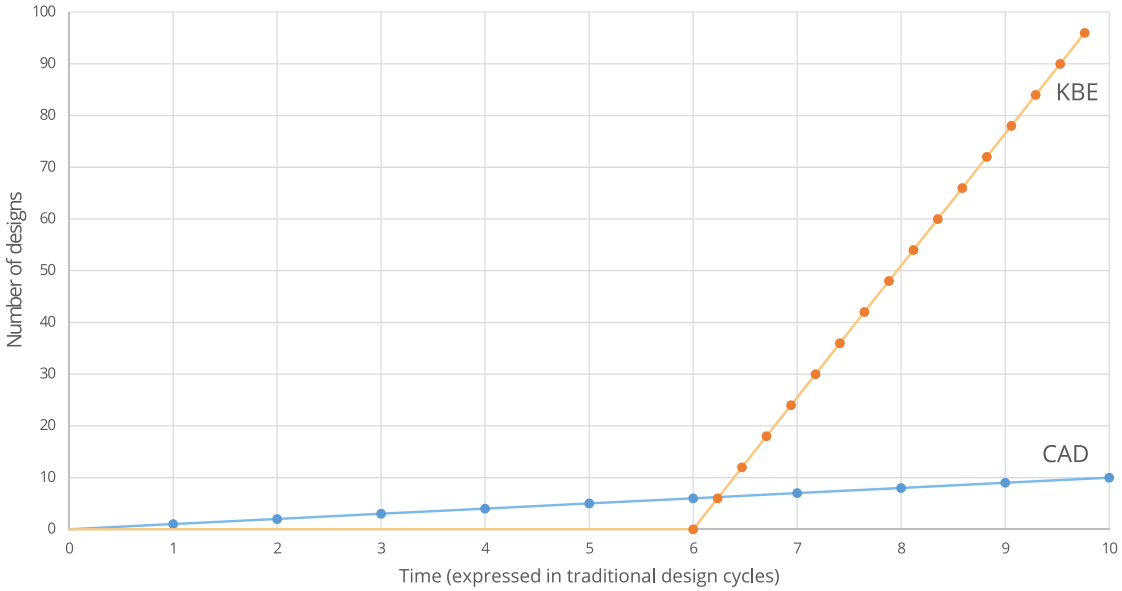


Abbildung 5.7: KBE Entwicklungszyklen (Verhagen et al. 2012)

5.7 Erfolgreiche Anwendung von KBE in anderen Branchen

In (Cooper et al. 1999) und (Chapman 2014) werden beeindruckende Erfolge durch die Implementierung von KBE aufgeführt:

- British Aerospace war in der Lage 60 Entwürfe auszuarbeiten, während eine Partnerfirma in der gleichen Zeit mit moderner CAD Technologie lediglich einen Entwurf generieren konnte.
- Jaguar konnte eine ursprünglich, unter Einbeziehung externer Fachkräfte, vier wöchig dauernde Machbarkeitsstudie in einen interaktiven Prozess umwandeln, der nun durch die eigenen Ingenieure in wenigen Minuten bewerkstelligt werden kann.
- Caradon Everest konnte die Freigabe von Fertigungsdaten aus kundenspezifischen Anforderungen von einem vier-wöchigen Prozess auf einen eintägigen verkürzen.

5.8 Kommerzielle KBE Softwaresysteme

5.8.1 Entwicklung

Um die Jahrtausendwende haben führende Entwickler von PLM (Product Lifecycle Management) Softwaresystemen das Potential von KBE erkannt und nach und nach ihre Softwareprodukte mit KBE Systemen aufgerüstet. (La Rocca 2012) gibt einen kurzen, aber umfassenden Überblick:

- 1999 führte PTC das Behavioral Modelling Toolkit für Pro/ENGINEER 2000i ein, welches es erlaubte, Regeln zu erfassen und damit die CAD Engine zu steuern.
- 2001 akquirierte UGS die KBE Programmiersprache Intent! von Heide Corporation und entwickelte Knowledge Fusion (2007 von Siemens PLM Software aufgekauft).
- 2002 wurde KTI und das zugehörige Softwareprodukt ICAD von Dassault Systemes akquiriert. DS stoppte ICAD und nutzte die KTI Expertise um das KBE AddIn KnowledgeWare für CATIA V zu entwickeln.
- 2005 kauft Autodesk die Engineering Intent Corporation auf und integrierte deren KBE System in Autodesk Inventor, namens Autodesk Intent
- Nach der Akquise von Design Power in Jahr 2007, integrierte Bentley das KBE System Design++ in Microstation.

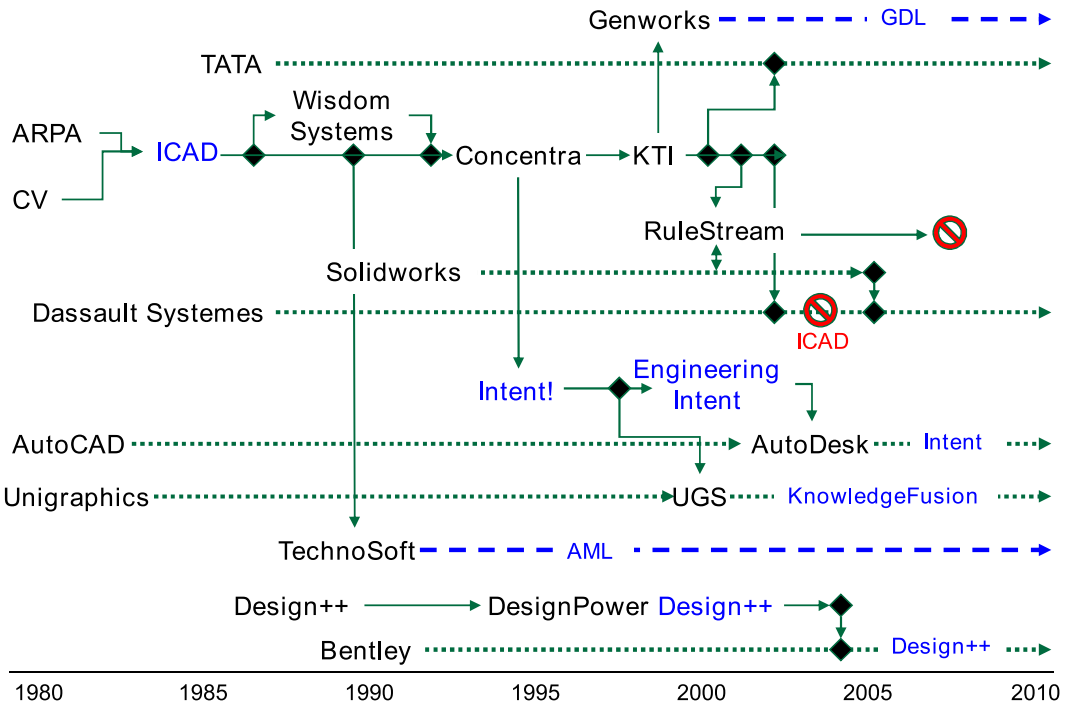


Abbildung 5.8: Entwicklung kommerzieller KBE Systeme (McGoey 2011)

5.8.2 Intelligent Computer-Aided Design

Intelligent Computer-Aided Design (ICAD) war das erste auf dem Markt verfügbare kommerzielle KBE System. Das 1984 von ICAD Inc. entwickelte System erlaubte es den Nutzer zum ersten Mal, Konstruktionswissen mit Hilfe einer semantischen Repräsentation abzubilden und wiederzuverwenden. Das Projekt ICAD entstand durch eine Zusammenarbeit von Larry Rosenfeld, von Computervision heute PTC, und Pat O'Keefe, einem Forscher im Bereich der Künstlichen Intelligenz am Massachusetts Institute of Technology (MIT). Um die Software kommerziell zu vertreiben wurde die ICAD Inc. gegründet. Der Einstieg von ICAD in den CAD Software Markt kann aus heutiger Sicht als Geburtsstunde von KBE betrachtet werden.

Die in ICAD implementierten KBE Systeme, erreichten anfangs aufgrund herausragender Resultate viel Aufmerksamkeit. Erstmals konnten Anwender selbst die Funktionalitäten eines CAD-Systems erweitern (End-User Computing (EUC)). Die Ingenieure konnten das identifizierte Expertenwissen selbst in einem KBE System implementieren und anschließend verwenden. Ausgehend von Anwendungen im Maschinenbau, folgten weitere in anderen Branchen. Um die Jahrtausendwende nutzten Boeing und Airbus ICAD intensiv zur Entwicklung verschiedener Komponenten. Im November 2002 wurde Knowledge Technologies International (KTI), die zu diesem Zeitpunkt für die Entwicklung und den Vertrieb von ICAD verantwortliche Firma, von Dassault Systemes übernommen (Dassault Systemes 2002). Kurze Zeit später wurde die Entwicklung von ICAD eingestellt, einige Jahre später auch der Vertrieb und Support des KBE Systems. Dassault Systemes nutzte die Expertise hinsichtlich der Entwicklung von KBE Systemen von KTI und veröffentlichte für ihre CAD Anwendung CATIA das KBE System Knowledgeware. Heute ist Genworks GDL das technisch und funktional nächste System zu ICAD.

ICAD stellt die deklarative Programmiersprache IDL (Intelligent Declarative Language) zur Verfügung. Abbildung 5.9 zeigt die Implementierung der Klasse *ConventionalAircraft* mit IDL. Das *defpart Makro* ist dabei das grundlegende Mittel, um das objekt-orientierte Programmierparadigma in ICAD umzusetzen. Damit lassen sich Klassen, Superklassen, Objekte und Vererbungsbeziehungen, Aggregationen und Assoziationen abbilden. Die auf einer UML-basierenden Wissensrepräsentation gegründete automatisierte Generierung von ICAD Code wurde in MOKA untersucht (Stokes 2001).

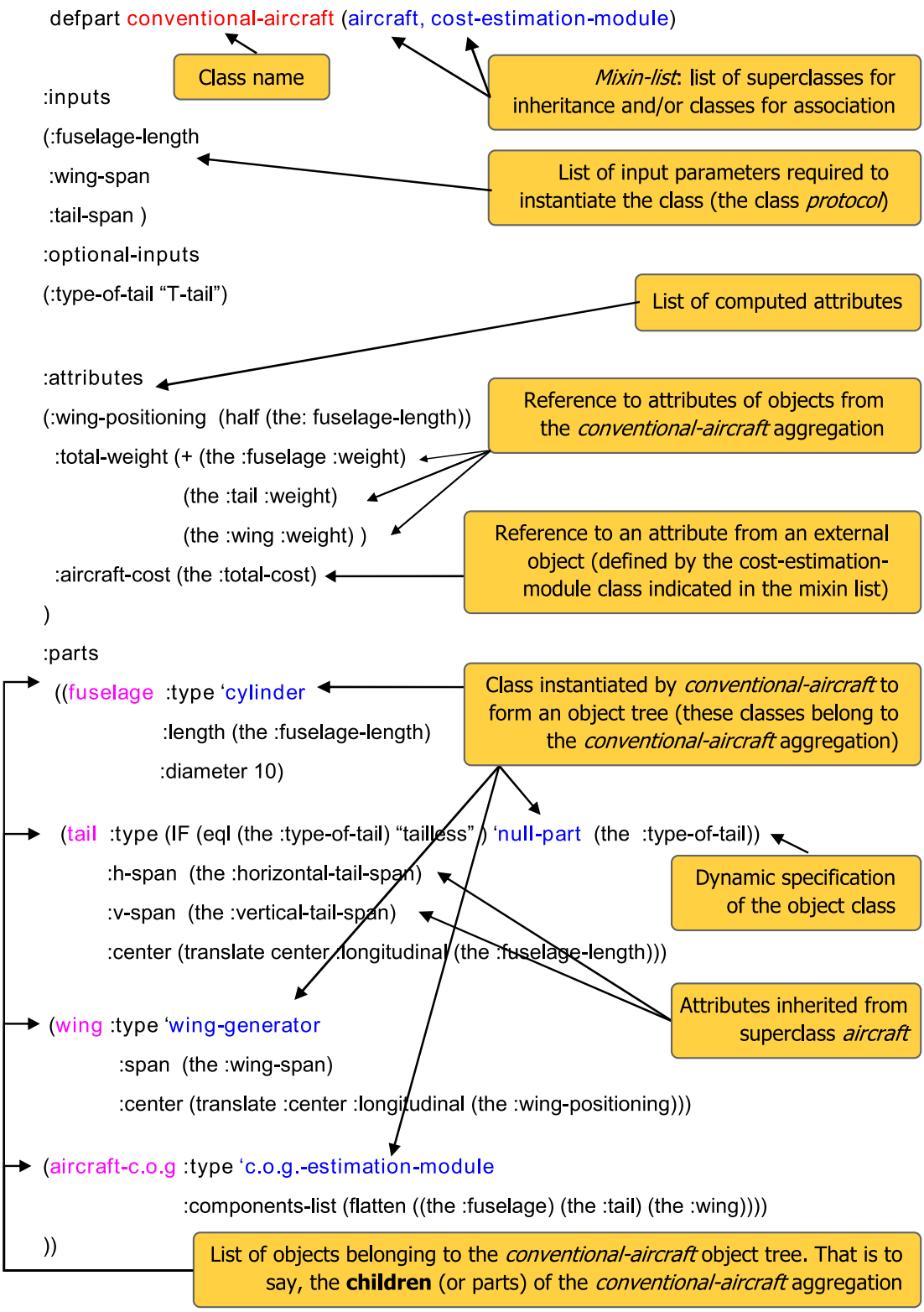


Abbildung 5.9: ICAD Code am Beispiel einer defpart Klasse nach (La Rocca 2011)

5.8.3 Siemens Knowledge Fusion

Siemens Knowledge Fusion ist das KBE Modul innerhalb des parametrischen CAD-Systems Siemens NX. Das Modul erweitert den Funktionsumfang von Siemens NX um Merkmale des Knowledge-based Engineering.

Für die Entwicklung einer KBE Anwendung in Knowledge Fusion (KF) ist eine Autoren-Lizenz nötig. Zur Ausführung einer KBE Anwendung in Siemens NX ist eine sogenannte Runtime-Lizenz notwendig. Da für die Bearbeitung dieser Arbeit keine der notwendigen Lizenzen vorlag, kann die Bewertung von Knowledge Fusion hinsichtlich der Eignung für die Anwendung im Infrastrukturbau nur anhand der verfügbaren Literatur erfolgen. Zudem ist die Anzahl frei verfügbarer Literatur sehr gering. Sie beschränkt sich auf ein veraltetes Handbuch (Unigraphics 2001), Marketing Broschüren (Siemens PLM Software 2008, 2010) und eine Präsentation (Kok 2005).

Die umfangreichsten Informationen zu Aufbau und Funktionsweise von Knowledge Fusion sind in (Liese 2003) zu finden:

Knowledge Fusion verwendet die objekt-orientierte und deklarative KBE Programmiersprache *Intent!*. Um Objekten Regeln zuweisen zu können ist eine Übernahme der Objekte aus dem CAD-Modell in die KF Umgebung mit Hilfe einer *Adoption* notwendig. Nach der *Adoption* steht das Objekt im Knowledge Fusion Navigator zur weiteren Bearbeitung zur Verfügung. Eine andere Möglichkeit ist die Instanziierung eines Objekts direkt in KF. Nach der Definition in KF wird das korrelierende Objekt im CAD-Modell automatisiert erzeugt. Über den Navigator können die Regeln der Objekte geändert oder geometrische Abhängigkeiten definiert werden. Diese Funktionen von Knowledge Fusion sind für Anwender ohne Programmierkenntnisse gedacht.

Um eigene Klassen, Methoden, Funktionen oder Attribute zu programmieren, steht eine Integrated Development Environment genannt *NX Knowledge Fusion ICE* für Knowledge Fusion (Kok 2005) zur Verfügung. Zudem existieren in KF weitere Module (Siemens PLM Software 2008). Mit dem Modul *Knowledge Feature* können externe Programme assoziativ in KF eingebunden werden. Mit *Knowledge Pipeline* ist es möglich auf Datenbanken zuzugreifen. Durch sogenannte *Checker* können Modelle hinsichtlich definierter Grenzwerte überprüft werden.

5.8.4 Autodesk Inventor Automation Professional

Sucht man nach Schlagwörtern zum Thema regelbasierte Entwurfsautomatisierung für Autodesk Inventor, so findet man schnell die Begriffe iLogic, Inventor Automation Professional, Intent und Engineer-To-Order (kurz ETO) vor. Dabei werden Intent und Inventor Automation Professional synonym verwendet. Intent ist nicht mit der KBE Programmiersprache Intent! zu verwechseln. ETO ist ein neuere Weiterentwicklung, um den Entwicklungsprozess vom Eingang einer neuen Kundenspezifikation bis hin zum fertiggestellten Produkt besser unterstützen zu können. Intent und iLogic sind Erweiterungen von Inventor zur Unterstützung einer regelbasierten Konstruktion. Beide wrappen die Inventor API (Application Programming Interface), um die angeboten Funktionen umzusetzen.

Intent (oder Inventor Automation Professional) ist keine Programmiersprache sondern viel mehr eine Rule Engine oder ein Regel Editor für Inventor in der Regeln auf Basis einer Syntax deklarativ definiert werden können. Da die Regeln deklarativ programmiert sind, kümmert sich Intent unter der Haube selbstständig um die Ausführung der Regeln. Immer dann, wenn sich in Relation stehende Größen wie Parameter oder Variablen ändern, folgt eine Neuberechnung der betroffenen Regeln. Dies ist neben dem größeren Funktionsumfang der bemerkenswerteste Unterschied zum prozeduralen iLogic.

Anders als iLogic ist Intent außerdem nicht im Funktionsumfang limitiert und bietet alle Funktionen der Inventor API an. Es können außerdem externe .dll Klassenbibliotheken (Dynamik Link Library) oder .exe Programme aus Intent heraus direkt aufgerufen werden. Weiter ist Intent generativer Natur, das heißt, Intent kann die Regeln nicht nur auf bestehende Geometrie anwenden, sondern auch selbst neue Geometrie erzeugen. Dies bildet damit ein wesentliches Merkmal des Generic Automatic Instantiation von (Amadori 2012) ab.

Eigene Benutzeroberflächen können sowohl in iLogic als auch Intent erstellt werden. Im Gegensatz zu Intent kann mit Hilfe von in iLogic definierten Regeln keine Geometrie erzeugen, sondern nur existierende Geometrie und Parameter regelbasiert gesteuert werden. Um diese erweiterte Parametrik in Regeln umzusetzen stehen dem Anwender Schleifen, Wenn-Dann Beziehungen, sowie string und boolean Parameter zur Verfügung. Programmiersprache in iLogic sind die prozeduralen Sprachen VBA / VB.Net.

Abbildung 5.10 und Abbildung 5.11 zeigen die Unterschiede zwischen iLogic und Intent anhand der Definition eines eigenen *Offset Constraint* auf. Es wird deutlich, dass die prozedurale Programmierung in iLogic zur Umsetzung derselben Aufgabe wesentlich mehr Code benötigt als die deklarative Schreibweise in Intent.

```
Public Sub MyMateConstraintOfWorkPlanes()
    Dim oAsmCompDef     As AssemblyComponentDefinition
    Dim oOcc1           As ComponentOccurrence
    Dim oOcc2           As ComponentOccurrence
    Dim oPartPlane1     As WorkPlane
    Dim oPartPlane2     As WorkPlane
    Dim oAsmPlane1      As WorkPlaneProxy
    Dim oAsmPlane2      As WorkPlaneProxy
    Dim oMateConstraint As MateConstraint

    Set oAsmCompDef = ThisApplication.ActiveDocument.ComponentDefinition
    Set oOcc1 = oAsmCompDef.Occurrences.Item(1)             'Our first component
    Set oOcc2 = oAsmCompDef.Occurrences.Item(2)             'Our second component
    Set oPartPlane1 = oOcc1.Definition.WorkPlanes.Item(2)   'YZ Workplane reference
    Set oPartPlane2 = oOcc2.Definition.WorkPlanes.Item(2)   'YZ Workplane reference
    Call oOcc1.CreateGeometryProxy(oPartPlane1, oAsmPlane1) 'Create proxies for geometry
    Call oOcc2.CreateGeometryProxy(oPartPlane2, oAsmPlane2) 'Create proxies for geometry
    Set oMateConstraint = oAsmCompDef.Constraints.AddMateConstraint(oAsmPlane1, oAsmPlane2, 0)
                                                            'Create the constraint
    oMateConstraint.Offset = 45                             'Set the offset
End Sub
```

Abbildung 5.10: Beispiel iLogic (Williams 2009)

```
Parameter Rule MyOffset as number = 45

Child MyMateConstraint as : IvMateConstraint
    Part1     = SideChannel_1     'Our first component
    Entity1   = "YZ Plane"        'Our first component's entity to mate
    Part2     = SideChannel_2     'Our second component
    Entity2   = "YZ Plane"        'Our second component's entity to mate
    Offset    = MyOffset          'Set the offset
End Child
```

Abbildung 5.11: Beispiel Intent (Williams 2009)

5.8.5 Bewertung

Ein Ziel dieser Arbeit ist die Bewertung der untersuchten kommerziellen KBE Softwaresysteme hinsichtlich der Anwendbarkeit für den Infrastrukturbau. In diesem Abschnitt werden einige Gedanken zu iLogic/Intent und Knowledge Fusion diskutiert.

Da Intent und iLogic lediglich die Funktionen der Inventor API wrappen, bieten sie keinen Mehrwert an Funktionalität an. Hervorzuheben ist selbstverständlich die deklarative Beschreibung von Regeln in Intent. Da Intent die Ansteuerung externer Bibliotheken erlaubt, wäre eine Erweiterung um eigene, für die Modellierung für den Infrastrukturbau notwendigen Funktionen denkbar, was aber umständlich ist.

Im Vergleich zu Intent bietet Knowledge Fusion einen ähnlichen Funktionsumfang. So können Anwender ohne Programmierkenntnisse standardmäßige einfache, deklarative Regeln definieren. Für die Programmierung von Erweiterungen steht ein Code Editor zur Verfügung. Damit ist auch für Knowledge Fusion die Anbindung externer Bibliotheken möglich.

Um die beiden KBE Systeme um Funktionen für den Infrastrukturbau zu erweitern, sind also bei beiden Systemen Programmierkenntnisse nötig. Viele Regeln in der Brücken- vor allem aber in der Tunnelplanung interagieren mit den Geländemodellen. Sie sind mit den hier diskutierten proprietären KBE Systemen nur schwer abbildbar. Um weitere branchenspezifische Stabwerks-, FEM-, Trassenplanungs- oder Kalkulationsprogramme assoziativ mit diesen KBE Systemen zu nutzen, ist großer Programmieraufwand erforderlich. Damit kann ein Paradigma von KBE nicht mehr erfüllt werden: Die deklarative Abbildung von Wissen. Das Wissen ist damit, beispielsweise im Umgang mit Geländemodellen, wie bisher in prozeduralem Code enthalten. Es wäre also keine zentrale Wissensrepräsentation mit Trennung von Regeln und Anwendungslogik mehr möglich.

Auch die Entwicklungen im Bereich BIM sollten an dieser Stelle berücksichtigt werden. Da sowohl Siemens NX als auch Autodesk Inventor nicht speziell für die Anwendung im Bauwesen entwickelt werden, werden sie auch zukünftig keine BIM spezifischen Schnittstellen wie IFC anbieten, was zusätzlichen Programmieraufwand bedeutet.

6 Entwicklung eines KBE Prototyps für den Brückenentwurf

In diesem Kapitel wird die prototypische Umsetzung eines einfachen KBE Systems für den Brückenentwurf erläutert. Zunächst folgt eine allgemeine Einführung in den Brückenentwurf, dabei werden die typischen Regelwerke für den Entwurf von Brücken vorgestellt. Anschließend wird die prototypische Implementierung eines KBE Systems mit Autodesk Revit (Autodesk 2014b) und Dynamo (Dynamo 2014) beschrieben. Autodesk Revit und Dynamo wurden als Umgebung gewählt, da beide Programme dem Entwickler bereits mächtige Werkzeuge an die Hand geben und somit eine gute Grundlage für eine prototypische Umsetzung darstellen. Nach der Vorstellung existierender Tools sowie der verwendeten Programme folgt die Beschreibung der Umsetzung einiger Entwurfsregeln.

6.1 Brückenentwurf

Ziel des Brückenentwurfs ist es, durch Untersuchung verschiedener Varianten einen in technischer, ökologischer, gestalterischer und wirtschaftlicher Hinsicht optimalen Kompromiss der Konstruktion eines geplanten Brückenbauwerks zu finden (siehe Abbildung 6.1). Er dient als Grundlage für die Ermittlung der Baukosten, der Ausschreibung und der Bauausführung. Der Brückenentwurf ist ein iterativer Prozess und stark abhängig von den äußeren Bedingungen wie beispielsweise Funktion und Lage des Bauwerks.

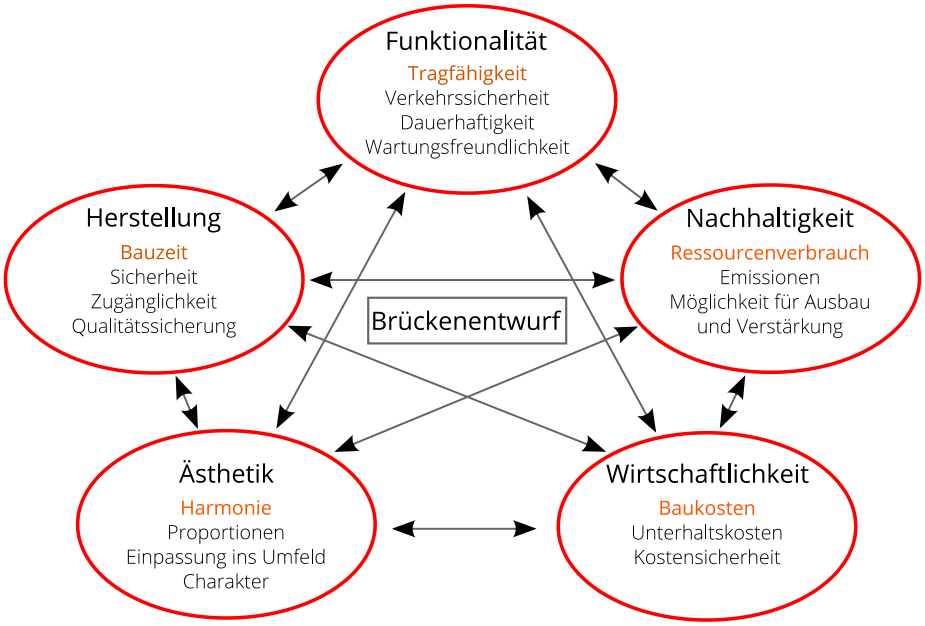


Abbildung 6.1: Entwurfskriterien Brückenentwurf (Fischer 2013)

Das Brückenbauwerk wird in der Entwurfsplanung durch die wesentlichen Entwurfselemente wie Längstragwerk, Quertragwerk, Unterbauten, Gründung, Widerlager, Lagerung, Bauverfahren, Baustoffe und Brückenausrüstung sowie die jeweils geltenden Regelwerke und Richtlinien eindeutig festgelegt und beschrieben.

Der Brückenentwurf ist zudem Beurteilungsgrundlage für die politischen Entscheidungsträger. Hier kann KBE einen wesentlichen Beitrag zu mehr Planungssicherheit leisten. Heutzutage sind öffentliche Bauprojekte regelmäßig von Budgetüberschreitungen im unteren bis mittleren zweistelligen Prozentbereich geprägt. Im Brückenbau entstammen die öffentlich kommunizierten Baukosten dem Brückenentwurf. Durch die regelbasierte Konstruktion und die Verknüpfung mit Kalkulationsprogrammen, können bereits in sehr frühen Phasen des Entwurfs Aussagen zu den Kosten des Bauwerks getroffen werden.

Abbildung 6.2 zeigt beispielhafte Szenarien (Nr. 1-6) für einen Brückenentwurf. Eine Planungsaufgabe während des Vorentwurfs könnte die Evaluierung möglicher Trassenvarianten zur Querung des Tals beinhalten. Mit Hilfe eines KBE Systems können hier mehrere Brückenvarianten mit jeweils unterschiedlichen Randbedingungen in kurzer Zeit konstruiert und anhand weiterer Kriterien wie den Baukosten bewertet werden.

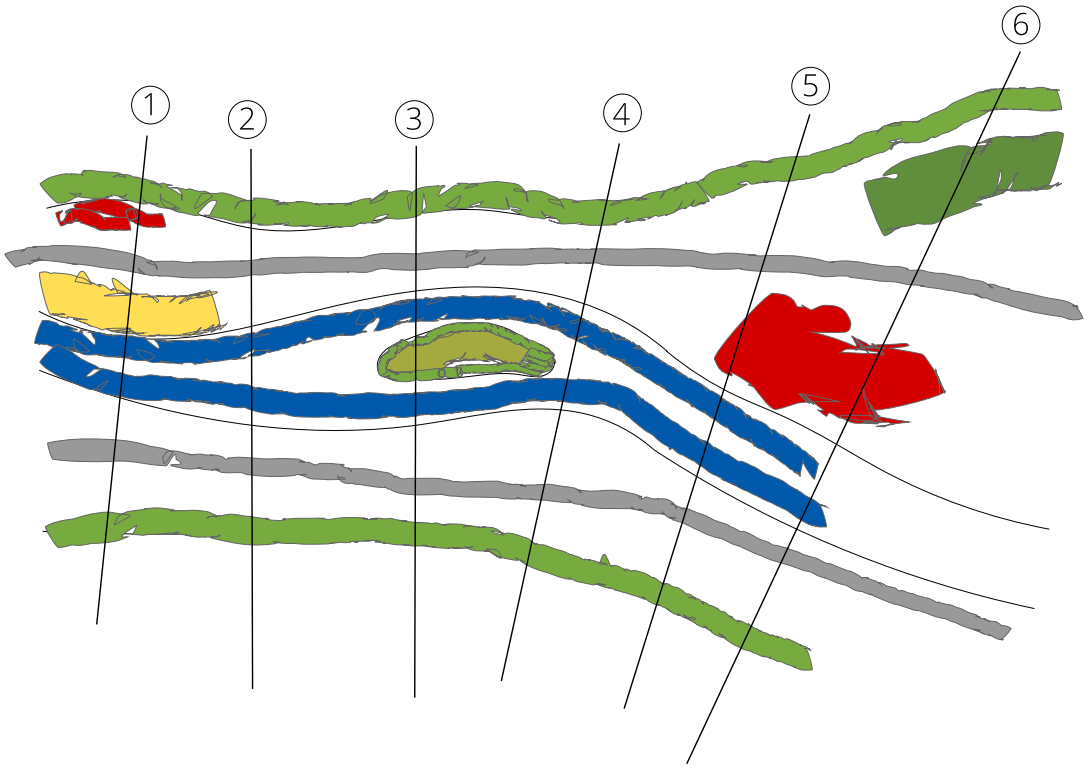


Abbildung 6.2: Entwurfsszenario

6.1.1 Der Entwurfsprozess einer Brücke

In (Standfuß 1995) wird der Ablauf des Entwurfsprozesses einer Brücke genauer beschrieben:

Die ingenieursmäßige Gestaltung eines Brückenbauwerks beginnt im ersten Schritt mit der überschlägigen Bestimmung seiner Gesamtlänge, der Anzahl der Felder und der Stützweiten in Abhängigkeit von den Baugrundverhältnissen, der Geländeform und der Lage der Fahrbahn über Gelände selbst. Es folgt dann die Wahl der Bauweise der Überbauten (Balken, Bogen, Fachwerk, Rahmen, seilverspanntes Tragwerk), der Bauart der Überbauten (Stahlbeton, Spannbeton, Stahlverbund, Stahl), der überschlägigen Bestimmung der Konstruktionshöhe der Überbauten sowie der Bestimmung der Grundformen und Abmessungen der Unterbauten.

Da es meist für die gestellte Aufgabe mehrere Lösungen gibt, ist die Untersuchung verschiedener Varianten Voraussetzung für die Entscheidung, welche Lösung unter Abwägung aller Vor- und Nachteile die Beste ist und weiter ausgearbeitet werden soll. Liegt die ausgewählte Lösung fest, erfolgt im zweiten Schritt die Festlegung der Abmessung der wesentlichen Bauwerksteile aufgrund einer statischen Vorberechnung und Bemessung.

Im dritten Schritt ist dann zu entscheiden, welche äußere Form und Gliederung im Detail die einzelnen Bauwerksteile (Überbau, Pfeiler, Widerlager, Gesimse, Geländer) in Abhängigkeit vom Typ und der Größe des Bauwerks und der jeweiligen örtlichen Situation erhalten sollen. Mit welchem Baustoff (Beton, Kunst- oder Naturstein), mit welchen Strukturen der Oberfläche (Schalungsarten und Schalungsformen) und Farben (Verkleidungsmaterial für die Unterbauten, Geländer, Brüstungen) das Gestaltungsziel zu erreichen ist.

6.1.2 Tragsysteme in Längsrichtung aus Stahl- / Spannbeton

Zur Vereinfachung fokussiert diese Arbeit auf die Implementierung eines Prototyps für den Entwurf einer Brücke aus Stahl-/ Spannbeton. Weitere Brückentypen aus Stahl oder in Stahlverbundbauweise werden nicht berücksichtigt. Im Massivbrückenbau werden im Groben drei Gruppen von Tragsystemen, nämlich Balkentragwerke, Rahmentragwerke und Bogentragwerk (siehe Abbildung 6.3) unterschieden. Im Folgenden werden diese nach (Holst 2013) vertiefter charakterisiert:

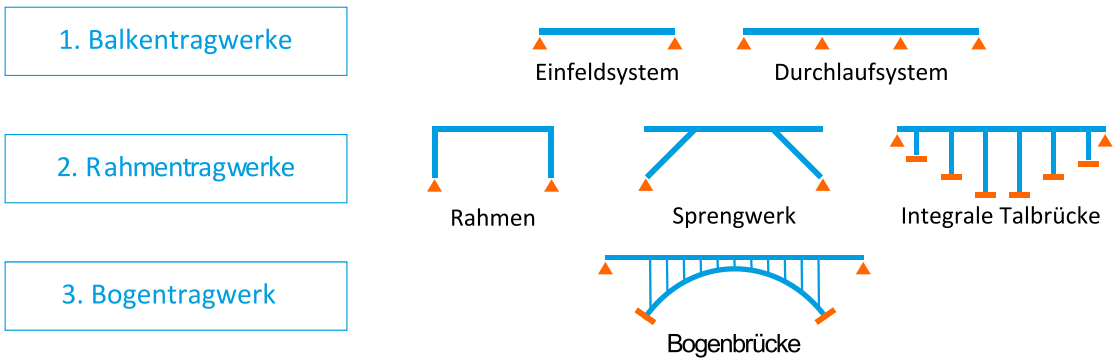


Abbildung 6.3: Tragsysteme in Längsrichtung nach (Fischer 2013)

Balkentragwerke

Eine Balkenbrücke liegt dann vor, wenn der Überbau unabhängig vom Überbautyp in Form eines Balkens ausgebildet ist. Das äußere Kennzeichen der Balkenbrücke liegt in der sichtbaren Trennung von Über- und Unterbau durch die Lagerebene. Balkenbrücken werden aus wirtschaftlichen Gründen in der Herstellung meist mit konstanter Konstruktionshöhe entlang der Brückenachse hergestellt, sie sind „parallelgurtig“. Die Stützenstellung beeinflusst beim Durchlaufträger wesentlich den Momentenverlauf und damit die Beanspruchungsgrößen. Im Großbrückenbau findet man daher vorwiegend gleichmäßig verteilte Stützweiten. Bei Dreifeldträgern strebt man ein Stützweitenverhältnis von 1:1,35:1 an, um Stützmoment und mittleres Feldmoment gleich groß zu halten.

Rahmentragwerke

Bei Rahmentragwerken ist der Überbau mit dem Unterbau monolithisch verbunden, es ist also keine Trennung der Bauteile sichtbar. Durch die biegesteife Ausbildung der Rahmenecken wird das Feldmoment reduziert, was deutlich geringere Konstruktionshöhen zur Folge hat. Um die unterschiedlichen Konstruktionshöhen im Feld und an der Stütze auszugleichen, wird der Überbau meist „gevoutet“ ausgebildet. Rahmenbrücken können sowohl als Einfeld- als auch als Mehrfeldsysteme ausgeführt werden. Im Autobahnbau hat sich aufgrund der klaren Formgebung und der größeren Freiheit in der Widerlagerkonstruktion eine Zweifeldlösung etabliert. Integrale Brücken, beziehungsweise fugenlose Konstruktionen, bieten durch den Wegfall von Lager- und Übergangskonstruktionen eine bessere Wartungsfreundlichkeit als die Ausführung mit Fugen.

Bogentragwerk

Das Bogentragwerk stellt die älteste Konstruktionsform von Stahlbetontragwerken dar. Der Hauptträger wird als Bogen ausgebildet, die Lasten des Überbaus werden über eine Aufständerung in den Bogen eingetragen und durch Druckkräfte in die Kämpfer weitergeleitet. Mit Hilfe eines Bogentragwerks war es erstmals möglich tiefe Täler zu überwinden, wobei eine aufwendige Lehrgerüstkonstruktion notwendig wurde. Heutzutage werden Bogentragwerke meist im Freivorbauverfahren hergestellt.

Kombinationen

Kombinationen aus Balkentragwerk und Rahmen- oder Bogentragwerk sind ebenfalls möglich. So ist die Überwindung von flachen Vorlandabschnitten per Balkentragwerk und die Ausführung der Flussüberquerung als Rahmentragwerk eine häufig verwendete Kombination.

Im Rahmen der Entwicklung eines Prototyps, fokussiert diese Arbeit auf die Umsetzung von Entwurfsregeln von Balkentragwerken.

6.1.3 Tragsysteme in Querrichtung

Für die Ausbildung des Tragsystems in Querrichtung existieren die drei Überbautypen Massivplatte, Plattenbalken und Hohlkasten. In diesem Abschnitt werden die wesentlichen Merkmale dieser Überbautypen kurz nach (Holst 2013) beschrieben:

Massivplatte

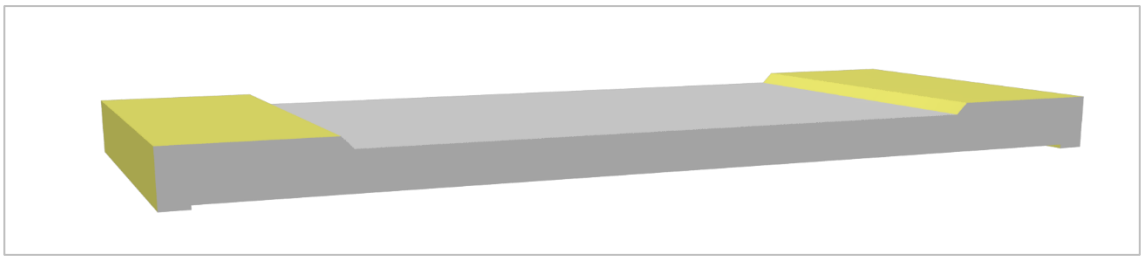


Abbildung 6.4: Massivplatte

Die Massivplatte (Abbildung 6.4) ist der bevorzugte Überbautyp für Brücken geringer Stützweite (bis ca. 18-20m), da sie den Vorteil der einfachen Herstellung und einer geschlossenen Untersicht besitzt. Die Lasten werden durch das Flächentragwerk der Platte in Brückenlängs- und -querrichtung abgetragen. Bei maximal ausgereizter Stützweite resultieren Konstruktionshöhen bis 0,95m. Es besteht die Möglichkeit die Tragfähigkeit der Platte durch Walzträger oder Vorspannung zu erhöhen.

Plattenbalken

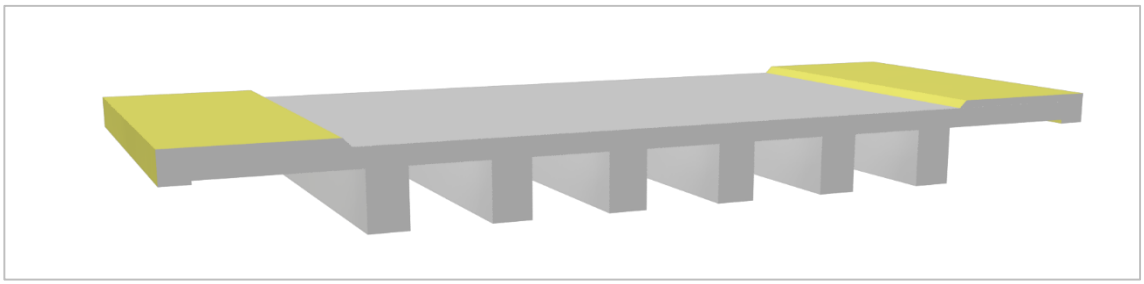


Abbildung 6.5: Mehrstegiger Plattenbalken

Der Plattenbalkenquerschnitt gliedert sich in die Hauptträger, die Fahrbahnplatte mit den Kragarmen und gegebenenfalls eine unter Druckplatte. Die Platte übernimmt die Verkehrslasten und verteilt sie auf die Hauptträger, welche dann die Lasten als Balken zu den

Lagern leiten. Häufigste Form ist der 2-stegige Plattenbalken (Abbildung 6.6), bei Querschnittsbereiten größer 12m kann ein dritter Hauptträger vorgesehen werden. Mehrstegige Plattenbalken (Abbildung 6.5) sind für die Herstellung des Überbaus in Fertigteilbauweise sinnvoll. Durch die Fertigteilbauweise wird der Eingriff der Baumaßnahme in den laufenden Verkehr stark reduziert.

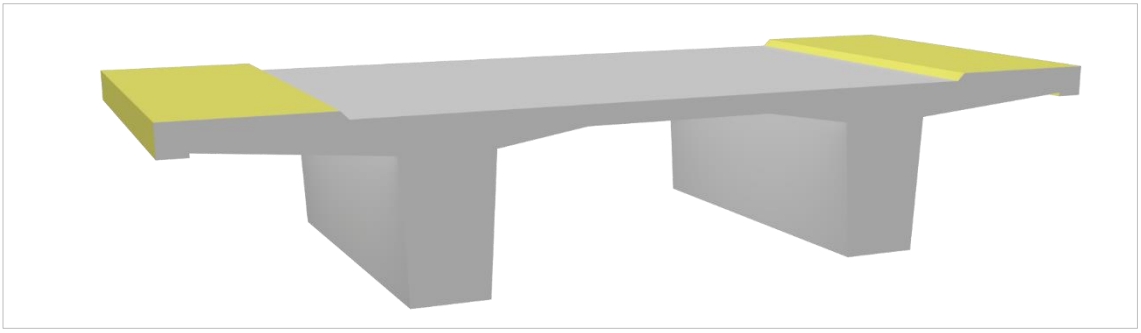


Abbildung 6.6: 2-stegiger Plattenbalken

Hohlkasten

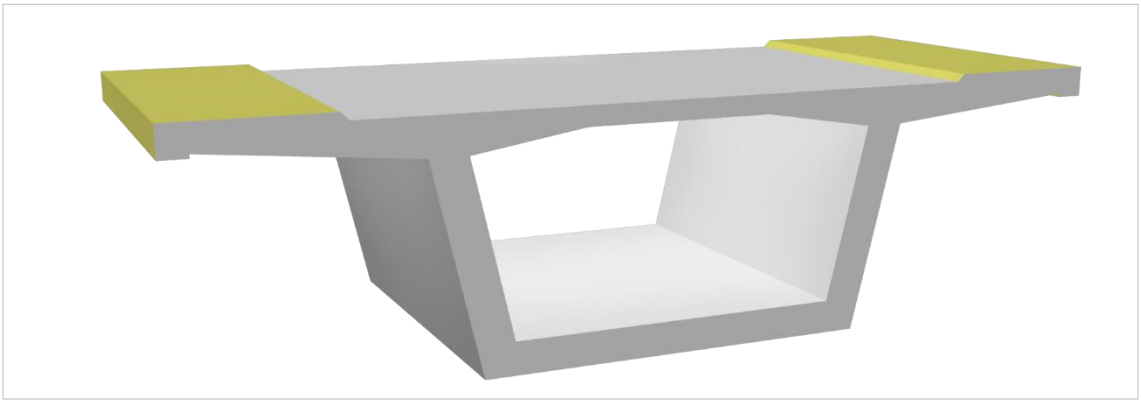


Abbildung 6.7: Hohlkasten

Der Hohlkasten (Abbildung 6.7) ist für Stützweiten von 40 bis 80m wirtschaftlich einsetzbar. Der Hohlkasten hat sich ursprünglich aus dem Plattenbalken entwickelt und wird vor allem wegen seines hohen Torsionstragvermögens verwendet. Dies bietet vor allem bei großen Querschnittsbreiten (ab 12m) und gekrümmten Achsen Vorteile. Außerdem ist bei großen Breiten eine Auflösung des Querschnitts in mehrere Einzelzellen sinnvoll. Aufgrund der Weiterentwicklung von Bauverfahren (bewegliche Rüstung, Taktschieben, Freivorbau) ist der Hohlkasten mittlerweile der bevorzugte Querschnittstyp für Großbrücken aus Spannbeton.

6.1.4 Richtzeichnungen und Richtlinien für Brücken- und Ingenieurbauten

Wie alle anderen Bauwerke, ist auch die Konstruktion von Brücken stark durch Normen und Richtlinien reglementiert. Diese Dokumente bilden damit eine wichtige Wissensquelle für ein KBE System im Brückenbau. 1972 wurden erstmals bundesweit einheitliche Richtlinien für den Brücken- und Ingenieurbau veröffentlicht. Das Bundesministerium für Verkehr und digitale Infrastruktur (BMVI) beschreibt auf ihrer Internetpräsenz die Funktionen der Richtzeichnungen wie folgt (BMVI 2014):

> *„Die Richtzeichnungen beschreiben die für die Planung, Kalkulation und Ausführung von Bauwerken und ihren Teilen geltenden Bedingungen, damit für wiederkehrende technische Aufgaben bewährte Lösungen vorgesehen werden können. Sie geben den jeweiligen Stand der Technik wieder und sind für den Bereich der Bundesfernstraßen im Rahmen von Bauverträgen grundsätzlich anzuwenden. Im Zuständigkeitsbereich der Länder, Kreise und Gemeinden ist ihre Anwendung empfohlen. Werden in begründeten Ausnahmefällen Abweichungen oder andere Konstruktionen erforderlich, so sind die Festlegungen in den Richtzeichnungen aber als Mindestanforderungen und als Maßstab für die erforderliche Qualität einzuhalten. Die Richtzeichnungen sind Teil der vom Bundesministerium für Verkehr, Bau und Stadtentwicklung (BMVBS) bzw. der Bundesanstalt für Straßenwesen (BASt) herausgegebenen Sammlung Brücken- und Ingenieurbau. Sie werden von der zuständigen BASt-Arbeitsgruppe bearbeitet, der neben Vertretern des BMVBS und der BASt auch Vertreterinnen und Vertreter der Straßenbauverwaltungen der Länder angehören. Die Fortschreibung erfolgt in der Regel einmal jährlich.“*

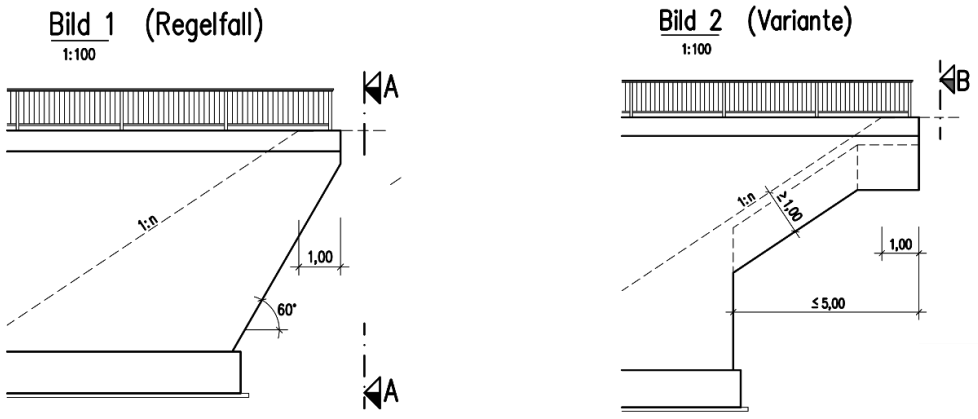


Abbildung 6.8: Richtzeichnung für die Ausbildung der Widerlagerflügel (BASt 2009)

6.2 Entscheidungsbäume für den Brückenentwurf

Um die vorhandenen Erfahrungen und Richtlinien des Brückenentwurfs als Regeln zu formalisieren und zur übersichtlichen Darstellung derselben, bieten sich Entscheidungsbäume an. Kann nach einer Reihe von Entscheidungen keine Regel mehr angewandt werden kann, ist es durch die Struktur des Entscheidungsbaums leicht möglich, schnell zu einem letzten validen Punkt zurückzukehren und einen anderen Zweig zu wählen (auch Vorwärtsverkettung genannt). Zur Erreichung einer Lösung des Problems sind unter Umständen mehrere Iteration notwendig. Weitere Informationen zu Entscheidungsbäumen sind in Kapitel 3.4 zu finden. Die gewonnen Entscheidungsbäume dienen als Grundlage für die Implementierung des KBE Prototypen. Abbildung 6.9 zeigt die Verknüpfung der einzelnen Entscheidungsbäume zu einem globalen Entscheidungsbaum für den Brückenentwurf.

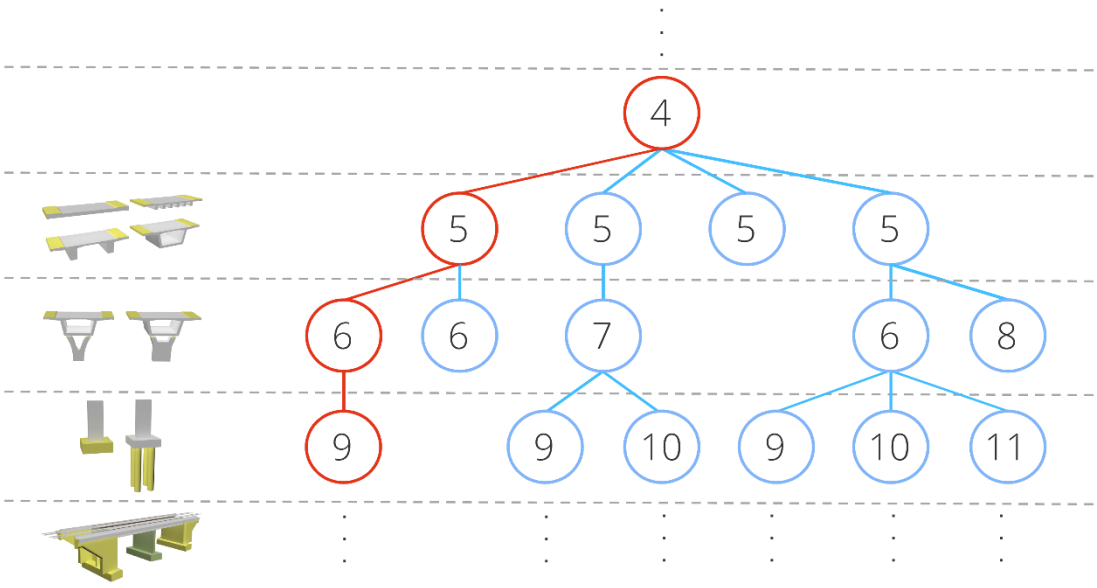


Abbildung 6.9: Globaler Entscheidungsbaum für den Brückenentwurf

Die folgenden Seiten zeigen jeweils Entscheidungsbäume für die einzelnen Brückenbauteile Überbau (Abbildung 6.10), Pfeiler (Abbildung 6.11), Gründung (Abbildung 6.12) und Widerlager (Abbildung 6.13). Die Bäume sind grundsätzlich gleich aufgebaut. Nach Definition der Eingangsparamater folgt die Regelkette, die bei erfolgreichem Ablauf zu einer Konstruktion führt. In blauer Farbe sind die jeweiligen Regeln markiert, grün gekennzeichnete Felder deuten auf eine Einflussnahme durch den Ingenieur hin.

6.2.1 Überbaukonstruktion

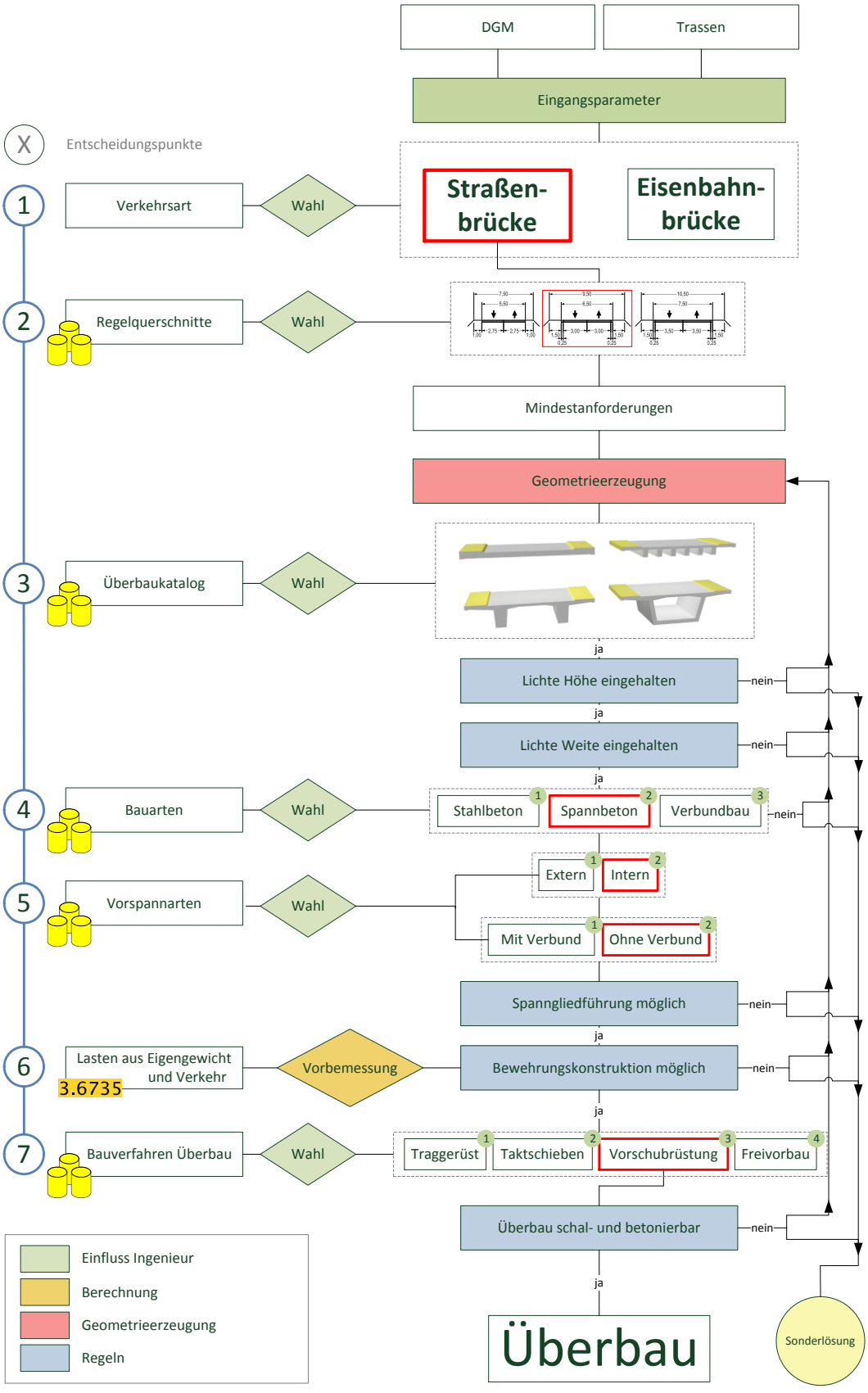


Abbildung 6.10: Entscheidungsbaum Überbaukonstruktion

6.2.2 Pfeilerkonstruktion

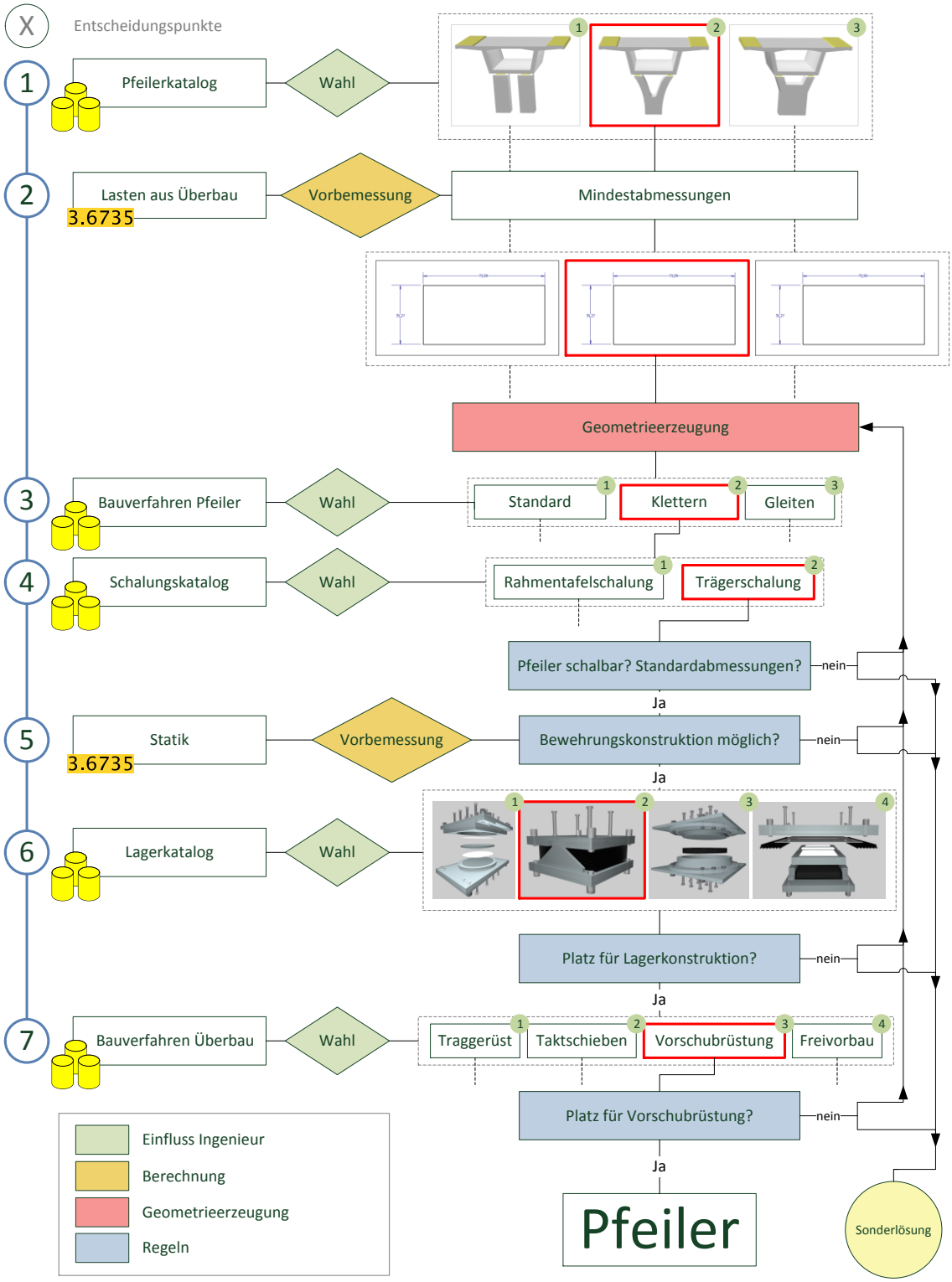


Abbildung 6.11: Entscheidungsbaum Pfeilerkonstruktion

6.2.3 Gründung

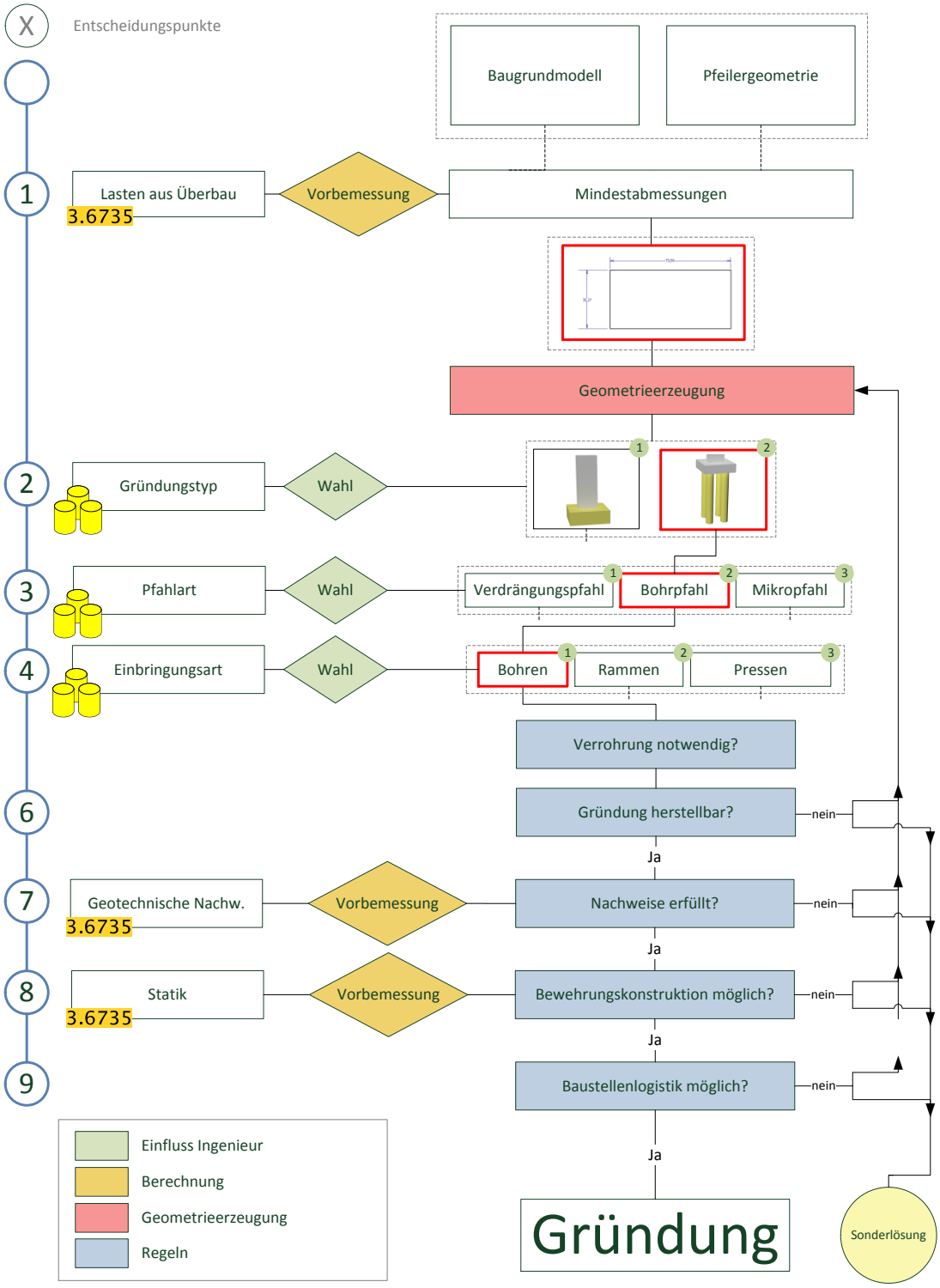


Abbildung 6.12: Entscheidungsbaum Gründungskonstruktion

6.2.4 Widerlager

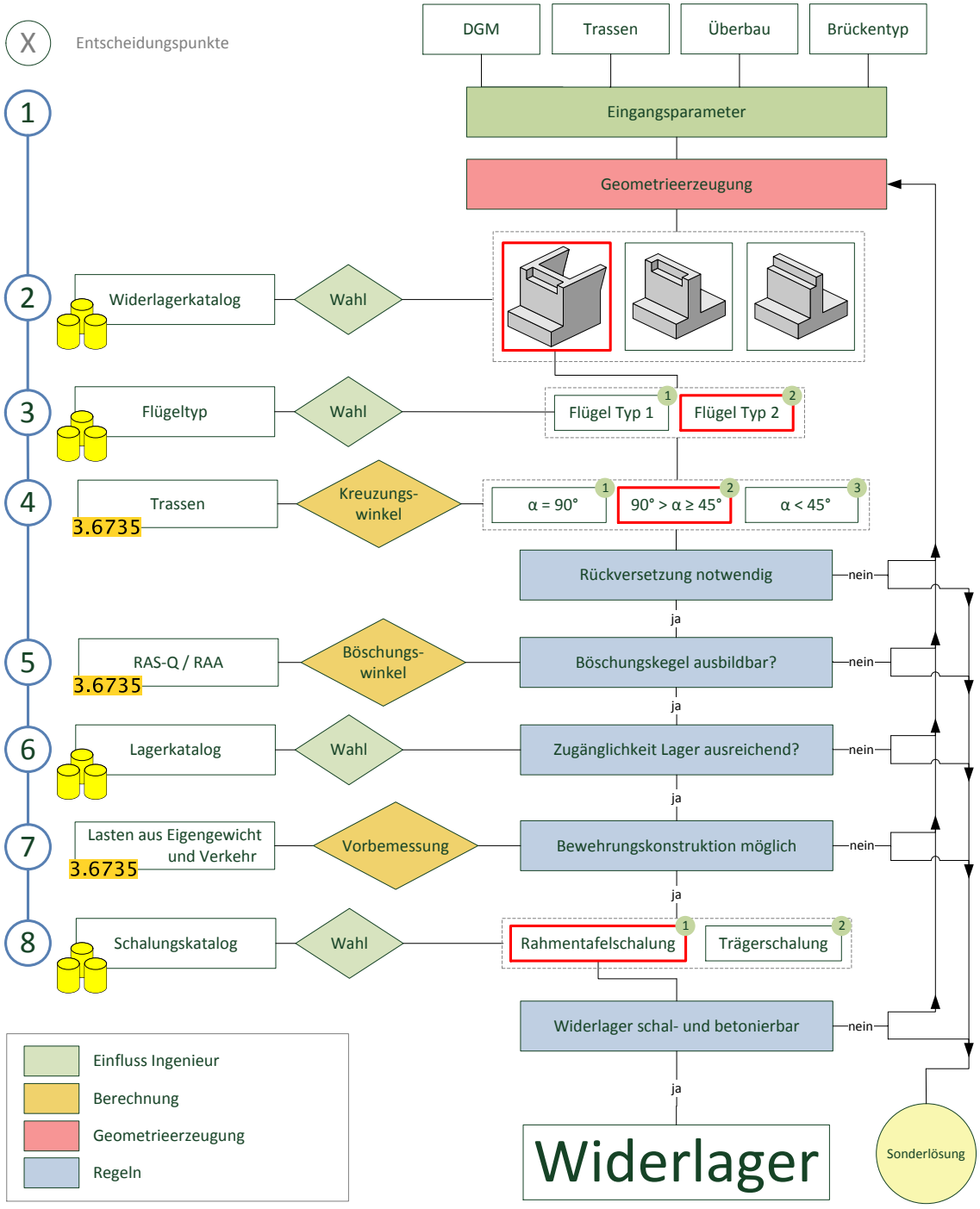


Abbildung 6.13: Entscheidungsbaum Widerlagerkonstruktion

6.3 Vorhandene Tools Parametrische Brückenmodellierung

Es existieren bereits einige Tools für die parametrische Modellierung von Brücken. So verfügt die Subscription App Civil Structures für Autodesk Revit 2014 (Autodesk 2014c) über einige Features zur Generierung von Brückenmodell in Revit und Civil3D. Infraworks360 (Autodesk 2014a) ist eine Autodesk-Software zur Entwurfsplanung im Infrastrukturbau und verfügt über einen sogenannten *Bridge Modeler* für den Brückenentwurf. Außerdem wurde in (Ji et al. 2010) ein *Ifc-Bridge Design Wizard* zur Generierung Ifc-basierter Brückenmodelle gezeigt. Im Folgenden werden diese Tools vorgestellt.

Civil Structures für Autodesk Revit 2014

Der Funktionsumfang der App Civil Structures ist in (Autodesk 2014c) wie folgt beschrieben: Mit Hilfe der Revit Bridge Modeling Erweiterungen, welche ausschließlich für kommerzielle Lizenzen erhältlich sind, können Brückenmodelle basierend auf Benutzerkriterien generiert werden. Der Anwender kann einfache Parameter der Brückengeometrie, wie Lageplan und Höhenprofil (zum Beispiel aus *LandXml*), Überbau, Pfeiler, Widerlager und Geländer definieren. Basierend auf vom Benutzer definierten Familien, welche mit den Erweiterungen geliefert werden, wird das Brückenmodell generiert. Die App ist nur in englischer Sprache installierbar.

Abbildung 6.14 zeigt zwei Screenshots der Benutzeroberfläche des PlugIns. Im linken Bild ist das Fenster zur Auswahl und Definition des Überbaus erkennbar. Auf der rechten Seite ist das Fenster zur Steuerung der Trassen und des Geländemodells sichtbar.

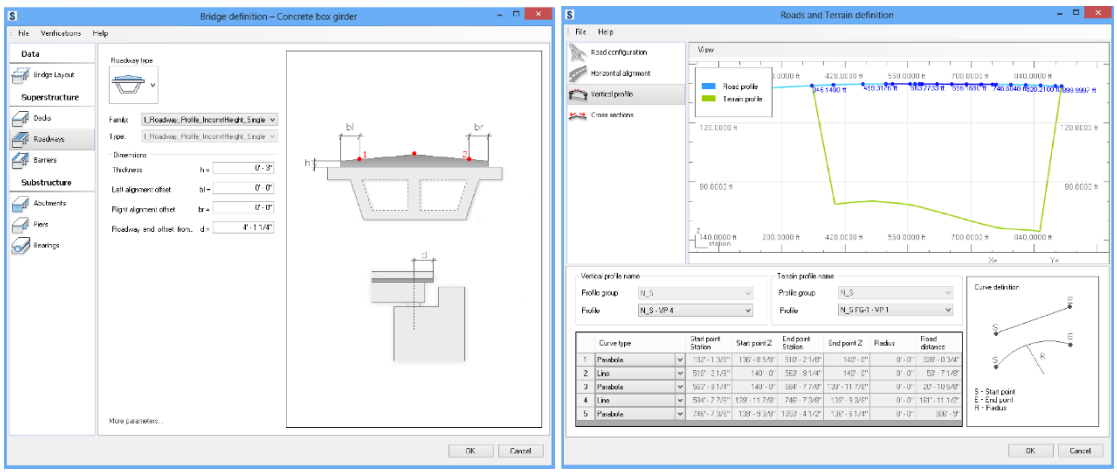


Abbildung 6.14: Civil Structures für Autodesk Revit 2014 (Autodesk 2014c)

Die Revit Bridge Modeling Erweiterungen bestehen aus den folgenden Modulen:

Das Modul Straßen- und Geländemodellierung erlaubt das Modellieren von Straßenelementen und des Geländemodells basierend auf gängigen Infrastrukturdatenformaten. Das Modul Integration in Civil3D bietet eine bi-direktionale Verknüpfung mit Civil3D, um die Zusammenarbeit zwischen Straßen- und Brückenplanern zu erleichtern. Die Module Brückenmodellierung-Massivplatte/Plattenbalken/Hohlkasten erlauben die Modellierung einer entsprechenden Brückengeometrie basierend auf einem Straßenelement. Außerdem besteht die Möglichkeit einer automatisierten Dokumentenableitung und Anpassung der Brückenfamilien mit Hilfe zweier weiterer Module.

Infraworks360 Bridge Design Module

Infraworks360 (Autodesk 2014a) ist eine Autodesk-Software zur Entwurfsplanung im Infrastrukturbau und verfügt über ein Brückenmodul für den Entwurf von Brücken. Ausgehend von einer modellierten, oder aus Civil3D oder Open Street Map (OSM) entnommenen Straße kann durch simple Definition der Start- und Endstationierung der Brücke über einen Slider eine Brücke automatisch generiert werden. Die Verknüpfung von Straße und Brücke bleibt stets erhalten, so dass Änderungen an Lage, Höhe oder Typ der Straße sich unmittelbar auf das Brückenmodell auswirken. Die Böschungs- und Geländemodellierung wird von *Infraworks* automatisch vorgenommen. *Infraworks* schlägt dem Benutzer eine bestimme Anzahl an Pfeiler vor, die aber jederzeit vom Benutzer geändert werden kann. Zudem ist ein einzuhaltendes Lichtraumprofil durch den Anwender definierbar. Eine Änderung der Stützweiten hat eine unmittelbare Neuberechnung der Brückenträger zur Folge. Zur Wahl stehen aktuell lediglich Stahl- oder Stahlbetonfertigteilträger. Dem Programm ist die amerikanische Herkunft deutlich anhand des hohen Anteils an Fertigteilen und den wenigen Konfigurationsmöglichkeiten anzumerken. So stehen die genannten Einstellungen nur für ein- oder mehrfeldrige Balkenbrücken mit dem immer gleichen Pfeilertyp zur Verfügung. Trotz allem bietet *Infraworks* im Vergleich zu anderen Produkten einmalige Features die zukünftig sicherlich verbessert oder den europäischen Bedürfnissen angepasst werden.

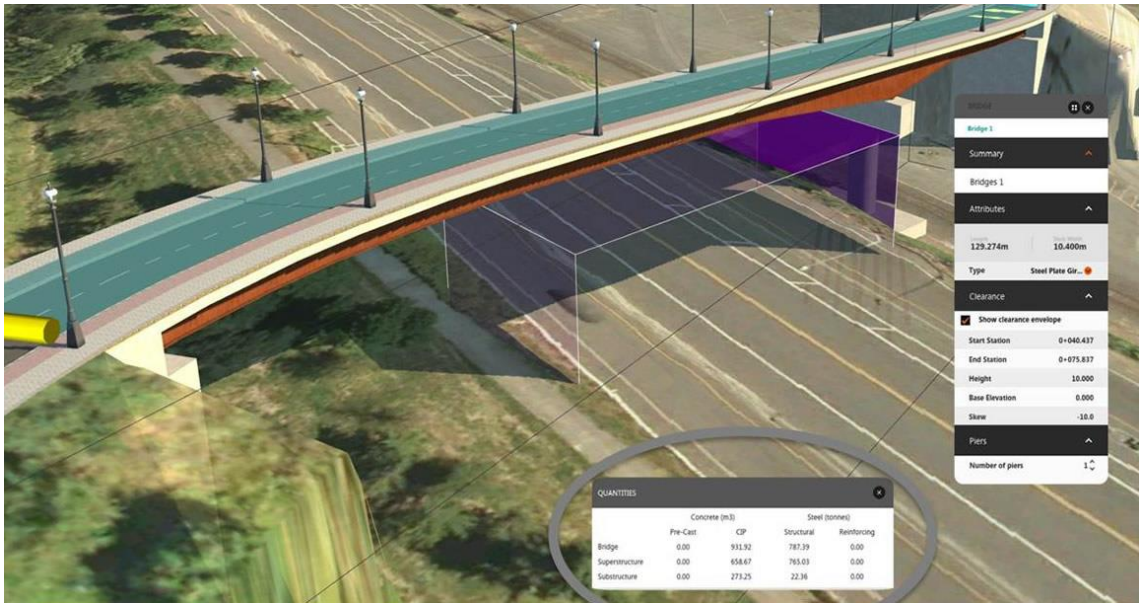


Abbildung 6.15: Brückenmodellierung in Infraworks 360 (Autodesk 2014a)

Das Brückenmodul implementiert einige heuristische Funktionen wie unter anderem die automatisierte Wahl des Trägers und der Trägerhöhe. Für eine Nutzung als KBE System sind diese Funktionen nicht umfassend genug, dennoch einmalig in der gegenwärtigen Softwarelandschaft der Baubranche.

Abbildung 6.15 zeigt einen Screenshot von Infraworks360 während der Bearbeitung eines Brückenmodells. Es ist eine zweifeldrige Rahmenbrücke erkennbar. Rechts im Bild ist ein Panel erkennbar, welches die Anpassung der Parameter des Brückenmodells ermöglicht.

Ifc-Bridge Design Wizard

Ifc-Bridge Design Wizard ist ein Tool, welches am Lehrstuhl für Computergestützte Modellierung und Simulation an der Technischen Universität München im Rahmen der Entwicklung eines Ifc-Standards für parametrische Brücken entstanden ist. Das Programm wurde ursprünglich als Prototyp zur Erzeugung von *IfcBridge* Beispieldateien entwickelt. So kann nach Wahl der verschiedenen Brückenkomponenten und Festlegung einiger Parameter durch den Anwender ein *IfcBridge*-Brückenmodell generiert werden (Ji et al. 2010). Durch Import dieser Modelle über weitere PlugIns könnten diese Modelle in Revit oder Siemens NX weiter bearbeitet werden. Abbildung 6.16 zeigt einen Screenshot des Programms. Im Ausschnitt ist die Definition der Überbauparameter erkennbar.

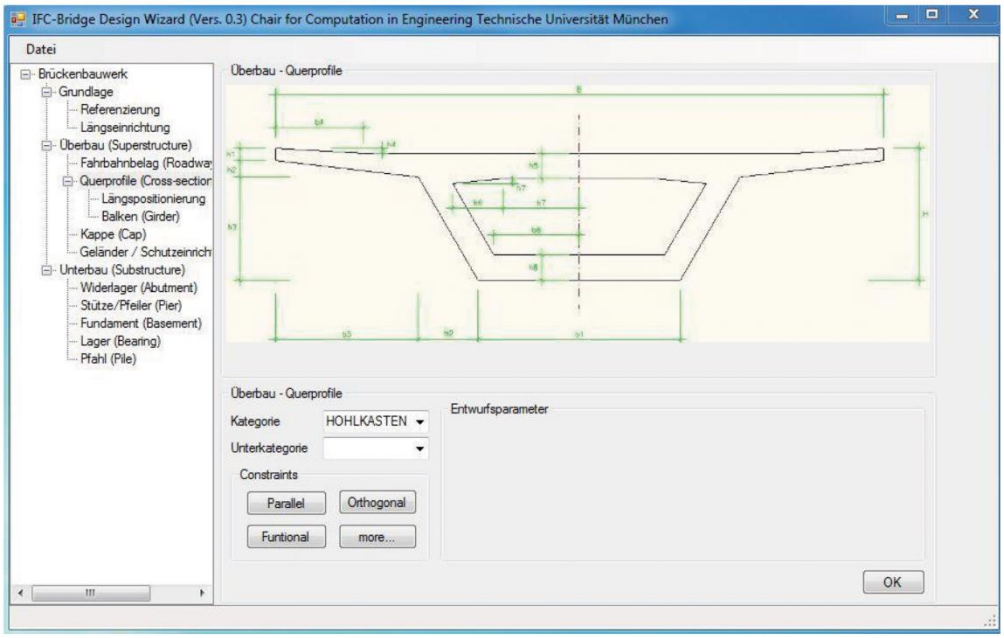


Abbildung 6.16: Ifc-Bridge Design Wizard (Ji et al. 2010)

6.4 Implementierung des KBE Prototypen

Dieser Abschnitt befasst sich mit der Implementierung des KBE Prototypen. Nach Vorstellung der verwendeten Programme zur Umsetzung der Implementierung, folgt die Beschreibung der Familienmodellierung in Revit. Außerdem werden die Funktionen der API Infrastructure.NET für das Lesen von LandXml Dateien beschrieben. Zum Schluss ist die Implementierung verschiedener Regeln des Brückenentwurfs in angeführten Graph-basierten Softwaresystem erklärt.

6.4.1 Revit 2014

Revit (Autodesk 2014b) ist ein 3D basiertes Building Information Modelling (BIM) Softwaresystem zur Planung von Bauwerken des Hoch- und Tiefbaus der Firma Autodesk. Revit enthält dafür Komponenten für Architekten (Revit Architecture), Ingenieure (Revit Structure) und Gebäudetechniker (Revit MEP).

Abbildung 6.17: Revit Logo (Autodesk 2014b)

Da später die Modellierung von Revit Familien behandelt wird, ist ein Blick auf die Elementstruktur von Revit sinnvoll (siehe Abbildung 6.18). Revit strukturiert Elemente nach Kategorien, Familien, Typen und Exemplaren (Autodesk 2009):

Kategorie Eine Kategorie gruppiert Familien ähnlicher Funktion und Eigenschaft zum Beispiel Wände oder Träger.

Familie Familien sind Klassen von Elementen gleicher Gestalt. In einer Familie werden Geometrie, Parameter und Verwendung des Elements definiert. Parameter besitzen meist Standardwerte und können von Typen oder Exemplaren belegt werden.

Typ Jede Familie kann, muss aber nicht, Typen besitzen. Typen belegen die Parameter einer Familie mit jeweils eigenen spezifischen Werten.

Exemplar Exemplare sind Instanzen einer Familie, welche in einem Projekt platziert wurden und sich durch Ort und Parameter von anderen Exemplaren der gleichen Familie unterscheiden.

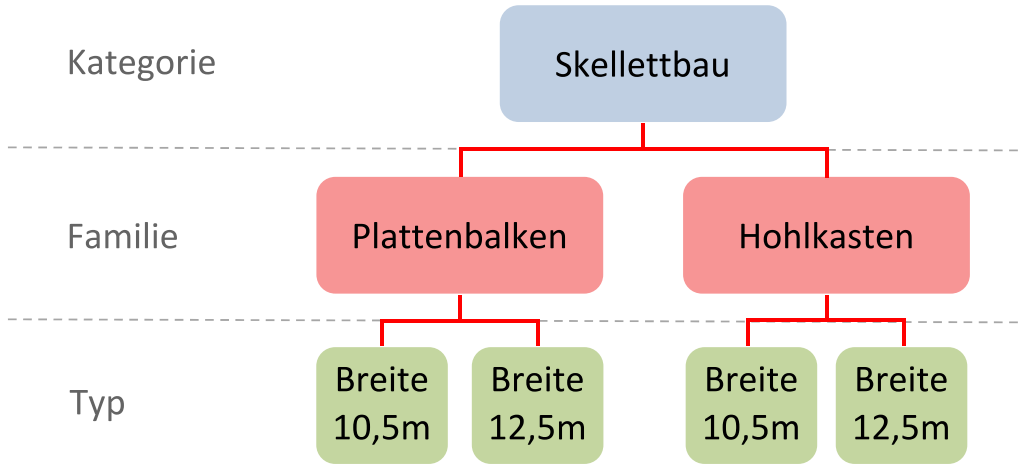


Abbildung 6.18: Revit Elementstruktur nach (Autodesk 2009)

Revit stellt Entwicklern eine API zur Verfügung. Über die API kann auf jedes Element in Revit zugegriffen, jedes Element bearbeitet sowie neue Elemente erstellt werden. Die Bearbeitung funktioniert sowohl für Familien, als auch für normale Revit Projekte.

6.4.2 Dynamo 0.7

Dynamo (Dynamo 2014) ist ein Open Source AddIn für Autodesk Revit und Autodesk Vasari. Die Entwicklung von Dynamo befindet sich aktuell im Betastatus, die derzeitige stabile Version ist 0.7. Dynamo erweitert die oben genannten Programme um den Leistungsumfang eines Graph-basierten Code Editors beziehungsweise einer visuellen Programmiersprache. Das AddIn stellt dem Anwender einige Grundfunktionalitäten wie mathematische Operationen, Listen, Scripting, Geometrieverarbeitung, Excel Im- und Export „out of the box“ zur Verfügung.

Abbildung 6.19: Dynamo Logo (Dynamo 2014)

Durch die Verknüpfung eines Revit Dokuments, sowohl Projekt als auch Familie, können Elemente innerhalb des Revit Projekts via Dynamo selektiert, gefiltert, geändert oder neue Elemente hinzugefügt werden. Hinter den Revit Knoten stehen die Funktionalitäten der Revit Application Programming Interface (API). Dadurch ist es auch für Anwender ohne Programmierkenntnisse möglich die Revit API für Aufgaben, wie Platzieren von Familienexemplaren, Erzeugen und Bearbeiten einer Familie, Änderung der Parameter eines Elements, zu nutzen. Dynamo verfügt außerdem über eine eigene Rendering Komponente zur Visualisierung der erzeugten Geometrie. Projekte von Dynamo können als .dyn Datei gespeichert und wieder aufgerufen werden. Zu beachten ist hierbei die Referenz zu einem bestimmten Revit Projekt, was bei Missachtung zu fehlerhaften Programmablauf führt.

Abbildung 6.20 zeigt die Benutzeroberfläche von Dynamo. Links im Bild ist die Bibliothek, welche alle geladenen und damit verwendbaren Knoten enthält, zu sehen. Zentral angeordnet ist der Bereich des graphischen Code Editors, in dem die Knoten platziert werden können. Über Ports können die platzierten Knoten verbunden werden. Mit Hilfe der Ports werden gleichzeitig die erwarteten Ein- und Ausgangsdatentypen definiert.

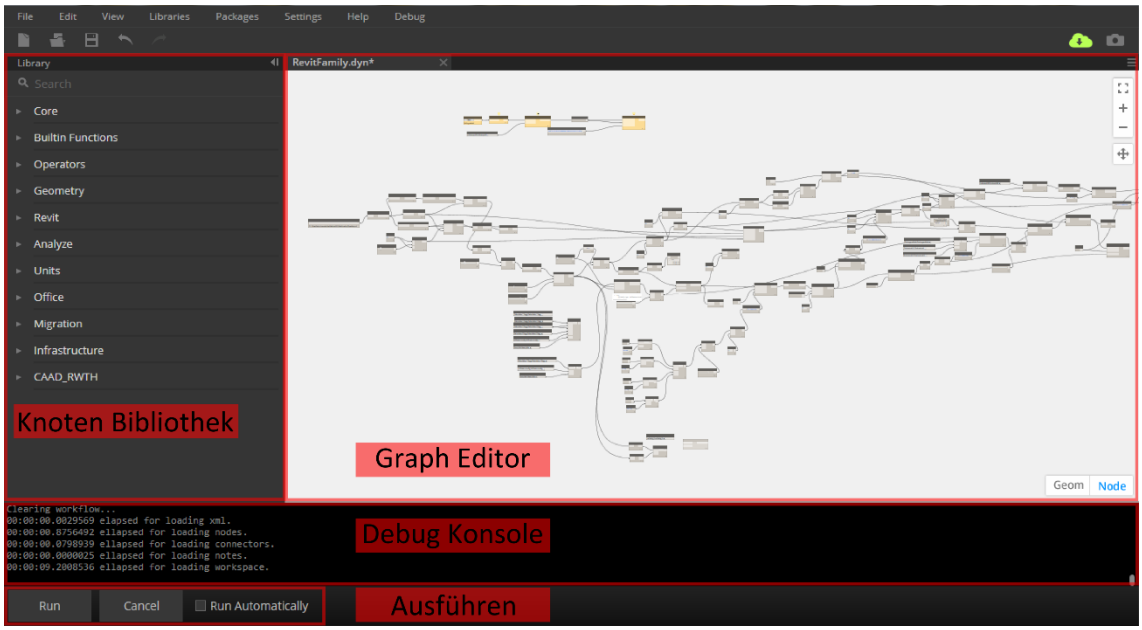


Abbildung 6.20: Dynamo Benutzeroberfläche

Im unteren Bereich befindet sich die Debug Konsole sowie Schaltflächen zur Ausführung des Programms. Wird *Run* gedrückt, so wird der vom Anwender definiert Programmablauf ausgeführt. Die dabei auftretenden Fehler oder Zwischenwerte werden dem Benutzer ausgegeben. Ist die automatische Ausführung eingeschaltet, findet bei Änderung der Ausgangsparameter eine sofortige Neuberechnung des Programms statt und das Ergebnis ist unmittelbar sichtbar. Dynamo kann sowohl aus Revit oder Vasari direkt als auch als „Standalone“ Version eigenständig betrieben werden.

Dynamo ist bereits standardmäßig ein mächtiges Werkzeug zur visuellen Programmierung für Revit oder Vasari. Sollten diese Funktionen nicht ausreichend sein, können sie bei Bedarf durch eigene Knoten erweitert werden. Dafür stehen mehrere Wege zur Wahl:

- Über *Custom Nodes*
- Über *Community* P*ackages*
- Über .dll Import
- Über *Fork* des Dynamo GitHub Projekts

Sind in Dynamo einige Knoten ausgewählt, so können diese zu einem *Custom Node* gruppiert werden. Dadurch wird Darstellung des Programms wesentlich übersichtlicher und das Arrangement kann mehrmals verwendet werden. *Custom Nodes* können außerdem über ein eigenes Projekt direkt implementiert werden. Allerdings stehen für *Custom Nodes* weiter nur die Grundfunktionalitäten zur Verfügung.

Über *Community Packages* können Knoten anderer Dynamo Anwender verwendet werden, falls diese ihre Knoten Packages in der Dynamo Community veröffentlicht haben. Damit stehen für häufig auftretende, aber nicht standardmäßig implementierte Problemstellungen schnell Lösungen bereit.

Eine dritte Möglichkeit Dynamo für seine Bedürfnisse anzupassen, ist das Importieren einer .dll Klassenbibliothek. Beim Import einer .dll generiert Dynamo automatisch Knoten aus den Klassen der Bibliothek. Abbildung 6.21 veranschaulicht diesen Vorgang anhand der Klasse *LandXmlParser*. Der Konstruktor der Klasse wurde in einen Knoten *LandXmlParser*.*LandXmlParser* konvertiert. Da der Konstruktor einen Parameter *filename* erwartet wurden dem Knoten auf der linken Seite ein Port mit dem Name des Parameters hinzugefügt. Als Ausgabe liefert der Knoten eine Instanz der Klasse *LandXmlParser* (rechter

Port). Gleichartig funktioniert die Konvertierung für Properties. Der Knoten erwartet eine Instanz der Klasse *LandXmlParser* und liefert ein Objekt im *return* Type des Properties.

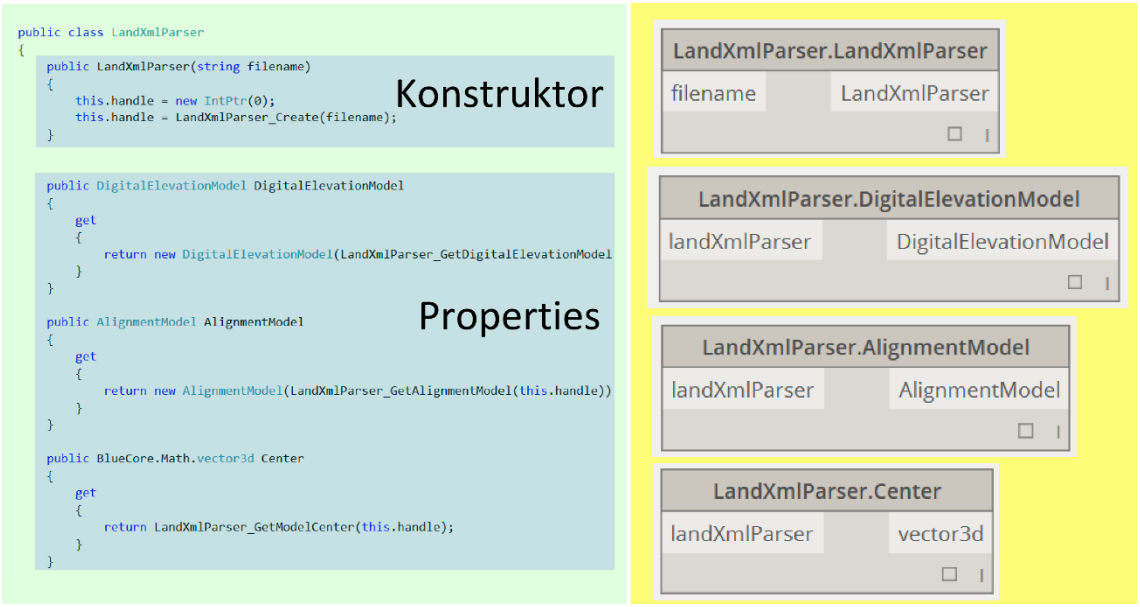


Abbildung 6.21: Dynamo .dll Import Funktion

Als ultima ratio kann ein Fork das Dynamo Projekts auf GitHub erstellt werden, um direkt im Dynamo Projekt individuelle Anpassungen oder Erweiterungen vorzunehmen.

6.4.3 Infrastructure.NET

Um in Revit beziehungsweise Dynamo Funktionalitäten für die Behandlung von *LandXml*-Dateien anbieten zu können und damit den Import von Trassierungs- und Geländedaten zu ermöglichen wurde für die bereits bestehende *TUM Open Infra Platform* (Amann 2014) ein C# Wrapper erstellt. Die *TUM Open Infra Platform* ist ein Programm zur Visualisierung von Trassierungsdaten und zur Konvertierung von *LandXml* zu *IfcAlignment* Dateien und anders herum. Entwickelt wird die *TUM Open Infra Platform* am Lehrstuhl für Computergestützte Modellierung und Simulation der Technischen Universität München und ist als Open Source Programm frei verfügbar.

Die entwickelte Schnittstelle I*nfrastructure.NET* wurde in C# programmiert und greift auf die Funktionen der in C++ entwickelten *TUM Open Infra Platform* zu. Somit steht einem Entwickler innerhalb des .NET Frameworks ein umfangreicher *LandXmlParser* zur Verfügung. Es können sämtliche Trasseninformationen und das Geländemodell einer *LandXml* gelesen und für die jeweils eigene Zwecke verwendet werden.

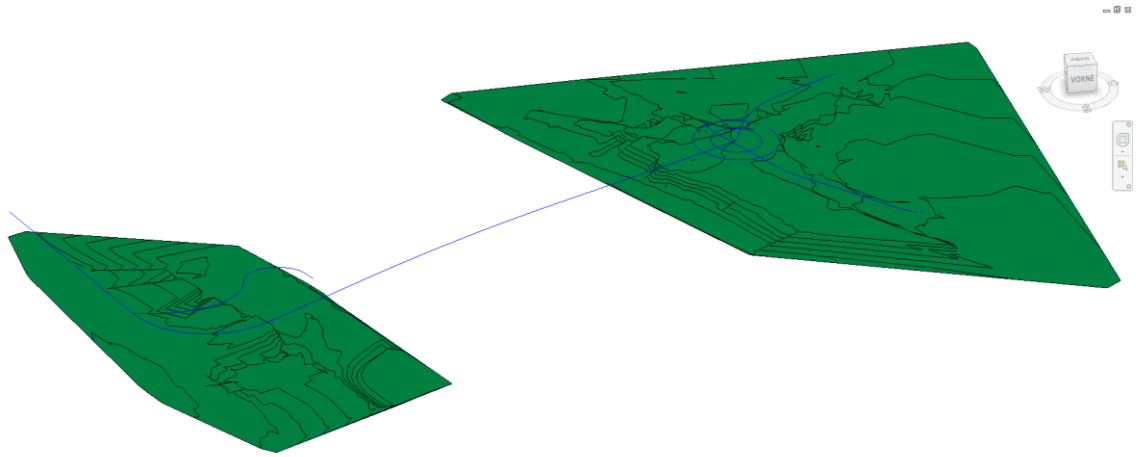


Abbildung 6.22: Nutzung der Infrastructure.NET Schnittstelle in Revit

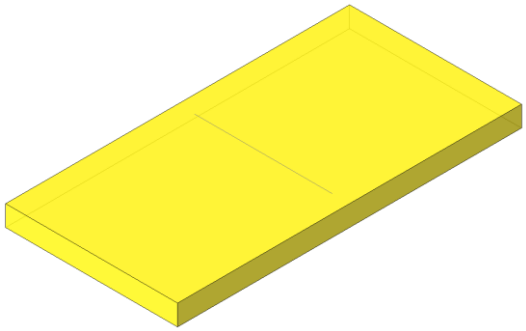
Wie in Kapitel 6.4.2 bereits gezeigt kann diese .dll Klassenbibliothek sehr einfach in Dynamo importiert werden. Abbildung 6.22 zeigt die Visualisierung einer importieren *LandXml* Datei in Revit. Die Trassierungsdaten bilden nun die Grundlage für die Modellierung des Brückenmodells mit Hilfe des KBE Prototypen. Mit Hilfe des Knoten *GetIfcAlignmentByIndex* kann eine Achse ausgewählt werden, entlang welcher in den folgenden Knoten das Brückenmodell modelliert werden kann.

6.4.4 Modellierung der Revit Familien

Für die Umsetzung des KBE Prototypen in Revit ist zunächst die Modellierung einiger Familien erforderlich. Alle Familien gehören der Kategorie Skelettbau an. Dadurch lassen sich diese Familien entlang einer Referenzlinie platzieren. In diesem Anwendungsfall sind die Straßen- und Pfeilerachsen die Referenzlinie zur Platzierung. Die Wahl der Kategorie wurde aus verschiedenen Gründen notwendig. Zunächst ist es möglich, dass die Straßenachse gekrümmt ist und demnach auch die Überbaufamilien entlang dieser Krümmung geführt werden müssen. Außerdem besitzt der Überbau eine Querneigung, welche mit Hilfe der Familie abgebildet werden muss. Es muss also möglich sein, die Familie um ihre Referenzlinie zu rotieren. Genauso sollen alle Pfeiler entlang ihrer Achse verdrehbar sein, um eine Ausrichtung im Raum zu ermöglichen. Weiter wären bei ebenenbasierten Familien eine Menge an neuen Ebenen zu generieren, da jede Stütze aufgrund des Geländes eine unterschiedliche Höhe aufweist und damit für jede Stütze eine eigene Ebene notwendig wäre. Außerdem sollte jeder Familie ein Berechnungsmodell des Tragwerks zugrunde liegen. All diese aufgeführten Anforderungen können von Familien der Kategorie Skelettbau erfüllt werden. Negativ zu bemerken ist, dass diese Familien nicht in zwei Raumrichtungen gleichzeitig gekrümmt werden können, wodurch eine Modellierung von Brücken mit nicht-linearem Höhenprofil und gleichzeitiger Krümmung in der horizontalen Ebene nicht möglich ist. Ebenso nicht möglich ist die Modellierung einer Querneigungsänderung entlang der Brückenachse. Eine mögliche Alternative wäre hier die Modellierung der kompletten Brücke in einer einzigen Familie mit Hilfe von weiteren Adaptiven Familien und deren Platzierung im Entwurfsprojekt. Die Parameter wurden so generisch festgelegt, dass jede Überbaufamilie mindestens die Parameter *Höhe*, *Breite*, *Querschnittdrehung* und *Pfeilerbreite* aufweisen. Für die Plattenbalken-Familien kommt noch der Parameter *BreiteSteg* hinzu. Diese Parameter werden zur Laufzeit dynamisch über Knoten in Dynamo mit Werten belegt. Analog wurden Familien für Fundament und Widerlager modelliert. Im Gegensatz zur Fundament-Familie, ist die Widerlager-Familie wesentlich komplexer. Neben der Abhängigkeit vom Überbau der Brücke, sind Böschungsneigung und Kreuzungswinkel wichtige Parameter für die Konstruktion des Widerlagers. Auf den folgenden beiden Seiten ist eine Übersicht der modellierten Familien, inklusiver ihrer zugehörigen Parameter aufgeführt.

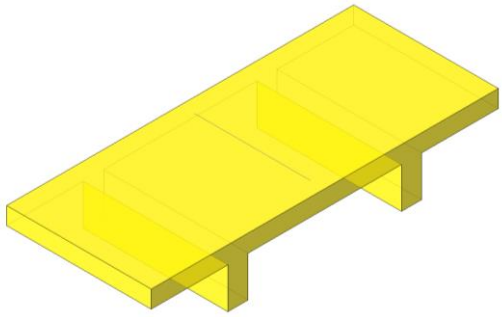
Massivplatte

- Höhe [m]
- Breite [m]
- Querschnittdrehung [°]
- Pfeilerbreite [m]



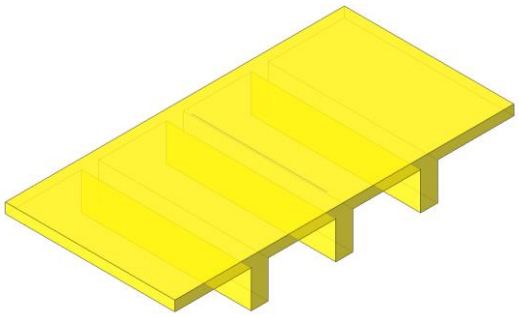
Plattenbalken 2-stegig

- Höhe [m]
- HöhePlatte [m]
- Breite [m]
- Querschnittdrehung [°]
- BreiteSteg [m]
- Pfeilerbreite [m]



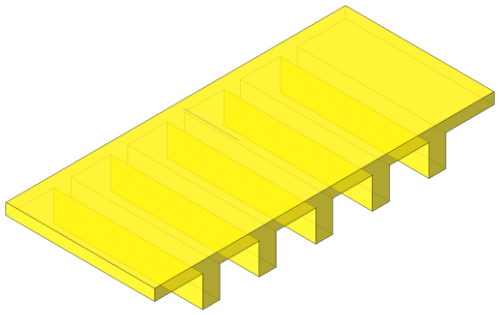
Plattenbalken 3-stegig

- Höhe [m]
- HöhePlatte [m]
- Breite [m]
- Querschnittdrehung [°]
- BreiteSteg [m]
- Pfeilerbreite [m]



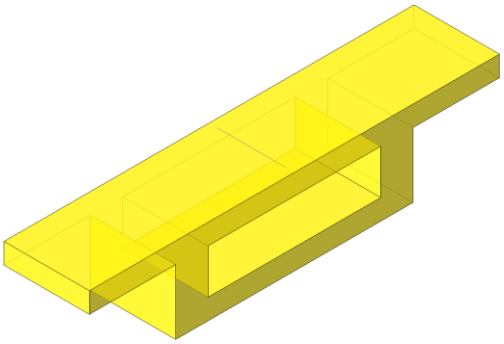
Plattenbalken 5-stegig

- Höhe [m]
- HöhePlatte [m]
- Breite [m]
- Querschnittdrehung [°]
- BreiteSteg [m]
- Pfeilerbreite [m]



Hohlkasten

- Höhe [m]
- HöhePlatte [m]
- Breite [m]
- Querschnittdrehung [°]
- BreiteSteg [m]
- Pfeilerbreite [m]

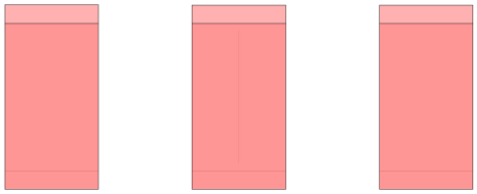


Stütze einzeln 2x

- Tiefe [m]
- Breite [m]

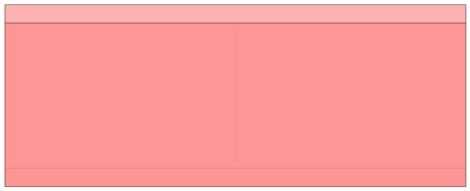
Stütze einzeln 3x

- Tiefe [m]
- Breite [m]



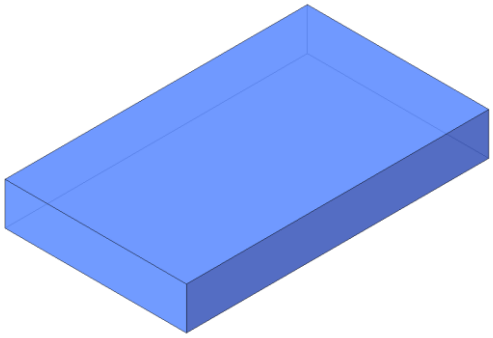
Stütze ganze Breite

- Tiefe [m]
- Breite [m]



Fundament

- Breite [m]
- Länge [m]
- Tiefe [m]



Widerlager

- Breite [m]
- FlügelTiefe [m]
- Wandstärke [m]
- HöheAuflagerbank [m]
- Flügelrichtung [°]

6.4.5 Implementierung der Entwurfsregeln

Dieser Abschnitt befasst sich nun mit der Implementierung der regelbasierten Brückenkonstruktion in Autodesk Revit und Dynamo. Die dafür notwendigen Vorarbeiten wie die Modellierung der Familien in Revit und die Erweiterungsfunktionalitäten von Dynamo wurden in den vorangegangen Kapiteln abgeschlossen. In Kapitel 6.2 wurden bereits mögliche Entwurfsregeln identifiziert. Die Umsetzung einiger dieser Regeln wird im Folgenden beschrieben. Dazu wird der Entwurfsprozess in kleinere Subprozesse aufgeteilt und die entsprechende Implementierung aufgezeigt.

Import LandXml

Die Brückenachse und das Geländemodell sind für alle weiteren Schritte als Referenz nötig. Im KBE Prototyp werden Straßenachsen und das Geländemodell aus einer *LandXml* Datei importiert. Die dafür verwendete API *Infrastructure.NET* ist in Kapitel 6.4.3 bereits beschrieben. Mit Hilfe der Knoten *File Path* kann eine zu importierende Datei ausgewählt und mit dem Knoten *LandXmlParser*.*LandXmlParser* geparst werden. Abbildung 6.23 zeigt wie dieser Vorgang im Graph modelliert ist. Alternativ zum *LandXml* Import kann die Brückenachse auch direkt in Revit modelliert werden. Anschließend wird die gezeichnete Kurve mit Hilfe des Knotens *Element Selection* in Dynamo verwendet.

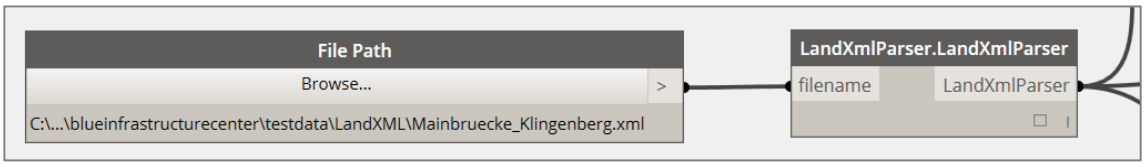


Abbildung 6.23: Knoten Import LandXml

Auswahl der Trasse

Wurde eine *LandXml* Datei importiert, kann nun aus allen verfügbaren Achsen die für die Brücke nötige Achse ausgewählt werden. Alle Achsen sind in Dynamo in einer Liste gespeichert. Elemente der Liste, also eine Kurve, kann mit Hilfe des Knotens *GetItemAtIndex* und einem *Integer Slider* ausgewählt werden. Durch Verschieben des *Sliders* kann die entsprechende Kurve aus der Liste gewählt werden und anderen Knoten zur Verarbeitung zugeführt werden. Abbildung 6.24 zeigt die Modellierung dieses Schrittes im Dynamo Graph.

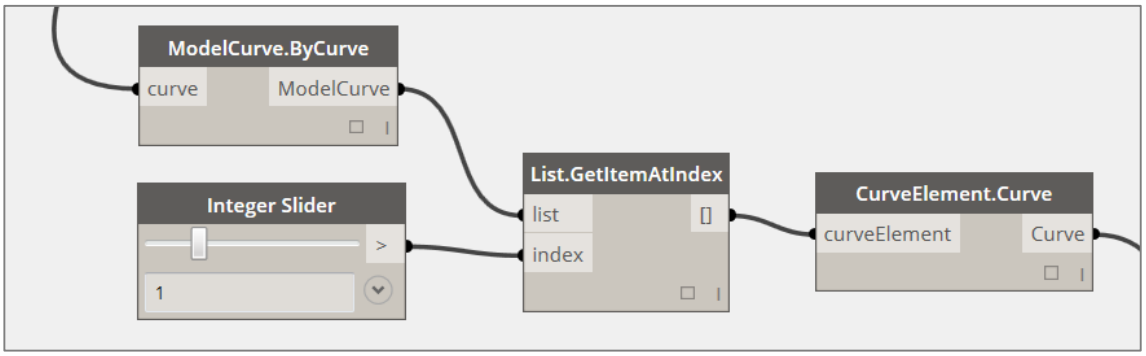


Abbildung 6.24: Knoten Auswahl Achse

Festlegen von Anfangs- und Endstationierung der Brücke

Zur Eingrenzung der Start- und Endstationierung der Brücke auf der zuvor ausgewählten Trasse, steht ein *Trim* Knoten zur Verfügung. Zwei *Slider* geben dabei den *StartParamter* und *EndParameter* (Werte zwischen 0 und 1) vor, nach denen dann die ausgewählte Kurve beschnitten wird. Nach Abschluss dieses Prozesses steht die endgültige Brückenachse zur weiteren Verwendung zur Verfügung. Abbildung 6.25 zeigt die Umsetzung dieses Vorgangs im Dynamo Graph.

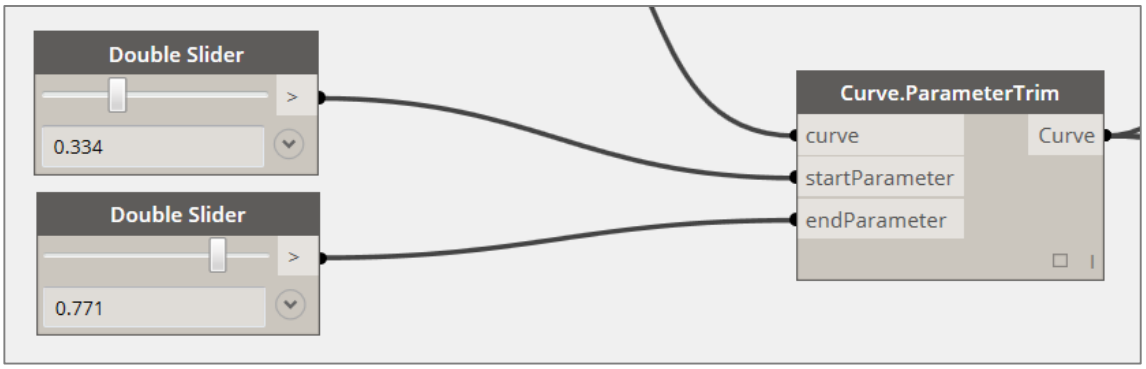


Abbildung 6.25: Knoten Stationierung Brückenachse

Festlegung der Pfeileranzahl

Nächster notwendiger Eingangsparameter zur Modellierung der Brücke ist die Anzahl der Pfeiler. Im Moment kann die Anzahl über einen *Slider* explizit festgelegt werden. Für weitere Entwicklungen ist aber die regelbasierte Ermittlung der Pfeilerpositionen denkbar. Ausgehend von dieser Angabe wird die Brückenachse in Abschnitte gleicher Länge unterteilt (Abbildung 6.26).

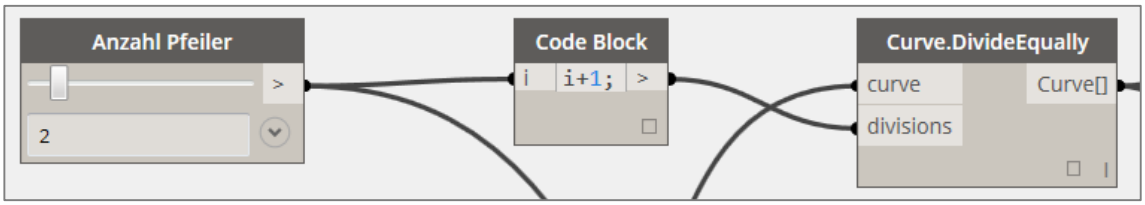


Abbildung 6.26: Knoten Anzahl Pfeiler

Wahl des Regelquerschnitts

Die Regelquerschnitte geben die Breite des Überbaus vor. Zunächst werden vier verschiedene Regelquerschnitte definiert, nämlich RQ7.5, RQ9.5, RQ10.5 und RQ15.5. Aus diesen vier Typen wird eine Liste erstellt, aus der dann der gewünschte Regelquerschnitt über den Knoten *GetItemAtIndex* und einem *Integer Slider* ausgewählt werden kann. Der Aufbau des Graphen zur Definition und Wahl der Regelquerschnitte ist in Abbildung 6.27 dargestellt.

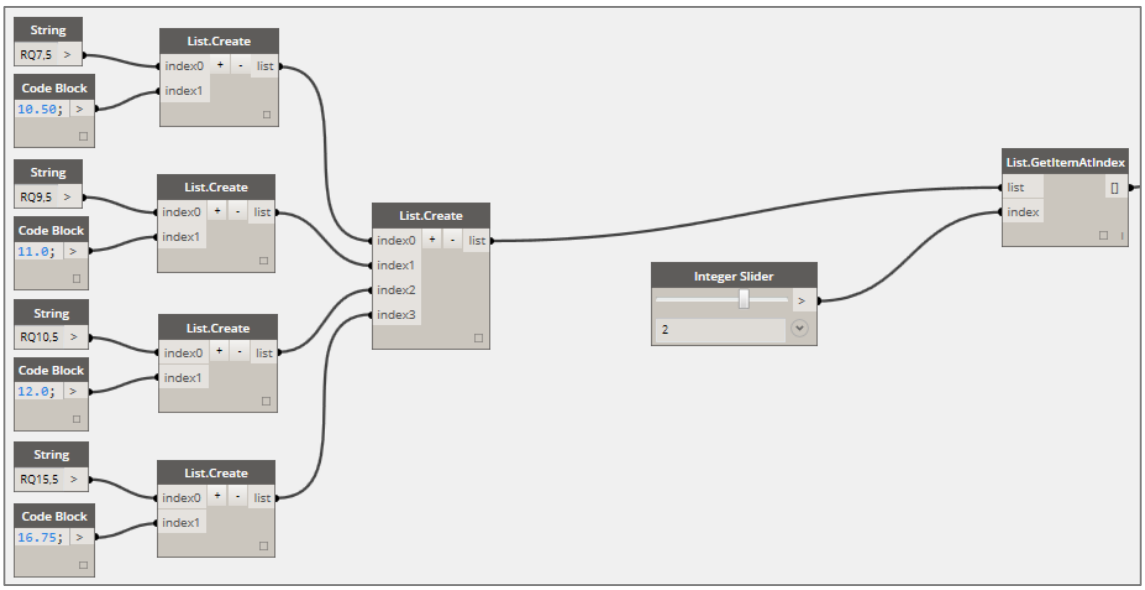


Abbildung 6.27: Knoten Regelquerschnitte

Laden der Überbaufamilien

Zur Erstellung des Brückenmodells wurden, wie in Kapitel 6.4.4 beschrieben, Familien modelliert. Um diese in Dynamo verwenden zu können, müssen sie zunächst geladen werden. Dies funktioniert mit Hilfe des vorgefertigten Knotens *Family Types*. Für den Pro-

totypen werden, wie in Abbildung 6.28 dargestellt, drei Überbaufamilien, nämlich *Massivplatte*, *Hohlkasten einzellig*, *Plattenbalken 2 Stegig* geladen und in einer Liste gespeichert.

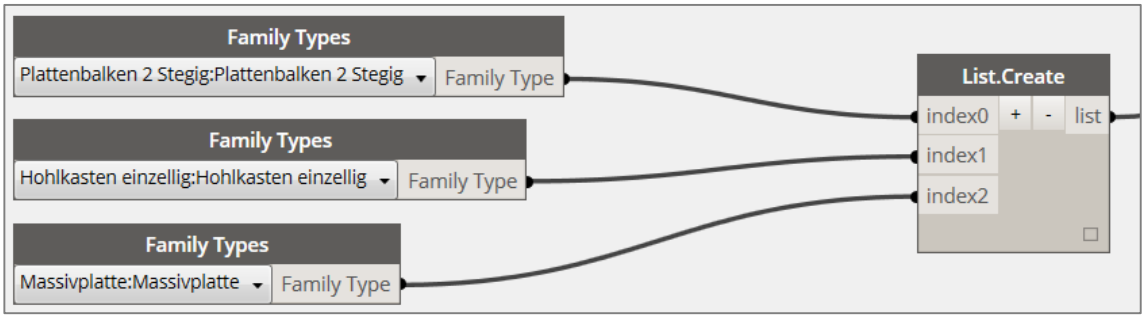


Abbildung 6.28: Knoten Überbaufamilien

Regelbasierte Wahl des Überbaus

Dieser Prozess beinhaltet die Umsetzung der ersten KBE Regel: Der Wahl des Überbautyps. Dazu wurde eigens der Knoten *KBE.WahlUeberbau* implementiert. Ausgehend von den geladenen Überbaufamilien, der Brückenachse und der Anzahl an Stützen, folgt die regelbasierte Wahl eines geeigneten Überbautyps. Im Rahmen der Prototypenentwicklung stehen Massivplatte, Plattenbalken und Hohlkasten zur Wahl. Aus ihnen wird abhängig von der berechneten Stützweite ein geeigneter Typ ausgewählt. Es ist zu beachten, dass keine Wahl sondern viel mehr eine Filterung in Frage kommender Typen stattfindet. So hat der Anwender im Nachgang die Möglichkeit aus einer Liste möglicher Überbautypen einen auswählen. Abbildung 6.29 zeigt den platzierten Knoten in Dynamo, während Abbildung 6.30 den zugrundeliegenden Code zeigt. Im Rahmen dieser Arbeit wurden die Regelgrenzen direkt im C# Code definiert. Denkbar ist hier die Erweiterung des Knotens um weitere Parameter. So kann der Anwender die Regelgrenzen selbst definieren.

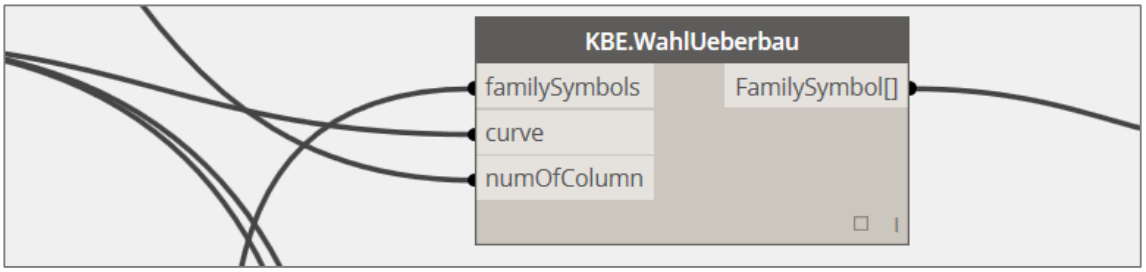


Abbildung 6.29: Knoten Wahl Überbau

```
public static List<FamilySymbol> WahlUeberbau(List<FamilySymbol> familySymbols, Curve curve,
int numOfColumn)
{
    List<FamilySymbol> output = new List<FamilySymbol>();

    // Massivplatte
    if (curve.Length / (numOfColumn + 1) < 15) // wenn Stützweite kleiner als 15m
    {
        output = familySymbols.Where(x => x.Family.Name.Contains("Massivplatte")).ToList();
    }
    // Plattenbalken
    else if (curve.Length / (numOfColumn + 1) < 25) // wenn Stützweite kleiner als 25m
    {
        output = familySymbols.Where(x => x.Family.Name.Contains("Plattenbalken")).ToList();
    }
    // Hohlkasten
    Else // wenn alle anderen Stützweiten
    {
        output = familySymbols.Where(x => x.Family.Name.Contains("Hohlkasten")).ToList();
    }

    if (output.Count == 0) output = null;

    return output; // Rückgabe der gefilterten Familien

}
```

Abbildung 6.30: Code Knoten Wahl Überbau

Platzieren des Überbaus auf der Brückenachse

Wurde ein Überbautyp ausgewählt, kann diese Überbaufamilie nun auf der gegebenen Kurve im Revit Projekt platziert werden. Dazu wurde eigens der Knoten *FamilyInstance.ByCurveAndSymbol* für Dynamo implementiert (siehe Abbildung 6.31). Der Knoten erwartet als Eingangsparameter die Kurve und die gewählte Überbaufamilie und liefet das platzierte Exemplar zurück.

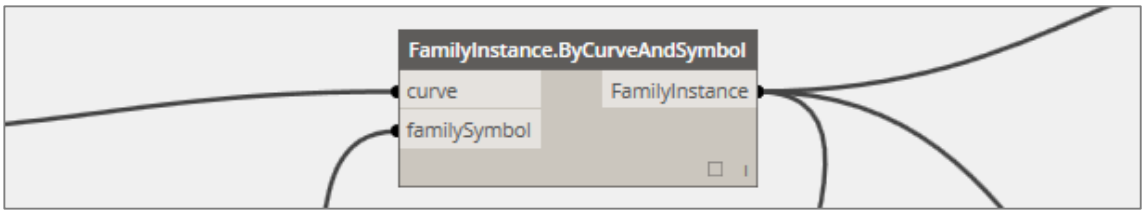


Abbildung 6.31: Knoten Platzierung Überbau auf Brückenachse

Festlegung der Überbauparameter Breite, Querschnittdrehung

Ausgehend vom gewählten Regelquerschnitt werden nun die Parameter des Exemplars festgelegt. Mit Hilfe des Knotens *Element.SetParameterByName* werden die Parameter *Breite* und *Querschnittdrehung* mit Werten belegt (Abbildung 6.32). Die weiteren Parameter wie *Hoehe*, *BreiteSteg* und *Pfeilerbreite* werden in der Familie selbst berechnet.

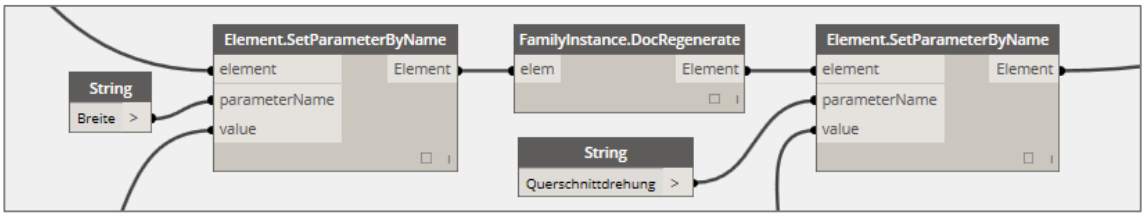


Abbildung 6.32: Festlegung der Überbauparameter

Laden der Pfeilerfamilien

Wie bei den Überbaufamilien müssen auch die Pfeilerfamilien zunächst geladen werden. Dies funktioniert auf selbe Art und Weise wie für die Überbautypen. Geladen werden die Stützenfamilien *ganzeBreite*, *einzeln2x* und *einzeln3x*, welche anschließend in einer Liste gespeichert werden.

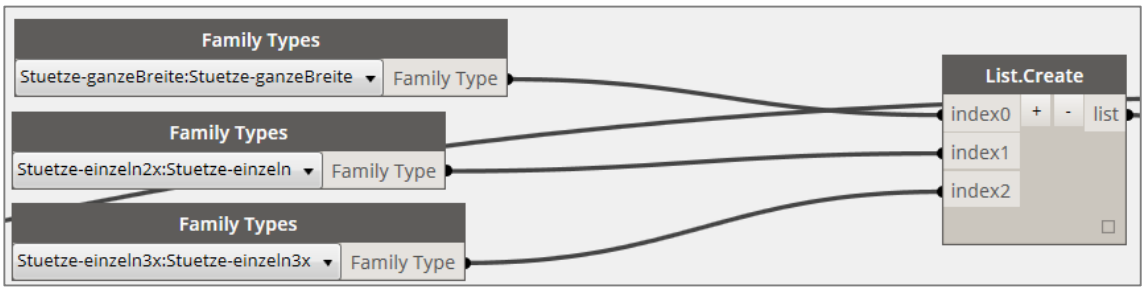


Abbildung 6.33: Knoten Pfeilerfamilien

Regelbasierte Wahl der Pfeiler

Dieser Vorgang bildet die zweite implementierte KBE Regel ab. Zur Umsetzung dieser Regel wurde der Knoten *WahlPfeiler* entwickelt. Abhängig vom bereits gewählten Überbautyp, den verfügbaren Pfeilerfamilien und der Breite des Überbaus werden geeignete Pfeilerfamilien ausgewählt. So ist beispielsweise bei bereits gewähltem 2-stegigem Plattenbalken nur die Wahl der Pfeilerfamilien *ganzeBreite* oder *einzeln2x* sinnvoll. Eine Pfeilerfamilie mit drei Pfeilern bei 2 Stegen im Überbau wäre wenig praktikabel. Außerdem werden die Familien in Abhängigkeit zur Überbaubreite ausgewählt. Die Regel ist wie in Abbildung 6.35 gezeigt, direkt im C# Code umgesetzt. Auch hier findet lediglich eine Filterung möglicher Pfeilerfamilien statt, der Anwender kann durch eine entsprechende Wahl nach wie vor Einfluss auf die Konstruktion nehmen. Abbildung 6.34 zeigt die Verwendung des Knotens *KBE.WahlPfeiler* in Dynamo.

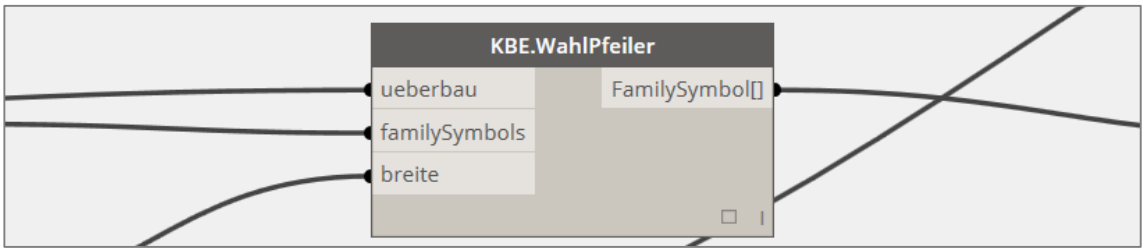


Abbildung 6.34: Knoten Wahl Pfeiler

```
public static List<FamilySymbol> WahlPfeiler(

FamilySymbol ueberbau, List<FamilySymbol> familySymbols, double breite

){
    List<FamilySymbol> output = new List<FamilySymbol>();

    // Hohlkasten
    if (ueberbau.Name.Contains("Hohlkasten") || ueberbau.Name.Contains("Massivplatte"))
    {
        output.AddRange(familySymbols.Where(x => x.Family.Name.Contains("ganzeBreite")).To
            List());

        if (breite < 8)
        {
            output.AddRange(familySymbols.Where(x => x.Family.Name.Contains("2x")).ToList());
        }
        else if (breite < 15)
        {
            output.AddRange(familySymbols.Where(x => x.Family.Name.Contains("3x")).ToList());
        }
    }
    // Plattenbalken
    else if (ueberbau.Name.Contains("Plattenbalken"))
    {
        output.AddRange(familySymbols.Where(x => x.Family.Name.Contains("ganzeBreite")).To
            List());

        if (ueberbau.Name.Contains("2 Stegig"))
        {
            output.AddRange(familySymbols.Where(x => x.Family.Name.Contains("2x")).ToList());
        }
        else if (ueberbau.Name.Contains("3 Stegig"))
        {
            output.AddRange(familySymbols.Where(x => x.Family.Name.Contains("3x")).ToList());
        }
    }

    if (output.Count == 0) output = null;

    return output;

}
```

Abbildung 6.35: Code Wahl Pfeiler

Platzieren der Pfeiler auf den Pfeilerachsen

Die Platzierung der Pfeilerfamilien geschieht auf selbe Art wie die Platzierung des Überbaus. Mit Hilfe des Knotens *FamilyInstance.ByCurveAndSymbol* werden die Pfeilerfamilien auf jeder zuvor ermittelten Pfeilerachse platziert.

Festlegung der Pfeilerhöhe abhängig vom Geländemodell

Eine weitere Regel ist die Festlegung der Pfeilerhöhe abhängig vom Geländemodell. Mit Hilfe der *Infrastructure.Net* API kann auf das Geländemodell zugegriffen und die Geländehöhe am Pfeilerstandort abgefragt werden. Dadurch ist es möglich, jedem Pfeiler eine individuelle Höhe zuzuweisen. Für die Umsetzung wurde der Knoten *GetGroundPoint* entwickelt. Ausgehend vom Achspunkt, dem Geländemodell und weiterer Hilfsgrößen, wird der passende Geländepunkt zurückgegeben.

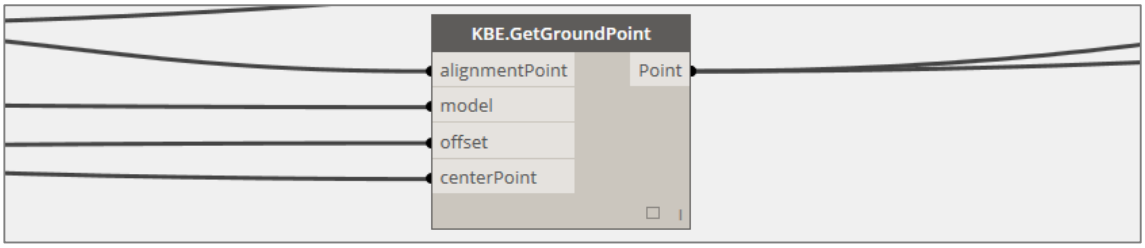


Abbildung 6.36: Knoten Pfeilerhöhe

Ausrichtung der Pfeiler

Platziert man die Pfeilerfamilien im Revit Projekt, so werden sie standardmäßig nach Norden ausgerichtet. Eine Ausrichtung der Pfeiler senkrecht zur Brückenachse oder parallel zur unterführten Trasse soll mittels des KBE Prototypen regelbasiert möglich sein. Zur Umsetzung der Rotation der Pfeiler wurde die *Custom Node GetRotationAngle* erstellt (Abbildung 6.37). Der Knoten liefert den Rotationswinkel, um den die Pfeiler anschließend gedreht werden.

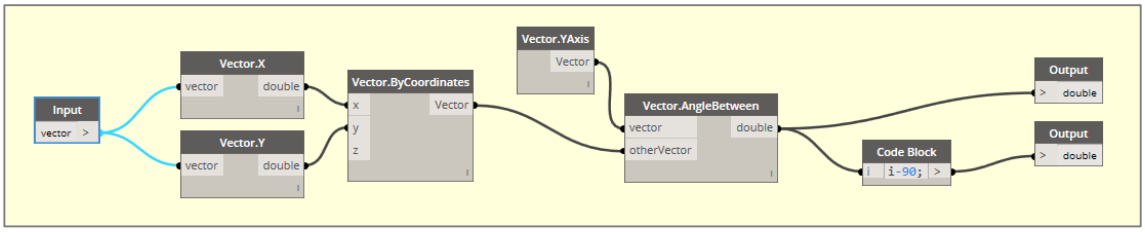


Abbildung 6.37: *Custom Node GetRotationAngle*

Festlegung des Pfeilerparameters Breite

Die Breite der Pfeiler wird in Abhängigkeit zur gewählten Überbaufamilie festgelegt. Die Breite der Pfeiler entspricht nur im Falle der Massivplatte der Breite des Regelquerschnitts. Im Falle des Plattenbalkens entspricht die Pfeilerbreite dem Abstand der Stege, für den Hohlkasten der Breite des Kastens. Um die Implementierung so generisch wie möglich zu halten, wurde für alle Überbaufamilien der Parameter *Pfeilerbreite* definiert (siehe Kapitel 6.4.4). Dieser Parameter wird zur Laufzeit aus den Exemplaren der Überbaufamilie ausgelesen und dem Parameter *Breite* der Pfeiler zugwiesen (Abbildung 6.38).

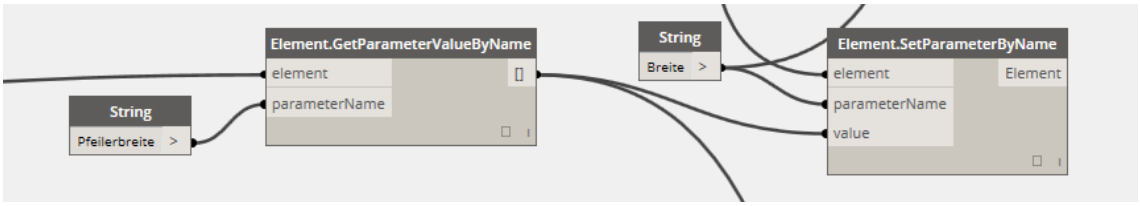


Abbildung 6.38: Festlegung der Pfeilerparameter

Laden der Fundamentfamilie und Platzierung des Fundaments

Durch die Berechnung der Pfeilerachsen sind die Fußpunkte der Stützen bekannt. Nach Laden der Fundamentfamilie wird diese an jedem Fußpunkt platziert. Für diesen Schritt kommt der Knoten *FamilyInstance.ByPoint* zur Anwendung, da es sich hierbei um eine punktbasierte Familie handelt (Abbildung 6.39). Anschließend können den Fundamentexemplaren die Parameter *Breite*, *Länge* und *Tiefe* zugwiesen werden.

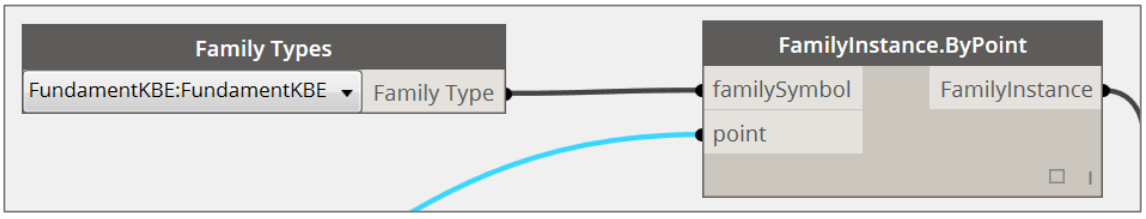


Abbildung 6.39: Laden und Platzieren der Gründung

Laden und Platzieren der Widerlager

Jeweils am Anfang und Ende der Brückentrasse werden Widerlager angeordnet. Die Punkte zur Platzierung der Widerlagerfamilie werden aus den Start- und Endpunkten der Kurve ermittelt. Nach Laden einer Widerlagerfamilie kann diese an den Punkten mit Hilfe

des Knotens *FamilyInstance.ByPoint* platziert werden (Abbildung 6.40). Nach der Platzierung ist die Belegung der Widerlagerparameter *Breite*, *FlügelTiefe*, *Wandstärke* , *Höhe-Auflagerbank* und *Flügelrichtung* möglich. Die Breite des Widerlagers ist vom gewählten Regelquerschnitt, die Höhe der Auflagerbank vom Überbau und die Ausrichtung der Widerlagerflügel vom Kreuzungswinkel abhängig.

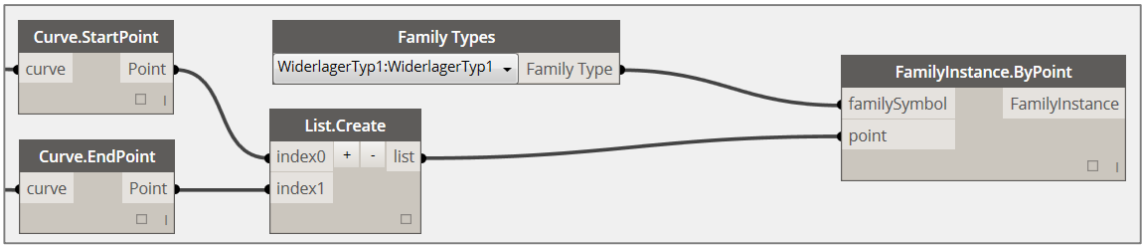


Abbildung 6.40: Platzierung Widerlager

Ergebnis

Nach Durchlaufen aller Vorgänge beziehungsweise Knoten ist das den Randbedingungen entsprechende Brückenmodell konstruiert. Das Modell kann nun für weitere Schritte wie statische Berechnung, Kalkulation oder andere verwendet werden. Auch eine händische Fortführung der Konstruktion in Revit ist möglich. In Abbildung 6.41, Abbildung 6.42, Abbildung 6.43 und Abbildung 6.44 sind jeweils unterschiedliche mit Hilfe des Prototypen erzeugte Brückenkonfigurationen zu sehen.

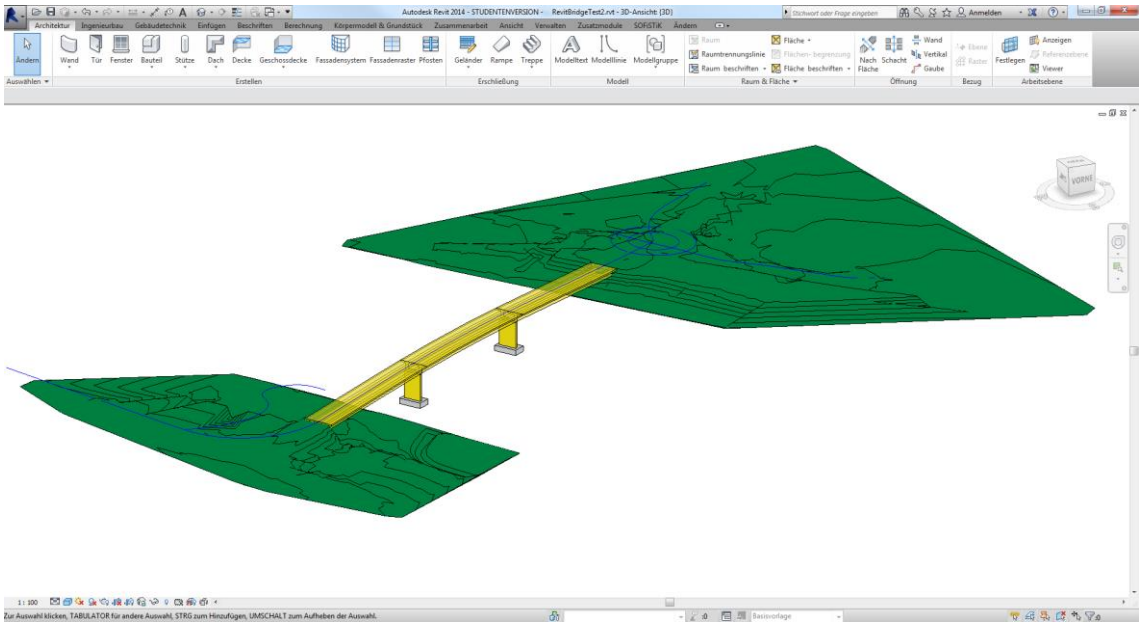


Abbildung 6.41: Screenshot Revit KBE Prototyp

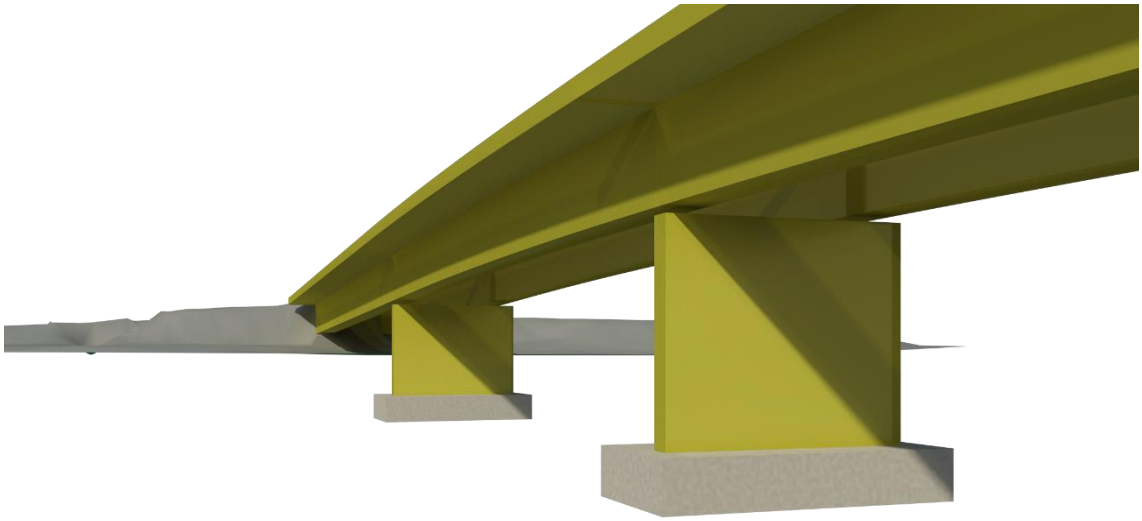


Abbildung 6.42: Revit Rendering 4 Felder, Plattenbalken mit Stütze "ganzeBreite"

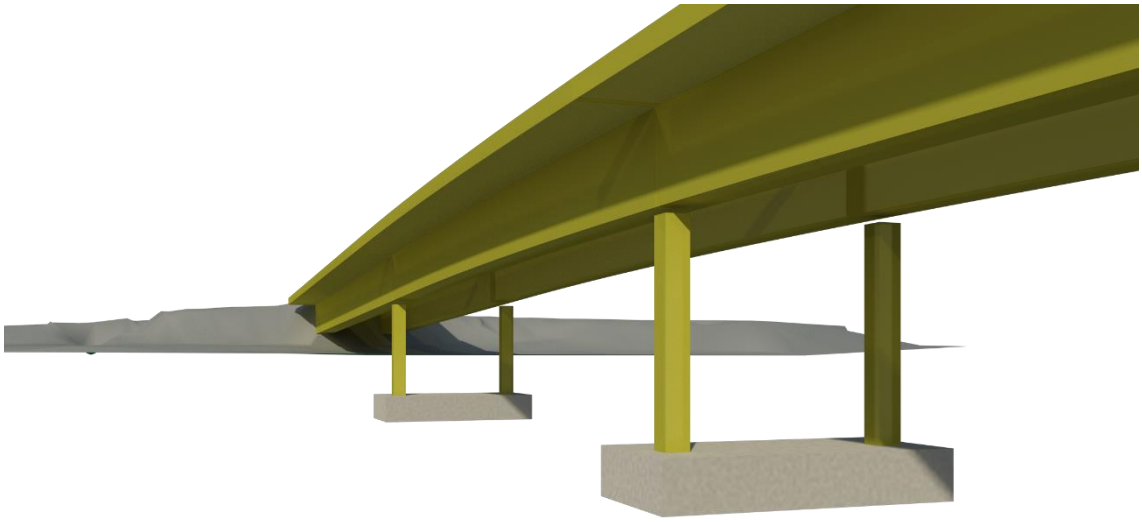


Abbildung 6.43: Revit Rendering 4 Felder, Plattenbalken mit Stütze "einzeln 2x"

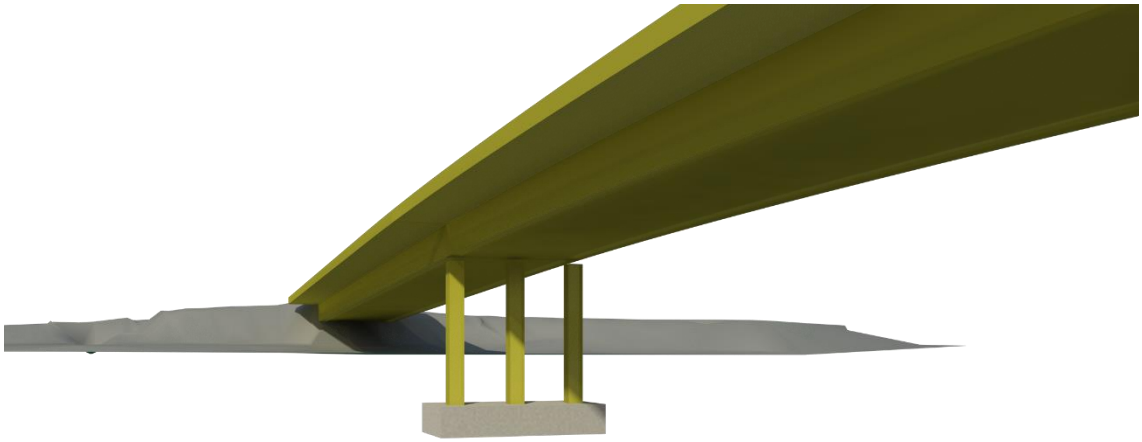


Abbildung 6.44: Revit Rendering 3 Felder, Hohlkasten mit Stütze "einzeln 3x"

7 Zusammenfassung

Ausgehend von der Behandlung verschiedener Wissenstypen, der Klassifizierung von Wissen im Kontext Wissensbasierter Systeme, der Vorstellung der Architektur Wissensbasierter Systeme und Formen der Wissensrepräsentationen und -modellierung wurde in dieser Arbeit die Anwendbarkeit von Knowledge-based Engineering für frühe Phasen des Brückenentwurfs untersucht.

Durch eine Literaturrecherche konnten die wichtigsten methodischen Grundlagen für die Entwicklung Wissensbasierter Systeme beziehungsweise KBE, nämlich KADS und MOKA, erfasst und ihre Funktionsweisen erklärt werden.

Für die Entwicklung des Prototypen wurden die grundlegenden Merkmale von Knowledge-based Engineering aufgezeigt. Neben einer Definition und der Entwicklung von KBE wurde ein umfangreicher Überblick zur Literatur rund um KBE gegeben. Weiter wurden verschiedene Typen von KBE Systemen vorgestellt und grundsätzliche Vor- und Nachteile der KBE Anwendung diskutiert. Außerdem wurden die kommerziellen KBE Systeme Knowledge Fusion und Autodesk Inventor Automation Professional auf ihre Eignung für die Anwendung im Infrastrukturbau untersucht.

In Hinblick auf die Implementierung eines Prototyps wurde der Entwurfsprozess in der Brückenkonstruktion, Tragsysteme in Längs- und Querrichtung und wichtige Regelwerke des Brückenbaus vorgestellt. Es wurden Regeln des Brückenentwurfs identifiziert und in Entscheidungsbäumen nach Bauteilen geordnet visualisiert. Der Stand der Technik in der Konstruktion von Brücken wurde ebenfalls aufgezeigt.

Im Rahmen der Arbeit wurde ein Prototyp auf Basis von Dynamo und Autodesk Revit implementiert. Es wurde die prinzipielle Anwendbarkeit und das Potential und die Machbarkeit von KBE für Fragen der Infrastrukturplanung aufgezeigt, wenngleich Erweiterungen für eine praktikable Nutzung der Anwendung notwendig sind.

8 Ausblick

Ausgehend von der vorliegenden Arbeit, ergeben sich nachfolgend erläuterte Forschungs- und Entwicklungsansätze zur Anwendung von Knowledge-based Engineering im Infrastrukturbau.

Zunächst gilt es den vorgestellten **Prototyp** weiter zu **verbessern**. Es wurden im Rahmen dieser Arbeit einige Entwurfsregeln implementiert, dennoch ist für eine praktikable Anwendung die Umsetzung weiterer Regeln notwendig. Dazu zählen die Implementierung weiterer Entwurfsregeln (zum Beispiel Einhaltung Lichtraumprofil, Ausrichtung der Pfeiler), die Modellierung komplexerer Brücken mit unterschiedlichen Tragwerkssysteme, Querneigungsänderungen entlang der Achse, Abbildung von Spannsystemen und Spanngliedführung.

Eine **Quantifizierung** des zeitlichen und finanziellen Einsparpotentials durch einen Vergleich mit einem realen Projekt, zum Beispiel anhand des BIM Forschungsprojekts Filstalbrücke oder Projekten der Industriepartner, sowohl während der Planung als auch des Bauprojekts selbst.

Darüber hinaus ist die Erweiterung des Prototypen für die Planung von Tunneln denkbar. Besonderheit in der **Tunnelplanung** ist hierbei die starke Abhängigkeit der Entwurfsparameter vom anstehenden Gebirge innerhalb der Entwurfsszenarien. Die Verwendung derselben Module wie für die Anwendung im Brückenbau sollte die generische Verwendung des Prototyps belegen.

Interessant ist weiter die Untersuchung einer **bi-direktionalen Kopplung** mit anderen Softwaresystemen. Beispiele wären die Überprüfung eines ausreichenden Durchflussquerschnittes unter einer geplanten Brücke für Starkregenereignisse durch Kopplung mit einer Computational Fluid Dynamik (CFD) Anwendung. Eine andere Kopplung wäre die Überprüfung der aerodynamischen Eigenschaften hoher Talbrücken mittels CFD. Weitere Kopplungen mit branchentypischen Anwendungen wie Stabwerks-, FEM- oder Kalkulationsanwendungen sind ebenfalls anzustreben. Eine wichtige Fragestellung ist dabei diejenige der sinnvollen Ausgestaltung der Verknüpfung zwischen KBE und den Anwendungen, um eine instantane Nutzung, also einen „Live-Touch“, zu gewährleisten. Zudem sollte die

Kopplung so generisch aufgebaut sein, um grundsätzlich jedes Programm daran anbinden zu können.

Eine Kopplung anderer Art wäre die Kombination von Methoden der **Highway Alignment Optimization** (HAO) mit Methoden des KBE für die großräumige Trassenplanung. Dies würde die Bewertung großräumiger Trassen auf ein deutlich detaillierteres Level heben. Derzeit werden Trassen bezüglich der Ingenieurbauwerke nur anhand sehr grober Parameter wie Streckenanteil der Bauwerke an der Gesamttrassenlänge, Länge des größten Einzelbauwerks und andere bewertet.

Ein weiterer Punkt ist die **Verbesserung** der **Graph**-basierten Modellierung eines KBE Systems. Für den Prototypen wurden wie beschrieben eigens Knoten per Code entwickelt. Zukünftig sollte der Nutzer selbst diese Regeln oder die Regelgrenzen definieren können. Hierzu wäre eine Verknüpfung von deklarativen Programmiersprachen mit einem hierarchischen Graphen zu untersuchen.

Die Nutzung verschiedener **Level of Detail** ist eine weitere mögliche Erweiterung. Dabei ist zu klären, zu welchem Zeitpunkt des Entwurfs welche Detailtiefe in der Konstruktion notwendig ist und wie dies die Ausführung der Regeln beeinflusst.

Außerdem sind für den Aufbau einer zentralen Wissensdatenbank zur Verwaltung der Regeln eines KBE Systems Strategien zu entwickeln. Denkbar ist beispielsweise die Nutzung von **NoSql-Datenbanken** wie MongoDB zur intuitiven Speicherung deklarativer Regeln. Möglicherweise sind die Merkmale von NoSql-Datenbanken bestens geeignet, deklarative Regeln zu speichern und auszuwerten.

Im Bauingenieurwesen liegen Regeln häufig unscharf vor. So sind exakte Aussagen über den Baugrund nur an wenigen explizit untersuchten Stellen möglich. Eine regelbasierte Auswertung eines Baugrundmodells kann also nur sehr unscharfe Antworten liefern. In diesem Zusammenhang hat sich die Anwendung der **Fuzzylogik** bewährt und sollte im Zusammenspiel mit KBE näher untersucht werden.

Wie im Kapitel 6.1 beschrieben, ist der Brückenentwurf abhängig von verschiedenen konkurrierenden Kriterien wie Funktionalität, Herstellung, Wirtschaftlichkeit, Ästhetik und Nachhaltigkeit. An dieser Stelle erscheint der Einsatz von KBE mit Elementen der **Multikriteriellen Optimierung** zur Optimierung einzelner Bauteile geeignet.

Für den Austausch der mit Hilfe des KBE Systems generierten Modelle, sollten entsprechende Schnittstellen angeboten werden. Sinnvoll sind hierbei die Ifc Derivate *IfcAlignment* (Amann et al. 2013; Singer und Amann 2014), *IfcRoad* (Lee und Kim 2011), *IfcBridge* (Ji et al. 2012) und *IfcTunnel* (Jubierre und Borrmann 2014).

All die aufgeführten Punkte unterstreichen nochmals das große Potential für die Anwendung von KBE im Infrastrukturbau.

9 Literaturverzeichnis

Abdullah, M. S.; Kimble, C.; Paige, R.; Benest, I.; Evans, A. (2005): Developing a UML Profile for Modelling Knowledge-Based Systems. In: David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell et al. (Hg.): Model Driven Architecture, Bd. 3599. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), S. 220–233.

Adeli, H.; Balasubramanyam, K. V. (1988): A Knowledge-Based System for Design of Bridge Trusses. In: *J. Comput. Civ. Eng.* 2 (1), S. 1–20. DOI: 10.1061/(ASCE)0887-3801(1988)2:1(1).

Aksamija, Ajla; Ali, M. (2008): Information Technology and Architectural Practice: Knowledge Modeling Approach and BIM. Online verfügbar unter http://ca.perkinswill.com/files/Information%20Technology%20and%20Architectural%20Practice.pdf.

Amadori, K. (2012): Geometry Based Design Automation: Applied to Aircraft Modelling and Optimization. Doctoral thesis. Linköping University. The Institute of Technology. Online verfügbar unter http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-73109.

Amadori, K.; Tarkian, M.; Ölvander, J.; Krus, P. (2012): Flexible and robust CAD models for design automation. In: *Advanced Engineering Informatics* 26 (2), S. 180–195. DOI: 10.1016/j.aei.2012.01.004.

Amann, J. (2014): TUM Open Infra Platform. Online verfügbar unter https://www.cms.bgu.tum.de/de/forschung/projekte/31-forschung/projekte/397-tum-open-infra-platform, zuletzt geprüft am 23.10.2014.

Amann, J.; Borrmann, A.; Hegemann, F.; Jubierre, J. R.; Flurl, M.; Koch, C.; König, M. (2013): A Refined Product Model for Shield Tunnels Based on a Generalized Approach for Alignment Representation. In: *Proc. of the ICCBEI 2013*.

Ammar-Khodja, S.; Perry, N.; Bernard, A. (2008): Processing Knowledge to Support Knowledge-based Engineering Systems Specification. In: *Concurrent Engineering* 16 (1), S. 89–101. DOI: 10.1177/1063293X07084642.

Autodesk (2009): Revit Architecture 2010 Dokumentation. Online verfügbar unter docs.autodesk.com/REVIT/2010/ENU/Revit Architecture 2010 Users Guide/RAC/index.html, zuletzt geprüft am 28.10.2014.

Autodesk (2014a): Infraworks360. Online verfügbar unter http://www.autodesk.de/products/infraworks-family/overview, zuletzt geprüft am 18.11.2014.

Autodesk (2014b): Revit. Online verfügbar unter http://www.autodesk.de/products/revit-family/overview, zuletzt geprüft am 28.10.2014.

Autodesk (2014c): Revit Bridge Modeler. Online verfügbar unter https://apps.exchange.autodesk.com/RVT/en/Detail/Index?id=appstore.exchange.autodesk.com:civilstructuresforautodeskrevit2014:en, zuletzt geprüft am 05.11.2014.

Barreiro, J.; Martínez, S.; Cuesta, E.; Álvarez, B.; Fernández, P. (2009): High level diagrams for identification of knowledge as a basis for a KBE implementation of inspection planning process. In: *III Manufacturing Engineering Society International Congress (CISIF-MESIC)*.

BASt (2009): Richtzeichnungen für Ingenieurbauten - Flügelausbildung.

Bermell-Garcia, P.; Ip-Shing, F. (2002): A KBE System for the design of wind tunnel models using reusable knowledge components.

BMVI (2014): Richtzeichnungen für Ingenieurbauten. Online verfügbar unter http://www.bmvi.de/SharedDocs/DE/Artikel/StB/richtzeichnungen-fuer-ingenieurbauten.html?nn=36114, zuletzt geprüft am 24.11.2014.

Brimble, R.; Sellini, F. (2000): The MOKA Modelling Language. In: Rose Dieng und Olivier Corby (Hg.): Knowledge Engineering and Knowledge Management Methods, Models, and Tools, Bd. 1937. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), S. 49–56.

Calkins, D. E. (2000): Knowledge-Based Engineering (KBE) Design Methodology at the Undergraduate and Graduate Levels. In: *International Journal of Engineering Education, Vol. 16, No. 1,*, S. 21–38.

Cenker, C. (2012): KMB-PR Praktikum aus Knowledge Management im Bildungsbereich. Online verfügbar unter https://cewebs.cs.univie.ac.at/mid-kmb/ss12/index.php?m=D&t=wikis&c=show&CEWebS_c=a8502340&CEWebS_what=Aufgabe~32~KMB~32~PR01, zuletzt geprüft am 19.10.2014.

Chandrasekaran, B. (1983): Expert Systems: Matching Techniques to Tasks. Annual Report for Research on Distributed Knowledge Base Systems for Diagnosis and Information Retrieval. Air Force Office of Scientific Research. Washington, DC.

Chapman, C.B. (2014): Knowledge Based Engineering. Online verfügbar unter http://www.bcu.ac.uk/engineering-design-and-manufacturing-systems/research/the-kbe-lab, zuletzt geprüft am 10.11.2014.

Chapman, C.B.; Pinfold, M. (1999): Design engineering—a need to rethink the solution using knowledge based engineering. In: *Knowledge-Based Systems* 12 (5-6), S. 257–267. DOI: 10.1016/S0950-7051(99)00013-1.

Chapman, C.B.; Pinfold, M. (2001): The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. In: *Advances in Engineering Software* 32 (12), S. 903–912. DOI: 10.1016/S0965-9978(01)00041-2.

Chassiakos, A. P.; Vagiotas, P.; Theodorakopoulos, D. D. (2005): A knowledge-based system for maintenance planning of highway concrete bridges. In: *Advances in Engineering Software* 36 (11-12), S. 740–749. DOI: 10.1016/j.advengsoft.2005.03.020.

Chiciudean, T.G.; La Rocca, G.; van Tooren, M. (2008): A Knowledge Based Engineering Approach to Support Automatic Design of Wind Turbine Blades. In: Houten, Fred J. A. M. van (Hg.): CIRP Design Conference 2008 ; April 7 - 9, 2008, Enschede. CIRP Design Conference. Enschede: Laboratory of Design, Production and Management, Faculty of Engineering Technology, Univ. of Twente.

Cooper, S.; Fan, I.; Li, G. (1999): Achieving Competitive Advantage through Knowledge-Based Engineering. A BEST PRACTICE GUIDE. Cranfield University.

Corallo, A.; Laubacher, R.; Margherita, A.; Turrisi, G. (2009): Enhancing product development through knowledge-based engineering (KBE): A case study in the aerospace industry. In: *Journal of Manufacturing Technology Management* 20 (8), S. 1070–1083. DOI: 10.1108/17410380910997218.

Curran, R.; Verhagen, W. J.C.; van Tooren, M. J.L.; van der Laan, T. H. (2010): A multidisciplinary implementation methodology for knowledge based engineering: KNOMAD. In: *Expert Systems with Applications* 37 (11), S. 7336–7350. DOI: 10.1016/j.eswa.2010.04.027.

Dassault Systemes (2002): Dassault Systemes Acquires KTI. Online verfügbar unter http://www.3ds.com/press-releases/single/dassault-systemes-acquires-kti/, zuletzt geprüft am 18.11.2014.

Duden Online (2013): wissen. Online verfügbar unter http://www.duden.de/rechtschreibung/wissen, zuletzt geprüft am 19.11.2014.

Dynamo (2014): DynamoBIM. Open source graphical programming for design. Online verfügbar unter http://dynamobim.org/, zuletzt geprüft am 17.11.2014.

Evdorides, H. (1993): A Prototype Knowledge-based System for Pavement Analysis. University of Birmingham, Birmingham. School of Civil Engineering. Online verfügbar unter http://etheses.bham.ac.uk/3580/1/Evdorides94PhD.pdf.

Ferrada, X.; Serpell, A. (2013): Using organizational knowledge for the selection of construction methods. In: *International Journal of Managing Projects in Business* 6 (3), S. 604–614. DOI: 10.1108/IJMPB-10-2012-0061.

Ferrada, X.; Serpell, A. (2014): Selection of Construction Methods for Construction Projects: A Knowledge Problem. In: *J. Constr. Eng. Manage.* 140 (4), S. B4014002. DOI: 10.1061/(ASCE)CO.1943-7862.0000715.

Fischer, O. (2013): Vorlesungsskript Massivbrücken TU München.

Gartner (2014): Gartner Hype Cycle. Online verfügbar unter http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp, zuletzt geprüft am 22.07.2014.

Genworks (2013): Genworks International. Online verfügbar unter http://www.genworks.com, zuletzt geprüft am 09.10.2014.

Hayes-Roth, F.; Jacobstein, N. (1994): The state of knowledge-based systems. In: *Commun. ACM* 37 (3), S. 26–39. DOI: 10.1145/175247.175249.

Holst, Ralph (2013): Brücken aus Stahlbeton und Spannbeton. Entwurf, Konstruktion und Berechnung. 6., akt. Aufl. Berlin: Ernst.

Ji, Y.; Obergrießer, M.; Borrmann, A. (2010): Prototypische Entwicklung IFC-Bridge-basierter Anwendung in parametrischen CAD-Systemen. In: *Forum Bauinformatik 2010*.

Ji, Yang; Borrmann, André; Beetz, Jakob; Obergrießer, Mathias (2012): Exchange of parametric bridge models using a neutral data format. In: *Journal of Computing in Civil Engineering* 27 (6), S. 593–606.

Jubierre, J. R.; Borrmann, A. (2014): A multi-scale product model for shield tunnels based on the Industry Foundation Classes. Technische Universität München.

Kalavrytinos, C.; Sievertsen, O. I. (2014): A knowledge-based engineering approach for offshore process plant design. In: *Engineering, Technology and Innovation (ICE), 2014 International ICE Conference*, S. 1–6. DOI: 10.1109/ICE.2014.6871540.

Kok, R. (2005): NX Knowledge Fusion ICE. Integrated Development Enviroment for KF development, zuletzt geprüft am 26.11.2014.

La Rocca, G. (2011): Knowledge based engineering techniques to support aircraft design and optimization. Dissertation. TU Delft. Aerospace Design, Integration & Operations. Online verfügbar unter http://repository.tudelft.nl/assets/uuid:45ed17b3-4743-4adc-bd65-65dd203e4a09/PhD-GLaRocca-2011-v2.pdf, zuletzt geprüft am 17.07.2014.

La Rocca, G. (2012): Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. In: *Advanced Engineering Informatics* 26 (2), S. 159–179. DOI: 10.1016/j.aei.2012.02.002.

La Rocca, G.; van Tooren, M. (2007): A Knowledge Based Engineering Approach to Support Automatic Generation of FE Models in Aircraft Design. DOI: 10.2514/6.2007-967.

La Rocca, G.; Van Tooren, L.; Michel, J. (2005): Development of Design and Engineering Engines to Support Multidisciplinary Design and Analysis of Aircraft, S. 139–154.

La Rocca, G.; Van Tooren, L.; Michel, J. (2009): Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization. In: *Journal of Aircraft* 46 (6), S. 1875–1885. DOI: 10.2514/1.39028.

Lee, S.-H.; Kim, B.-G. (2011): IFC Extension for Road Structures and Digital Modeling. In: *Procedia Engineering* 14, S. 1037–1042. DOI: 10.1016/j.proeng.2011.07.130.

Liebig, T. (2008): Wissensmodellierung und wissensbasierte Systeme. Online verfügbar unter http://www.informatik.uni-ulm.de/ki/Edu/Vorlesungen/Wissensmodellierung/SS08/Folien/Wissensmodellierung08.pdf, zuletzt geprüft am 04.09.2014.

Liese, Harald (2003): Wissensbasierte 3D-CAD-Repräsentation. Dissertation. TU Darmstadt, Darmstadt. Fachbereich Maschinenbau.

Liu, F.; Jallow, A. K.; Anumba, C. J.; Wu, D. (2013): Building Knowledge Model: Integrating Knowledge in BIM. Online verfügbar unter faculty.ist.psu.edu/wu/papers/BIM-CIB-W78.pdf, zuletzt geprüft am 12.06.2014.

Lovett, P.J.; Ingram, A.; Bancroft, C.N. (2000): Knowledge-based engineering for SMEs — a methodology. In: *Journal of Materials Processing Technology* 107 (1-3), S. 384–389. DOI: 10.1016/S0924-0136(00)00728-7.

McGoey, P. J. (2011): A Hitch-hikers Guide to: Knowledge Based Engineering in Aerospace (& other Industries) Do not Panic!! Definition, History, Value, Technology, and Resources.

Motamedi, A.; Hammad, A.; Asen, Y. (2014): Knowledge-assisted BIM-based visual analytics for failure root cause detection in facilities management. In: *Automation in Construction* 43, S. 73–83. DOI: 10.1016/j.autcon.2014.03.012.

Motawa, I.; Almarshad, A. (2013): A knowledge-based BIM system for building maintenance. In: *Automation in Construction* 29, S. 173–182. DOI: 10.1016/j.autcon.2012.09.008.

Pinfold, Martyn; Chapman, Craig (2001): The application of KBE techniques to the FE model creation of an automotive body structure. In: *Computers in Industry* 44 (1), S. 1–10. DOI: 10.1016/S0166-3615(00)00079-8.

Rasdorf, W. J. (1985): Perspectives on Knowledge in Engineering Design. In: Proceedings of the 1985 ASME International Computers in Engineering Conference and Exhibition, August 4-8, 1985, S. 249–253.

Rezgui, Y.; Hopfe, C. J.; Vorakulpipat, C. (2010): Generations of knowledge management in the architecture, engineering and construction industry: An evolutionary perspective. In: *Advanced Engineering Informatics* 24 (2), S. 219–228. DOI: 10.1016/j.aei.2009.12.001.

Sainter, P.; Oldham, K.; Larkin, A. (2000): Achieving benefits from knowledge-based engineering systems in the longer term as well as in the short term. In: *Proceedings International Conference on Concurrent Enterprising, Toulouse, France*.

Sandberg, M. (2003): Knowledge Based Engineering - In Product Development. Online verfügbar unter http://epubl.ltu.se/1402-1536/2003/05/LTU-TR-0305-SE.pdf, zuletzt geprüft am 08.05.2014.

Schreiber, G.; Akkermans, H.; Anjewieren, A,; de Hoog, R.; Nigel Shadbolt, N.; Van de Velde, W.; Wielinga, B. (2000): Knowledge engineering and management. The CommonKADS methodology. Cambridge, Mass: MIT Press.

Schreiber, G.; Wielinga, B.; Breuker, J. (1993): KADS. A principled approach to knowledge-based system development. London, San Diego: Academic Press (Knowledge-based systems, v. 11).

Siemens PLM Software (2008): NX™ Knowledge Fusion Knowledge Based Engineering. Online verfügbar unter http://www.pbu-cad.de/downloads/category/118-nx-knowledge-fusion-wissensbasierte-automatisierung/20-nx-broschueren-und-datenblaetter, zuletzt geprüft am 17.04.2014.

Siemens PLM Software (2010): NX Programming and Customization. Online verfügbar unter http://www.endesin.com/SiteAssets/Resources/NXProgramming/NXCommonAPI.pdf, zuletzt geprüft am 05.10.2014.

Singer, D.; Amann, J. (2014): Erweiterung von IFC Alignment um Straßenquerschnitte. In: *Proc. of the 26th Forum Bauinformatik*.

Skarka, Wojciech (2007): Application of MOKA methodology in generative model creation using CATIA. In: *Engineering Applications of Artificial Intelligence* 20 (5), S. 677–690. DOI: 10.1016/j.engappai.2006.11.019.

Speel, P. H.; Schreiber, A. Th; van Joolingen, W.; van Heijst, G.; Beijer, G. J. (2001): Conceptual modelling for knowledge-based systems. In: *Encyclopedia of Computer Science and Technology* 44, S. 107–132.

Standfuß, F. (1995): Gestaltung von Brücken an Bundesfernstrassen. In: *Beton- und Stahlbetonbau* 90 (4), S. 91–98. DOI: 10.1002/best.199500120.

Stokes, M. (2001): Managing engineering knowledge. MOKA: methodology for knowledge based engineering applications. London: Professional Engineering Pub.

TechnoSoft (2014): TechnoSoft. Online verfügbar unter http://www.technosoft.com/, zuletzt geprüft am 06.11.2014.

Tripathi, K. P. (2011): A Review on Knowledge-based Expert System: Concept and Architecture. In: *IJCA Special Issue on Artificial Intelligence Techniques - Novel Approaches & Practical Applications* (4), S. 21–25.

Unigraphics (2001): Knowledge Fusion for Designers. Online verfügbar unter http://de.scribd.com/doc/6539345/Knowledge-Fusion, zuletzt geprüft am 06.11.2014.

Verhagen, W.; Bermell-Garcia, P.; van Dijk, R.; Curran, R. (2012): A critical review of Knowledge-Based Engineering: An identification of research challenges. In: *Advanced Engineering Informatics* 26 (1), S. 5–15. DOI: 10.1016/j.aei.2011.06.004.

Voss, E.; Overend, M. (2012): A Tool that Combines Building Information Modeling and Knowledge Based Engineering to Assess Façade Manufacturability. Hg. v. University of Cambridge, Department of Engineering. Online verfügbar unter http://www-g.eng.cam.ac.uk/gft/media/PG%20projects/Eleanor%27s%20publication/120330%20ABS%20EVOSS.pdf.

Weih, H.-P.; Schü, J.; Calmet, J. (1994): CommonKADS and cooperating knowledge based systems. In: Proceedings of the 4th KADS User Meeting, Bonn 1994. Ed.: GMD. 1994. S. 1-17. Bonn, S. 1–17.

Woo, J.-H.; Clayton, M.; Johnson, R.; Flores, B.; Ellis, C. (2004): Dynamic Knowledge Map: reusing experts' tacit knowledge in the AEC industry. In: *Automation in Construction* 13 (2), S. 203–207. DOI: 10.1016/j.autcon.2003.09.003.

Anhang

Compact Disc

Auf der beigefügten CD befindet sich folgender Inhalt:

- Die schriftliche Ausarbeitung der Thesis als .pdf Datei
- Das Revit Projekt und die Revit Familien
- Die Projektdateien des Dynamo Projekts
- Der Quellcode der KBE Knoten sowie der *Infrastructure.NET* API

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht.

.....

München, 28.11.2014 Singer Dominic