Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme

Diplomarbeit

Entwicklung eines Datenmodells zur Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung

Verfasser:

Michael Stoye

11. April 2011

Betreuer:

Prof. Dr. rer. nat. habil. Gunter Saake
Dipl.-Inform. Stephan Vornholt
Dipl.-Inf. Ingolf Geist

Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Postfach 4120, D-39016 Magdeburg

Stoye, Michael:
Entwicklung eines Datenmodells zur Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung
Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, 2011.

Danksagung

An dieser Stelle möchte ich mich zuerst bei all denen bedanken, die mich während meines Studiums begleitet haben. Dieser Dank gilt den Dozenten und Dozentinnen, die mich unterrichtet haben, sowie Kommilitonen und Freunden für jegliche Hilfe und Unterstützung in dieser Zeit. Mein ganz besonderer Dank gilt:

- Stephan Vornholt für die Betreuung dieser Diplomarbeit, die zahlreichen Konsultationen, die hilfreiche Ratschläge und Diskussionen sowie das Korrekturlesen dieser Arbeit
- Prof. Dr. rer. nat. habil. Gunter Saake für die Übernahme des Erstgutachtens sowie das Wecken meines Interesses für Datenbanken
- Prof. Dr.-Ing. habil. Georg Paul für die Übernahme des Zweitgutachtens, die Betreuung meiner Studienarbeit sowie für die Unterstützung und Ratschläge während meines Studiums
- Ingolf Geist, Alexander Pelzer und meinen Großeltern für das Korrekturlesen dieser Arbeit
- Dr.-Ing. Martin Endig für die Betreuung und Förderung während meiner Tätigkeit als Hilfswissenschaftler am Fraunhofer-Institut für Fabrikbetrieb und -automatisierung
- meinen Kommilitonen Alexander Pelzer und Christian Schulz für die gegenseitige Unterstützung während des gesamten Studiums
- meiner Familie und meiner Verlobten Priscila für die Unterstützung und den Rückhalt während meines Studiums und insbesondere während des Schreibens dieser Arbeit

Inhaltsverzeichnis

Abbildungsverzeichnis **VII**

Tabellenverzeichnis **IX**

Abkürzungsverzeichnis **XI**

1 Einleitung **1**
1.1 Problemdefinition . . . 1
1.2 Ziele der Arbeit . . . 3
1.3 Gliederung der Arbeit . . . 4

2 Grundlagen **5**
2.1 Grundlegende Begriffe . . . 5
2.1.1 Produkt . . . 5
2.1.2 Prozess . . . 5
2.1.3 Produktentwicklung . . . 6
2.1.4 Produktlebenszyklus . . . 7
2.1.5 Mechatronik . . . 8
2.2 Virtuelle Produktentwicklung . . . 11
2.2.1 Virtueller Prototyp . . . 12
2.2.2 Virtual Reality . . . 12
2.2.3 Rapid Prototyping . . . 13
2.2.4 CAx-Systeme . . . 13
2.3 Dateibasierter Datenaustausch . . . 15
2.3.1 Dateibasierte Schnittstellen . . . 15
2.3.2 Systemspezifische und systemneutrale Schnittstellen . . . 16
2.3.3 Neutrale und native Dateiformate . . . 17
2.4 Prozessunterstützung in der Produktentwicklung . . . 19
2.4.1 Parallelisierung von Prozessen . . . 19
2.4.2 Workflowmanagement-Systeme . . . 20
2.5 Zusammenfassung . . . 21

3 Analyse **23**
3.1 Datenaustausch bei einem Entwicklungsprozess von Industrierobotern . . . 23
3.1.1 Mehrkörpersysteme . . . 24
3.1.2 Mehrkörpersystem-Simulationen mit Dymola . . . 25
3.1.3 Ablauf des Prozesses . . . 26
3.1.4 Analyse der erzeugten Daten . . . 32
3.1.5 Schlussfolgerungen . . . 32
3.2 Dateiformate zum Austausch von CAD-Daten . . . 35
3.2.1 Initial Graphics Exchange Specification (IGES) . . . 36
3.2.2 Standard for the Exchange of Product Model Data (STEP) . . . 36

3.2.3 Jupiter Tessellation (JT) . . . 40
3.2.4 Drawing Interchange Format (DXF) . . . 42
3.2.5 Surface Tessellation Language (STL) . . . 42
3.2.6 Virtual Reality Modeling Language (VRML) . . . 43
3.2.7 Schlussfolgerungen . . . 43
3.3 Nachvollziehbarkeit des Datenaustauschs . . . 45
3.3.1 Export . . . 46
3.3.2 Import . . . 47
3.3.3 Schlussfolgerungen . . . 48
3.4 Zusammenfassung . . . 48

4 Konzept **51**
4.1 Systemverwaltung . . . 51
4.1.1 Anforderungen . . . 52
4.1.2 Datenmodell . . . 54
4.1.3 Anwendungsfälle . . . 56
4.1.4 Vorgehensweise zum Anlegen von Daten . . . 59
4.2 Prozessverwaltung . . . 63
4.2.1 Anforderungen . . . 63
4.2.2 Datenmodell . . . 64
4.2.3 Anwendungsfälle . . . 71
4.3 Zusammenfassung . . . 76

5 Implementierung **79**
5.1 Prototyp . . . 79
5.1.1 Architektur . . . 79
5.1.2 Google Web Toolkit . . . 80
5.2 Erweiterung der Datenhaltung . . . 82
5.2.1 Logischer Entwurf . . . 82
5.2.2 Datendefinition . . . 85
5.3 Erweiterung der Präsentation und Anwendungslogik . . . 87
5.3.1 Entwurfsmuster »Model-View-Presenter« . . . 87
5.3.2 Realisierung der Erweiterung . . . 89
5.4 Zusammenfassung . . . 91

6 Zusammenfassung und Ausblick **93**

Quellenverzeichnis **97**

Anhang **105**
A Anwendungsfallbeschreibungen . . . 105
B Weitere Aktivitätsdiagramme des Konzeptes . . . 108
B.1 Aktivitätsdiagramme zum Teilkonzept »Systemverwaltung« . . . 108
B.2 Aktivitätsdiagramme zum Teilkonzept »Prozessverwaltung« . . . 112
C SQL-Skript . . . 113

Abbildungsverzeichnis

1.1 Möglichkeiten zur Interoperabilität von CAx-Systemen (angelehnt an [Stekolschik, 2007, S. 1 und 2]) 2

2.1 Produktlebenszyklus (angelehnt an [VDI 2221, 1993, S. 8]) 7
2.2 Grundstruktur eines mechatronischen Systems (angelehnt an [VDI 2206, 2004, S. 14]) 9
2.3 Vorgehensmodell nach VDI-Richtlinie 2206 [VDI 2206, 2004, S. 29] 10
2.4 Möglichkeiten zur Realisierung des Datenaustauschs zwischen CAx-Systemen (angelehnt an [Grabowski u. Anderl, 1990, S. 34]) 17
2.5 Datenverlust beim Datenaustausch (angelehnt an [Reim et al., 2007, S. 3]) 18
2.6 Möglichkeiten zur Parallelisierung von Prozessen (angelehnt an [Anderl u. Trippner, 2000, S. 14]) 19

3.1 Modellvorstellung eines Mehrkörpersystems [Vajna et al., 2009, S. 287] . . 24
3.2 Modellierung in Dymola am Beispiel eines Antriebstrangs (angelehnt an [Otter u. Elmqvist, 2000, S. 3]) 26
3.3 Beim Prozess am Fraunhofer IFF eingesetzte CAx-Systeme 27
3.4 Aktivitätsdiagramm »CAD-Daten für RobotMax vorbereiten« 29
3.5 Aktivitätsdiagramm »mechatronisches MKS-Modell in RobotMax erstellen« 31
3.6 Aktivitätsdiagramm »MKS-Simulation durchführen« 31
3.7 Klassendiagramm »Beim Prozess entstehende Daten« 33
3.8 Aufbau von STEP (angelehnt an [Dyla, 2002, S. 24]) 37
3.9 Produktstruktur und 3D-Modell eines vereinfachten Industrieroboters . . . 45
3.10 Ablauf eines Datenaustauschprozesses 49

4.1 Klassendiagramm »Systemverwaltung« 53
4.2 Zusammenhang zwischen einer Schnittstelle und Austauschvorgängen . . . 55
4.3 Anwendungsfalldiagramm »Systemverwaltung« 57
4.4 Aktivitätsdiagramm zum Anwendungsfall »Export- oder Importmöglichkeit einer CAx-System_Version definieren« 58
4.5 Aktivitätsdiagramm zum Anwendungsfall »Schnittstelle definieren« 59
4.6 Vorgehensweise zum Anlegen der Daten in der »Systemverwaltung« 61
4.7 Objektdiagramm »Systemverwaltung« 61
4.8 Klassendiagramm »Prozessverwaltung« 65
4.9 Alternative Beschreibung für Vorgänger eines Prozessschrittes 67
4.10 Beispiel für eine Prozesskette 68
4.11 Beispiel für eine Prozessketteninstanz 69
4.12 Anwendungsfalldiagramm »Prozessverwaltung« 71
4.13 Aktivitätsdiagramm zum Anwendungsfall »Prozesskette anlegen« 72
4.14 Aktivitätsdiagramm zum Anwendungsfall »Prozessschritt anlegen« 73
4.15 Aktivitätsdiagramm zum Anwendungsfall »durchgeführte Prozesskette beschreiben« 74

4.16 Aktivitätsdiagramm zum Anwendungsfall »durchgeführten Austauschvorgang beschreiben« 76
4.17 Klassendiagramm »Gesamtmodell« 78

5.1 Architektur des Prototypen 80
5.2 Relationales Schema der geplanten Datenbank 83
5.3 Auflösung der Assoziationsklasse »Einstellungsmöglichkeiten« 84
5.4 Implementiertes Datenbankschema 88
5.5 Entwurfsmuster »Model-View-Presenter« 89
5.6 Klassendiagramm »Anzeigen aller CAx-Systeme« 90
5.7 Sequenzdiagramm »CAx-Systeme anzeigen« 91
5.8 Anzeigen aller CAx-Systeme im Prototypen 91

B.1 Aktivitätsdiagramm zum Anwendungsfall »CAx-System_Version anlegen« 108
B.2 Aktivitätsdiagramm zum Anwendungsfall »CAx-System anlegen« 109
B.3 Aktivitätsdiagramm zum Anwendungsfall »Dateiformat anlegen« 109
B.4 Aktivitätsdiagramm zum Anwendungsfall »CAx-System_Version deaktivieren« 110
B.5 Aktivitätsdiagramm zum Anwendungsfall »Systemlandschaft anzeigen« . . 110
B.6 Aktivitätsdiagramm zum Anwendungsfall »Export- oder Importmöglichkeiten einer CAx-System_Version anzeigen« 111
B.7 Aktivitätsdiagramm zum Anwendungsfall »Schnittstellen anzeigen« 111
B.8 Aktivitätsdiagramm zum Anwendungsfall »angelegte Prozesskette anzeigen« 112
B.9 Aktivitätsdiagramm zum Anwendungsfall »durchgeführte Prozesskette anzeigen« 112

Tabellenverzeichnis

2.1 Beispiele für CAx-Systeme 13
2.2 Vergleich von systemspezifischen und -neutralen Schnittstellen 17

3.1 Beschreibung der Schnittstellen zwischen den beim Prozess am Fraunhofer IFF eingesetzten CAx-Systemen 27
3.2 Vergleich von STEP und JT 42
3.3 Vergleich von neutralen Dateiformaten 43
3.4 Vergleich der Exportmöglichkeiten von neutralen Dateiformaten in Pro/E . 46
3.5 Vergleich der Importmöglichkeiten von neutralen Dateiformaten in Pro/E . 47

5.1 Verwendete Datentypen von PostgreSQL 8.4 86

A.1 Anwendungsfallbeschreibungen Teilkonzept »Systemverwaltung« 105
A.2 Anwendungsfallbeschreibungen Teilkonzept »Prozessverwaltung« 107

Abkürzungsverzeichnis

AE	Ajax engine
Ajax	Asynchronous JavaScript and XML
AP	Anwendungsprotokoll
API	Application Programming Interface
ASCII	American National Standard Code for Information Interchange
AWF	Anwendungsfall
CAD	Computer Aided Design
CAD*I	CAD-Interfaces
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAP	Computer Aided Planning
CAx	Computer Aided x
CSS	Cascading Style Sheets
DBMS	Datenbankmanagement-System
DIN	Deutsches Institut für Normung e. V.
DXF	Drawing Interchange Format
FEM	Finite-Elemente-Methode
GWT	Google Web Toolkit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IGES	Initial Graphics Exchange Specification
ISO	International Organization for Standardization
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
JT	Jupiter Tessellation
MKS	Mehrkörpersystem

MVP	Model-View-Presenter
NC	Numeric Control
PDDI	Product Definition Data Interface
PDES	Product Data Exchange Specification
PDM	Produktdatenmanagement
PMI	Product Manufacturing Information
Pro/E	Pro/ENGINEER Wildfire
SDAI	Standard Data Access Interface
SET	Standard d'Echange et de Transfert
SQL	Structured Query Language
STEP	Standard for the Exchange of Product Model Data
STL	Surface Tessellation Language
UML	Unified Modeling Language
VDA	Verband der Automobilindustrie e. V.
VDA-FS	Verband der Automobilindustrie - Flächenschnittstelle
VDI	Verband Deutscher Ingenieure e.V.
VP	virtueller Prototyp
VPE	virtuelle Produktentwicklung
VR	Virtual Reality
VRML	Virtual Reality Modeling Language
WFM	Workflowmanagement
XML	Extensible Markup Language

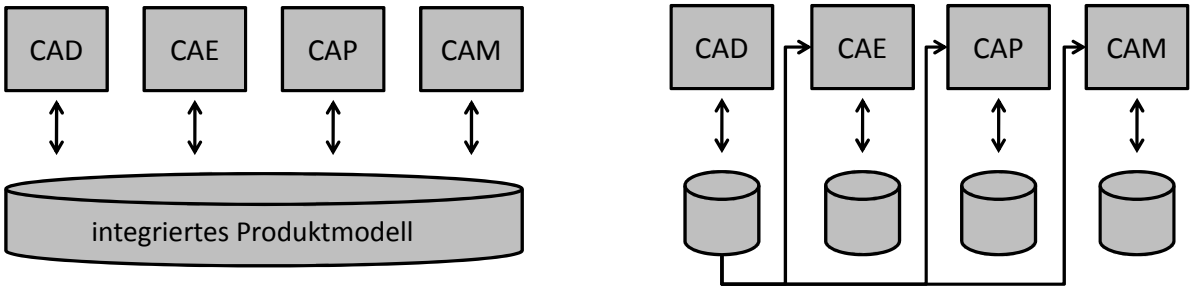
1 Einleitung

Die zunehmende Globalisierung in den letzten Jahrzehnten hat sich neben Bereichen, wie zum Beispiel Umwelt, Politik und Kultur, auch auf die weltweite Wirtschaft ausgewirkt. Einerseits können produzierende Unternehmen heutzutage auf der ganzen Welt ihre Produkte vertreiben und mit anderen Unternehmen zusammenarbeiten. Andererseits stehen sie unter einem hohen Wettbewerbsdruck, da es weltweit konkurrierende Unternehmen gibt. Die Anforderungen des Marktes können sich dabei schnell ändern. Um darauf reagieren zu können, müssen neue oder verbesserte Produkte in möglichst kurzer Zeit entwickelt werden. Dadurch entsteht für die Unternehmen ein hoher Zeit- und Innovationsdruck [Tenbusch, 2002, S. 4]. Durch die ständige Weiterentwicklung nimmt außerdem die Produktkomplexität immer mehr zu [Eigner et al., 2007, S. 582] und entsprechend auch die Komplexität der Fertigungsprozesse und der Entwicklung. Damit Unternehmen am Markt bestehen können, müssen sie wirtschaftlich arbeiten. Faktoren, welche die Wirtschaftlichkeit beeinflussen, sind Entwicklungszeit, Produktqualität und Kosten. Unternehmen müssen versuchen Verbesserungen dieser drei Faktoren zu erzielen, das heißt Produkte in der für den Markt erforderlichen Qualität zu akzeptablen Preisen in möglichst kurzer Zeit zu entwickeln und zu produzieren.

Um den beschriebenen Anforderungen gerecht werden zu können, wurden seit den 70er Jahren immer mehr rechnerunterstützte Systeme und virtuelle Technologien in der Produktentwicklung eingesetzt, so dass ausgehend von der Modellierung der Gestalt eines Produktes immer mehr Eigenschaften bis hin zum Verhalten eines Produktes rechnerintern abgebildet werden können [Abramovici, 2002]. Als Ergebnis dieser Entwicklung ist es seit Mitte der 90er Jahre möglich, eine durchgehende Rechnerunterstützung der Produktentwicklung zu realisieren, die auch als »virtuelle Produktentwicklung« bezeichnet wird [Eigner, 2009, S. 250]. In diesem Zusammenhang werden viele Software-Systeme eingesetzt, die zusammenfassend als »Computer Aided x (CAx)-Systeme« bezeichnet werden. Das »x« steht dabei als Platzhalter für eine Vielzahl von Akronymen, wie zum Beispiel CAD, CAE, CAP und CAM.

1.1 Problemdefinition

In den Anfängen der Rechnerunterstützung wurden die CAx-Systeme isoliert voneinander eingesetzt und daher auch als »Insellösungen« bezeichnet. Von einem CAx-System erzeugte Daten konnten nur durch eine erneute Eingabe in ein anderes CAx-System übertragen werden. Diese manuelle Übertragung der Daten war fehleranfällig und zeitaufwändig. Daher war das Ziel, Daten digital zwischen CAx-Systemen auszutauschen. Von der Wissenschaft wurde eine Integration von CAx-Systemen auf der Grundlage eines sogenannten »integrierten Produktmodells« propagiert (siehe zum Beispiel [Grabowski et al., 1993]). Dieser Ansatz ist in Abbildung 1.1 (a) dargestellt. Das Produktmodell bildet eine einheitliche Datenbasis, die aus verschiedenen integrierten Teilmodellen, welche auch als »Partialmodelle« bezeichnet werden, zusammengesetzt ist. Nach [Stekolschik, 2007, S. 1 f.] hat



(a) integriertes Produktmodell (b) dateibasierter Datenaustausch

Abbildung 1.1: Möglichkeiten zur Interoperabilität von CAx-Systemen (angelehnt an [Stekolschik, 2007, S. 1 und 2])

sich dieser Ansatz aber auf Grund der fehlenden Akzeptanz der CAx-System-Hersteller nicht durchgesetzt. Dadurch ist die Integration von CAx-Systemen verschiedener Hersteller mit integrierten Produktmodellen in der Praxis nicht üblich.

Von den CAx-System-Herstellern wurden nach [Stekolschik, 2007, S. 2] aber eigene integrierte CAx-Systeme entwickelt, welche den Ansatz des integrierten Produktmodells nutzen. Dafür wurden bestehende CAD-Systeme um zusätzliche Module erweitert. Diese Module bieten Funktionen, die auch von spezialisierten CAE-, CAP-, CAM- und anderen CAx-Systemen bereitgestellt werden. Der Vorteil dieser integrierten Systeme ist, dass Daten ohne Probleme zwischen den beteiligten Modulen ausgetauscht werden können, da sie ein einheitliches Datenmodell verwenden. Der Funktionsumfang der Module ist aber im Vergleich zu spezialisierten CAx-Systemen eingeschränkt. Daher können nicht alle Problemstellungen der Produktentwicklung mit den integrierten CAx-Systemen gelöst werden, sondern es werden verschiedene CAx-Systeme benötigt. Zum Austausch von Daten zwischen CAx-Systemen können Dateien verwendet werden. Dabei werden von einem CAx-System »A« Daten, die zu einem CAx-System »B« übergeben werden sollen, in eine Datei exportiert. Anschließend werden im CAx-System »B« die Daten aus der Datei importiert. In Abbildung 1.1 (b) ist der dateibasierte Austausch im Vergleich zur Verwendung eines integrierten Produktmodells dargestellt.

Ein Problem beim dateibasierten Datenaustausch ist, dass in den Unternehmen, auf Grund der Vielzahl an eingesetzten CAx-Systemen und Schnittstellen zwischen diesen komplexe Systemlandschaften entstehen. Ein Beispiel dafür wird in einer Abbildung in [Gümbel, 2006, S. 47] gezeigt. Es sind 28 CAx-Systeme dargestellt, die bei der Firma »Dr. Ing. h.c. F. Porsche AG« für Simulationszwecke eingesetzt werden. Die CAx-Systeme sind über 58 Schnittstellen miteinander verbunden. Da in der Firma nicht nur Simulationen durchgeführt werden, ist die Anzahl der eingesetzten CAx-Systeme und benötigten Schnittstellen noch höher. Weiterhin ergibt sich eine hohe Anzahl an Schnittstellen, wenn während der Produktentwicklung mehrere Unternehmen zusammenarbeiten. Zum Beispiel werden bei komplexen Produkten einzelne Komponenten von sogenannten Zulieferfirmen entwickelt und gefertigt. Der Hersteller des Produktes benötigt aber von den Zulieferfirmen auch Daten über die Komponenten, um gesamtheitliche Analysen des Produktes durchführen zu können. Ein weiteres Beispiel für den Datenaustausch zwischen mehreren Unternehmen sind sogenannte »virtuelle Unternehmen« [Faisst u. Stürken, 1997]. Hier

arbeiten mindestens zwei Unternehmen für einen bestimmten Zeitraum zusammen, um ein oder mehrere Projekte gemeinsam zu absolvieren.

Weiterhin kann es zwischen zwei CAx-Systemen verschiedene Schnittstellen geben und damit unterschiedliche Möglichkeiten, um Daten auszutauschen. Die Art und der Umfang der Daten kann dabei variieren. Für die Mitarbeiter fehlt oft das Wissen darüber, welche Daten wie übertragen werden können und welche Vor- und Nachteile die einzelnen Möglichkeiten bieten.

Ein weiteres Problem im Zusammenhang mit dem dateibasierten Datenaustausch ist, dass sich durchgeführte Datenaustauschprozesse schwer nachvollziehen lassen. Es werden viele Dateien erzeugt, die aber keine Daten über den durchgeführten Austauschprozess enthalten. Beispielsweise lässt sich anhand einer Datei nicht immer nachvollziehen, von welchem CAx-System sie erzeugt wurde. Weitere Daten, wie zum Beispiel Einstellungen beim Export, gehen ebenfalls verloren. Um die durchgeführten Datenaustauchprozesse nachvollziehen zu können, müssen die Dateien sowie zusätzliche beschreibende Daten (Metadaten) verwaltet werden. Die Verwaltung von Dateien und zugehörigen Metadaten ist technisch kein Problem. Hierfür können zum Beispiel »Produktdatenmanagement-Systeme« [Eigner u. Stelzer, 2009; VDI 2219, 1993] eingesetzt werden. Dafür muss aber zunächst ermittelt werden, welche Metadaten zur Nachvollziehbarkeit des Datenaustauschs notwendig sind und welche Beziehungen die Metadaten untereinander besitzen.

Zusammenfassend treten in der Produktentwicklung durch den dateibasierten Datenaustausch im wesentlichen zwei Probleme auf. Zum einen die Komplexität der Systemlandschaft und zum anderen Schwierigkeiten bei der Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen. Die beschriebenen Probleme machen deutlich, dass eine Unterstützung erforderlich ist.

1.2 Ziele der Arbeit

Im Rahmen dieser Arbeit erfolgt daher die Entwicklung eines Konzeptes für ein Software-System zur Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung. Das Ziel ist dabei nicht, vorhandene Schnittstellen für den dateibasierten Datenaustausch zu verändern, sondern zu untersuchen, wie bestehende Schnittstellen und CAx-Systeme eines Unternehmens verwaltet werden können, um dadurch den Überblick über die Systemlandschaft des Unternehmens zu verbessern. Weiterhin soll die Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen durch eine vollständige Beschreibung der in diesem Zusammenhang durchgeführten Schritte erreicht werden.

Der Schwerpunkt des Konzeptes liegt auf der Entwicklung eines »semantischen Datenmodells«, welches die zur Unterstützung notwendigen Daten und ihre Beziehungen untereinander beschreibt. Dazu zählen zum einen Daten über die Systemlandschaft und zum anderen zur Nachvollziehbarkeit des Datenaustauschs. Diese Daten können auch als Metadaten für die bei den Datenaustauschprozessen entstehenden Dateien verwendet werden. Daher soll eine Beziehung zwischen den Daten und erzeugten Dateien beschrieben werden. Im Folgenden wird zur Vereinfachung an Stelle von »semantisches Datenmodell« nur »Datenmodell« verwendet.

Weiterhin sollen im Konzept grundlegende Funktionen für ein Software-System beschrieben werden, welches auf Basis des Datenmodells eine Unterstützung für den dateibasierten Datenaustausch bietet. Als Grundlage für die Konzeptentwicklung soll zunächst eine Analyse durchgeführt werden, um zu ermitteln, welche Daten für die Abbildung der Systemlandschaft und zur Nachvollziehbarkeit benötigt werden. Das Konzept wird implementierungsunabhängig definiert und eine mögliche Umsetzung des Konzeptes wird anhand einer prototypischen Implementierung gezeigt.

1.3 Gliederung der Arbeit

Die Diplomarbeit gliedert sich in sechs Kapitel. Im Anschluss an die Einleitung werden in **Kapitel 2** die wesentlichen Grundlagen zum Verständnis der Arbeit beschrieben. Dazu gehören Erläuterungen zur Produktentwicklung, zur virtuellen Produktentwicklung und zum dateibasierten Datenaustausch. Das **Kapitel 3** beschreibt die Ergebnisse von drei durchgeführten Analysen. Als erstes wird eine Analyse des dateibasierten Datenaustauschs bei einem Teilprozess in der Entwicklung von Industrierobotern beschrieben. Ausgehend von den dabei festgestellten Problemen wurden zwei vertiefende Analysen durchgeführt, die im Anschluss daran erläutert werden. Zum einen ein Vergleich von Dateiformaten zum Austausch von CAD-Daten und zum anderen eine Analyse zur Nachvollziehbarkeit des Datenaustauschs. Ausgehend von den Ergebnissen der Analysen wird in **Kapitel 4** ein Konzept für ein Software-System zur Unterstützung des dateibasierten Datenaustauschs beschrieben. Eine prototypische Implementierung des Konzeptes auf Basis eines bestehenden Prototypen wird in **Kapitel 5** erläutert. In **Kapitel 6** werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick für weiterführende Arbeiten gegeben.

2 Grundlagen

In diesem Kapitel werden wesentliche Grundlagen zum Verständnis der Arbeit vermittelt. Es werden Begriffe definiert und erläutert, deren Bedeutung nicht vorausgesetzt werden kann oder für die es unterschiedliche Bedeutungen gibt. In Abschnitt 2.1 werden zunächst einige grundlegende Begriffe definiert und erläutert. Im darauffolgenden Abschnitt 2.2 wird erläutert, was unter dem Begriff »virtuelle Produktentwicklung« zu verstehen ist. In diesem Zusammenhang werden Technologien und Software-Systeme beschrieben, die zur Unterstützung der Produktentwicklung eingesetzt werden. Da sich die Arbeit mit der Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung befasst, werden in Abschnitt 2.3 verschiedene Ansätze zum dateibasierten Datenaustausch beschrieben. In Abschnitt 2.4 wird die Unterstützung von Prozessen in der Produktentwicklung erläutert, weil in dieser Arbeit darauf Bezug genommen wird.

2.1 Grundlegende Begriffe

In diesem Abschnitt werden die Begriffe »Produkt«, »Prozess«, »Produktentwicklung«, »Produktlebenszyklus« und »Mechatronik« definiert und erläutert, weil sie für das Verständnis der darauffolgenden Abschnitte von Bedeutung sind.

2.1.1 Produkt

Nach der VDI-Richtlinie 2221 können Produkte in materielle und immaterielle Erzeugnisse eingeteilt werden [VDI 2221, 1993]. Immaterielle Erzeugnisse sind zum Beispiel Dienstleistungen oder Software-Systeme. In dieser Arbeit wird der Begriff Produkt ausschließlich für materielle technische Erzeugnisse, wie zum Beispiel Maschinen, Anlagen, Fahrzeuge, Flugzeuge, aber auch deren Komponenten, verwendet. Zum Teil ist mit dem Begriff Produkt in dieser Arbeit an Stelle eines physischen Produktes ein geplantes beziehungsweise zu entwickelndes Produkt gemeint. Die Bedeutung geht aus dem Zusammenhang hervor oder wird ansonsten durch die Attribute »geplant« und »physisch« gekennzeichnet.

2.1.2 Prozess

Für den Begriff Prozess gibt es verschiedene Definitionen. In dieser Arbeit soll eine Definition verwendet werden, auf der im Folgenden weitere Definitionen aufbauen. Nach [Krause et al., 2002, S. 272] ist ein Prozess:

> »Eine in ihrer Länge oder Dauer nicht begrenzte Folge von ... Aktivitäten, wobei eine ... Aktivität durch ein oder mehrere Ereignisse gestartet wird

> und in einem oder mehreren Ereignissen endet. Die einzelnen ... Aktivitäten sind inhaltlich abgeschlossen, sie stehen in einem logischen Zusammenhang zueinander. Relationen zwischen den Aktivitäten werden durch Verbindungen repräsentiert, die eine zeitliche oder logische (kausale) Reihenfolge beschreiben.«

Als Synonym für den Begriff Prozess kann der Begriff »Prozesskette« verwendet werden. Weiterhin lässt sich ein Prozess in mehrere »Teilprozesse« zerlegen, die jeweils mindestens eine Aktivität umfassen. Ein Teilprozess, der aus einer Aktivität besteht, wird in dieser Arbeit als »Prozessschritt« bezeichnet.

2.1.3 Produktentwicklung

Für den Begriff Produktentwicklung gibt es in der Fachliteratur unterschiedliche Definitionen. Einerseits wird in der VDI-Richtlinie 2221 [VDI 2221, 1993] die Produktentwicklung mit dem Begriff Konstruktion gleichgesetzt. Andererseits umfasst die Produktentwicklung in [Pahl et al., 2007] neben der Konstruktion auch die Produktplanung. Des Weiteren kann die Produktentwicklung auch als Prozess aufgefasst werden und ist nach [Krause et al., 2002, S. 272] die:

> »Folge von Aktivitäten, die zur Entwicklung eines Produktes von der ersten Idee bis zur Freigabe für die Fertigung notwendig sind.«

In dieser Arbeit wird für den Begriff Produktentwicklung die zuletzt genannte Definition verwendet. Zur Beschreibung einer allgemeinen Vorgehensweise bei der Entwicklung von Produkten können Vorgehensmodelle verwendet werden. Sie dienen nach [Lindemann, 2007, S. 33] zur Unterstützung bei der Planung von Prozessen und der Orientierung innerhalb der Produktentwicklung sowie der Reflexion abgeschlossener Prozesse und basierend auf Methoden, die sich in der Praxis bewährt haben. Zwei verbreitete und allgemeine Vorgehensmodelle für die Entwicklung von technischen Produkten beschreiben die VDI-Richtlinie 2221 [VDI 2221, 1993] sowie Pahl und Beitz [Pahl et al., 2007]. Die VDI-Richtline 2221 stellt eine allgemein anerkannte Vorgehensweise dar und beschreibt eine branchenunabhängige Entwicklung technischer Produkte, die sich an den zu erstellenden Arbeitsdokumenten orientiert. Das Buch von Pahl und Beitz ist ein international anerkanntes Standardwerk der Produktentwicklungsmethodik. Zur Beschreibung der wesentlichen Schritte, die bei einer Produktentwicklung durchgeführt werden, wird als Beispiel das Vorgehensmodell nach Pahl und Beitz erläutert, welches vier Phasen umfasst:

- **Produktplanung:** In der Phase der Produktplanung wird zunächst die zu lösende Aufgabe aus technischer, organisatorischer und wirtschaftlicher Sicht untersucht und beschrieben. Als Ergebnis werden die Anforderungen an das zu entwickelnde Produkt lösungsneutral in Form einer Anforderungsliste festgehalten. Sie bildet die Grundlage für alle weiteren Arbeitsschritte.
- **Konzeptphase:** Basierend auf den Anforderungen werden die Gesamt- und Teilfunktionen des Produktes lösungsneutral herausgearbeitet. Im Anschluss daran werden für alle Funktionen ein oder mehrere Lösungsprinzipien gesucht. Anschließend

werden verschiedene Kombinationen der Lösungsprinzipien, auch als Konzeptvarianten bezeichnet, genauer untersucht und bewertet. Das Ergebnis der Konzeptphase ist die Konzeptvariante, welche die Anforderungen am besten erfüllt.

- **Entwurfsphase:** Auf Basis des funktional geprägten Konzeptes findet die konkrete Gestaltung der Lösung statt. Die Abmessungen des Produktes werden festgelegt und mit Hilfe von Simulationen und Berechnungen untermauert. Außerdem findet eine Strukturierung der Lösung in Baugruppen und Einzelteile statt. Eine Baugruppe besteht jeweils aus mehreren Einzelteilen und/oder untergeordneten Baugruppen. Nach und nach werden alle Einzelteile und Baugruppen dimensioniert. Am Ende der Phase liegt ein Gesamtentwurf vor, der alle wesentlichen gestalterischen Festlegungen zur Produktrealisierung enthält.
- **Ausarbeitungsphase:** Ausgehend vom Entwurf wird eine immer detailliertere Lösung entwickelt. Nach und nach entstehen für alle Einzelteile endgültige Festlegungen, unter anderem für Form, Maße, Oberflächenbeschaffenheit und Material. Als Ergebnis der Ausarbeitungsphase entsteht die Technische Produktdokumentation, welche alle zur Herstellung und Nutzung eines Produktes notwendigen Dokumente enthält [DIN 6789-1, 1990]. Dokumente für die Nutzung sind zum Beispiel die Betriebsanleitung, aber auch Wartungs- und Instandhaltungsanleitungen. Für die Herstellung werden unter anderem Zeichnungen und Stücklisten erstellt.

Die beschriebenen Phasen laufen in der Regel nicht linear ab, sondern es sind mehrere Iterationen notwendig, um das Produkt nach und nach zu optimieren. Im Anschluss an die Produktentwicklung kann das geplante Produkt hergestellt werden. Die Produktentwicklung stellt damit die erste Phase im sogenannten Lebenszyklus eines Produktes dar, in den sie im folgenden Abschnitt eingeordnet wird.

2.1.4 Produktlebenszyklus

Der Lebenszyklus eines Produktes umfasst den Zeitraum von der ersten Idee bis zur Entsorgung des Produktes [DIN ISO 15226, 1999, S. 4]. Je nach Produkt, Branche und Unternehmen gestaltet sich der Produktlebenszyklus unterschiedlich. Da sich die Arbeit mit der Entwicklung von technischen Produkten befasst, wird ein allgemeiner Produktlebenszyklus für technische Produkte nach [VDI 2221, 1993, S. 7 ff.] beschrieben, welcher in Abbildung 2.1 dargestellt ist.

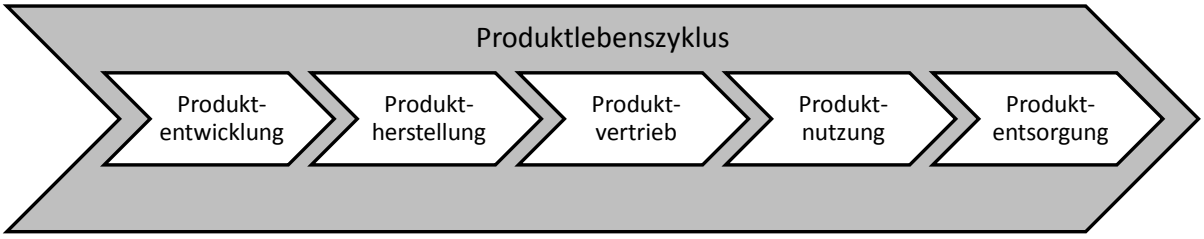


Abbildung 2.1: Produktlebenszyklus (angelehnt an [VDI 2221, 1993, S. 8])

Die erste Phase im Produktlebenszyklus ist die im vorangehenden Abschnitt beschriebene Produktentwicklung. Im Anschluss an die Produktentwicklung findet die Phase der »Produktherstellung« statt. Dabei muss zunächst eine Arbeitsplanung durchgeführt werden, die sich damit befasst, wie das in der Produktentwicklung definierte Produkt gefertigt werden kann. Dafür müssen unter anderem geeignete Fertigungsverfahren, Rohteilabmessungen und Fertigungsmittel ausgewählt werden. Anschließend werden Einzelteile gefertigt, eingekauft und montiert sowie Prüfungen durchgeführt, bis das physische Produkt vollständig vorliegt. In der Phase »Produktvertrieb« wird das Produkt an einen Kunden gebracht, der es in der »Produktnutzungsphase« verwendet. In dieser Phase können auch produktbezogene Dienstleistungen durchgeführt werden, wie zum Beispiel Instandhaltung oder laufende Verbesserungen des Produktes. Nach dem Gebrauch des Produktes findet die »Produktentsorgung« statt. Dafür ist gegebenenfalls eine Demontage notwendig, um Teile des Produktes getrennt voneinander zu entsorgen oder zu recyceln.

Die Produktentwicklung spielt im Produktlebenszyklus eine wichtige Rolle, weil die Entscheidungen, die dort getroffen werden, sich auf alle nachgelagerten Phasen direkt oder indirekt auswirken. Damit werden auch die Kosten für die späteren Phasen größtenteils festgelegt, so dass nach [Richter, 2009, S. 433] ca. 70% der Gesamtkosten aus der Produktentwicklung resultieren. Um die Kosten so niedrig wie möglich zu halten, ist es daher wichtig, dass alle nachgelagerten Phasen während der Produktentwicklung berücksichtigt und vorausgeplant werden. Hinweise zu einer vorausschauenden Entwicklung werden zum Beispiel in [Pahl et al., 2007, S. 393 ff.] und [Schuh u. Baessler, 2009] beschrieben. Dies sind unter anderem Hinweise zur fertigungs-, montage-, instandhaltungs- und recyclinggerechten Gestaltung. Im internationalen Bereich wird dies auch als »Design for X« bezeichnet. Das »X« steht dann zum Beispiel für Manufacturing (Fertigung) oder Assembly (Montage).

Das Problem bei der vorausschauenden Entwicklung ist nach [Schuh u. Baessler, 2009, S. 237], dass Optimierungsziele im Widerspruch zueinander stehen können. Zum Beispiel kann es einen Konflikt zwischen der fertigungs- und montagegerechten Gestaltung geben. Um die Anzahl der Montageschritte zu minimieren, müssen möglichst wenig Einzelteile hergestellt werden. Dadurch ergeben sich aber komplexe Einzelteile, die schwerer zu fertigen sind. Es muss daher ein Mittelweg zwischen fertigungs- und montagegerechter Gestaltung gefunden werden. Dieses Beispiel soll veranschaulichen, wie kompliziert die Berücksichtigung der unterschiedlichen Gestaltungshinweise und damit die Suche nach einer optimalen Gestaltung ist. Wichtig ist vor allem, dass grobe Fehler bei der Gestaltung ausgeschlossen werden. Zur Unterstützung der Entscheidungen bei der Gestaltung können rechnerunterstützte Simulationen durchgeführt werden (siehe Abschnitt 2.2.4).

2.1.5 Mechatronik

Für den Begriff »Mechatronik« existiert eine Vielzahl an Definitionen (siehe [Isermann, 2008, S. 3 f.] und [VDI 2206, 2004, S. 10 ff.]). Als Gemeinsamkeit lässt sich feststellen, dass mit dem Begriff »Mechatronik« das Zusammenwirken der Fachdisziplinen (Domänen) Maschinenbau, Elektrotechnik und Informationstechnik bezeichnet wird. »Mechatronische

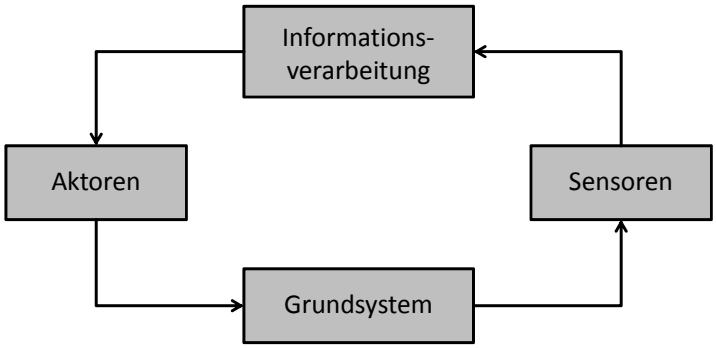


Abbildung 2.2: Grundstruktur eines mechatronischen Systems (angelehnt an [VDI 2206, 2004, S. 14])

Systeme« sind technische Produkte, welche aus Komponenten dieser drei Disziplinen bestehen. Das Ziel ist es, Synergien aus dem Zusammenwirken der einzelnen Komponenten zu nutzen [VDI 2206, 2004, S. 10]. Dafür muss von Beginn an eine ganzheitliche Entwicklung erfolgen, bei der das Zusammenwirken der Komponenten berücksichtigt und aufeinander abgestimmt wird. Anstelle von mehreren unabhängigen Systemen entsteht so ein mechatronisches Gesamtsystem.

In Abbildung 2.2 ist der grundsätzliche Aufbau eines mechatronischen Systems dargestellt, welcher in [VDI 2206, 2004, S. 14 ff.] beschrieben ist. Beim Grundsystem handelt es sich um eine mechanische, elektromechanische, hydraulische oder pneumatische Struktur beziehungsweise eine Kombination aus diesen. Die Aufgabe der Sensoren ist die Bestimmung ausgewählter Informationen über die aktuellen Eigenschaften des Grundsystems und dessen Umgebung. Auf Basis der erfassten Informationen bestimmt die Informationsverarbeitung die notwendigen Einwirkungen, um den Zustand des Grundsystems so zu verändern, dass es sich in gewünschter Weise verhält. Die Umsetzung der Einwirkungen erfolgt durch Aktoren am Grundsystem. Anschließend messen die Sensoren wieder neue Größen. Dadurch entsteht ein Regelkreis mit dem Ziel, das Verhalten des Grundsystems so zu verbessern, dass es im jeweiligen Kontext als optimal angesehen werden kann.

Mit Hilfe der Mechatronik lassen sich innovative Produkte mit erweiterten, verbesserten oder neuen Funktionen entwickeln, wie zum Beispiel Produkte, die ohne Eingriff von außen ihre Verhaltensweise anpassen oder selbständig Fehlerdiagnosen durchführen können. Mechatronische Systeme umfassen komplexe Produkte (Industrieroboter, Werkzeugmaschinen, Waschmaschinen, ...), aber auch Komponenten von Produkten (Anti-Blockier-Systeme, Motorsteuerungen, geregelte Katalysatoren, ...).

Eine Herausforderung bei der Entwicklung von mechatronischen Systemen ist die steigende Komplexität, die nach [Schlund et al., 2009] zu mehr Systemausfällen führt, da keine ausreichenden Testes gemacht werden. Die Folgen sind kostenintensive Rückrufaktionen oder Stillstandszeiten. Weitere Probleme sind nach [Pahl et al., 2007, S. 609 f.], dass:

- elektronische Komponenten anfällig für mechanische/thermische Belastungen sind,
- Reparaturen oft nicht möglich oder zweckmäßig sind,
- das Preis-Leistungs-Verhältnis oft nicht gegeben ist.

Die Entwicklung von mechatronischen Systemen erfordert außerdem eine interdisziplinäre Zusammenarbeit. Das Problem dabei ist, dass sich in den einzelnen Domänen (Maschinenbau, Elektrotechnik, Informationstechnik) über Jahrzehnte jeweils eigene Vorgehensweisen, Methoden und Software-Systeme etabliert haben [VDI 2206, 2004, S. 22 f.]. Die Entwicklung erfolgt daher meist getrennt in den einzelnen Domänen. Um eine interdisziplinäre Zusammenarbeit zu ermöglichen, werden zum einen Software-Systeme benötigt, welche eine ganzheitliche Entwicklung von mechatronischen Systemen ermöglichen. Dafür existieren aber nach [Franke et al., 2009] bisher nur wenige Lösungen. Der übliche Weg ist die Entwicklung in unterschiedlichen Software-Systemen. Zum anderen werden domänenübergreifende Vorgehensmodelle benötigt, wofür die »VDI-Richtlinie 2206« ein Beispiel ist, die im Folgenden kurz erläutert wird.

Der wesentliche Ablauf des Vorgehensmodells wird durch ein »V-Modell« beschrieben, welches in Abbildung 2.3 dargestellt ist. Es ist an das V-Modell aus der Softwareentwicklung angelehnt (siehe zum Beispiel [Dumke, 2003, S. 119 f.]). Basierend auf den Anforderungen eines Entwicklungsauftrags wird in der Phase »Systementwurf« zunächst ein domänenübergreifendes Lösungskonzept entwickelt. Anschließend erfolgt der »domänenspezifische Entwurf«, bei dem in den beteiligten Domänen eine weitere Konkretisierung mit Hilfe domänenspezifischer Vorgehensmodelle, Methoden und Software-Systeme stattfindet. In der Phase »Systemintegration« werden die Ergebnisse aus den einzelnen Domänen zu einem Gesamtsystem integriert und das Zusammenwirken wird untersucht. Dabei ist zu überprüfen, ob die erzielten Systemeigenschaften den im Rahmen des Systementwurfs festgelegten Merkmalen entsprechen und die gewünschten Anforderungen erfüllen. Als Ergebnis entsteht als Produkt ein mechatronisches System.

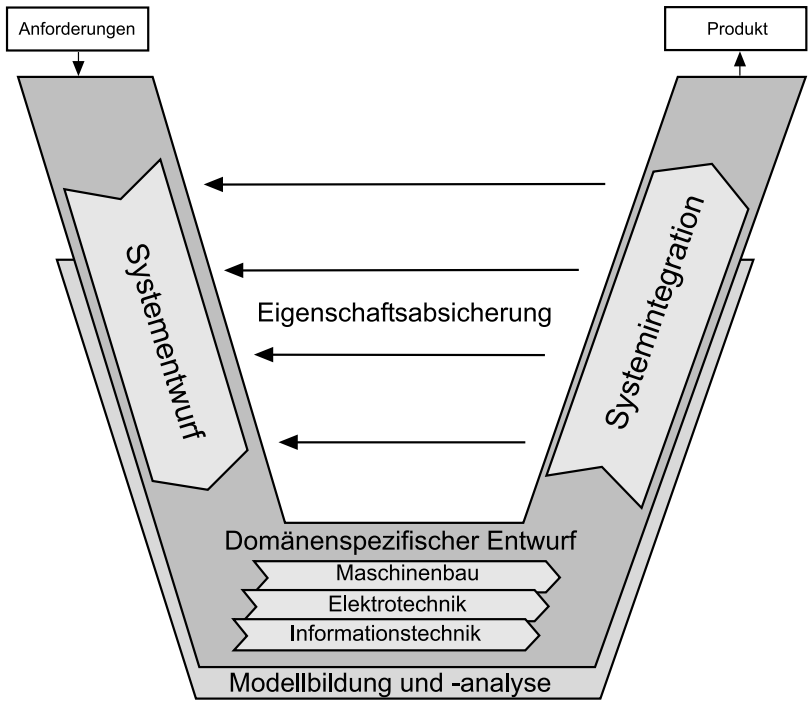


Abbildung 2.3: Vorgehensmodell nach VDI-Richtlinie 2206 [VDI 2206, 2004, S. 29]

2.2 Virtuelle Produktentwicklung

Auf Grund der hohen Anforderungen an innovative Produkte und die Notwendigkeit, Produkte in kurzer Zeit zu geringen Kosten und in hoher Qualität zu entwickeln und herzustellen, wurden im Laufe der Zeit immer mehr rechnerunterstützte Systeme und virtuelle Technologien in der Produktentwicklung eingesetzt. Ausgehend von der Modellierung der Gestalt eines Produktes konnten nach und nach immer mehr Eigenschaften, bis hin zum Verhalten eines Produktes, rechnerintern abgebildet werden [Abramovici, 2002]. Als Ergebnis dieser Entwicklung ist es seit Mitte der 90er Jahre möglich, eine sogenannte »virtuelle Produktentwicklung (VPE)« zu realisieren. Darunter wird nach [Eigner, 2009, S. 250] die:

> »... durchgehende Rechnerunterstützung bei der Produktentwicklung unter intensiver Anwendung von Simulations- und Verifikationstechniken auf der Basis digitaler, realitätsnaher Modelle verstanden.«

Die Grundlagen der VPE werden umfassend in [Spur u. Krause, 1997] beschrieben. Das Ziel der VPE ist eine möglichst vollständige und realitätsnahe digitale Beschreibung des Produktes, um Untersuchungen virtuell durchführen zu können. Außerdem sollen einmal erzeugte Daten digital weitergegeben werden können. Dafür fehlt heutzutage nach [Schenk u. Schmucker, 2009, S. 54] zum Teil noch die Interoperabilität der rechnerunterstützten Systeme. Wenn kein digitaler Austausch möglich ist, müssen Daten erneut eingegeben werden. Dies ist fehleranfällig und zeitaufwändig. Des Weiteren können auch nicht alle Untersuchungen virtuell durchgeführt werden, wie zum Beispiel Anforderungen im Zusammenhang mit den menschlichen Sinnen Fühlen, Schmecken und Riechen. Der Begriff VPE sollte daher nicht so gedeutet werden, dass alle Untersuchungen virtuell durchgeführt werden, sondern es wird versucht, die virtuellen Technologien dort einzusetzen, wo es möglich und sinnvoll ist.

Mit dem Begriff »virtuelle Produktentstehung« wird die Idee der VPE auf die Phase der Produktherstellung ausgeweitet (siehe [Krause et al., 2002; Stark et al., 2011]). Neben der Unterstützung der Produktentwicklung ist damit auch der Einsatz von virtuellen Technologien für die Analyse und Simulation der Herstellung eines Produktes gemeint. Dies umfasst außerdem die Unterstützung der Fabrikplanung mit Hilfe von virtuellen Untersuchungen (»Digitale Fabrik«) [Westkämper u. Niemann, 2009; VDI 4449, 2008]. Als Synonym für die »virtuelle Produktentstehung« wird auch der Begriff »Virtual Engineering« verwendet (siehe [Warschat, 2009; Bernard, 2005]). Da der Schwerpunkt dieser Arbeit auf der Produktentwicklung liegt, wird im Folgenden nur die »virtuelle Produktentwicklung« und nicht die »virtuelle Produktentstehung« betrachtet.

Um ein geplantes Produkt rechnerintern zu beschreiben, werden bei der VPE verschiedene Modelle benötigt. Nach [VDI 2211, 2003, S. 13] ist ein Modell eine vereinfachte Nachbildung eines Originals, um dieses für einen bestimmten Zweck zu repräsentieren. Dabei beschreibt ein Modell nur eine Auswahl der Merkmale eines Originals. Zusätzlich zur Beschreibung des Produktes können die Modelle für virtuelle Untersuchungen verwendet werden. Dadurch können Untersuchungen an »physischen Prototypen« reduziert werden. Die Modelle zur Durchführung von virtuellen Untersuchungen werden als »virtuelle Prototypen (VP)« bezeichnet [Zorriassatine et al., 2003].

2.2.1 Virtueller Prototyp

Bei der Untersuchung an physischen Prototypen sollen Erkenntnisse auf das geplante Produkt übertragen werden. Mit dem gleichen Ziel werden Untersuchungen an VP durchgeführt. Für einen bestimmten Untersuchungszweck muss der VP dafür die relevanten Eigenschaften des Produktes ausreichend genau repräsentieren. Ansonsten können die gewonnenen Erkenntnisse nicht auf das Produkt übertragen werden. Die Überprüfung, ob der VP das Produkt ausreichend genau repräsentiert, wird als »Validierung« bezeichnet [VDI 2206, 2004, S. 53]. Die Erkenntnisse aus den Untersuchungen an einem physischen oder virtuellen Prototypen können beispielsweise folgende Aufgaben unterstützen:

- Machbarkeitsanalysen
- Bewertung von Konzeptalternativen
- Auslegung und Optimierung von Parametern
- Überprüfung der festgelegten Produkteigenschaften mit den Anforderungen

Ein wesentlicher Vorteil des Einsatzes von VP gegenüber physischen Prototypen ist nach [Schenk u. Straßburger, 2005, S. 507] die Einsparung von Herstellungskosten und Zeit für die Durchführung von Untersuchungen. Weiterhin können VP schneller erzeugt und leichter geändert werden, so dass viele Varianten untersucht werden können. Mit Hilfe von VP können außerdem Untersuchungen durchgeführt werden, die mit physischen Prototypen nicht möglich sind, weil sie zum Beispiel zu teuer oder zu gefährlich wären. Ein weiterer Vorteil ist, dass ein VP beliebig oft eingesetzt werden kann.

2.2.2 Virtual Reality

»Virtual Reality (VR)« beziehungsweise »virtuelle Realität« bezeichnet eine Technologie zur realitätsnahen Erzeugung einer virtuellen Welt [Eigner, 2009, S. 254 f.]. Dabei sollen möglichst viele Sinne des Anwenders angesprochen werden, um die virtuelle Welt so real wie möglich erscheinen zu lassen. Außerdem sollte sich der Anwender in der Welt frei bewegen und mit ihr in Echtzeit interagieren können. Das Ziel ist es, dass der Anwender in die virtuelle Welt »eintaucht«, was auch als »Immersion« bezeichnet wird [Anderl u. Grabowski, 2007, S. 22]. Der Schwerpunkt bei VR ist die Visualisierung, wofür 3D-Modelle benötigt werden.

VR-Systeme können zur Bewertung von VP verwendet werden. Im einfachsten Fall wird ein VP nur visualisiert, um zum Beispiel die Gestalt und das Aussehen bewerten zu können. Ein Vorteil bei der Visualisierung von VP gegenüber physischen Prototypen ist, dass innen liegende Elemente durch Ausblenden oder Transparenz sichtbar gemacht werden können [Schenk u. Straßburger, 2005, S. 508]. Weiterhin kann mit Hilfe eines VR-Systems eine Interaktion mit einem VP realisiert werden. Dies lässt sich zum Beispiel für Schulungszwecke an Maschinen (Bedienung, Instandhaltung) nutzen. Eine virtuelle Maschine kann nicht zerstört werden und ermöglicht ein gefahrloses Training von kritischen Situationen. Ein weiterer Vorteil ist, dass Schulungen bereits durchgeführt werden können, bevor eine Maschine hergestellt ist.

2.2.3 Rapid Prototyping

»Rapid Prototyping« bezeichnet eine Technologie, mit der durch schichtweises Auftragen von Material in kurzer Zeit ein kostengünstiger Prototyp hergestellt werden kann [Eigner, 2009, S. 253]. In Abhängigkeit vom Material werden verschiedene Verfahren unterschieden (siehe [Tenbusch, 2002, S. 21 ff.] und [Kimura, 2002, S. 67 ff.]). Mit Hilfe von Rapid Prototyping ist es möglich, aus einem VP in kurzer Zeit einen physischen Prototypen zu erzeugen. Dies ist notwendig, da sich nicht alle Untersuchungen virtuell durchführen lassen. Mit Rapid Prototyping können Prototypen von komplexen Produkten erzeugt werden, die durch konventionelle Verfahren nur mit hohen Kosten herstellbar wären [VDI 2209, 2009, S. 116]. Zum Beispiel auch bei geometrisch komplexen Formen mit Hohlräumen und Hinterschneidungen. Nachteile von »Rapid Prototyping« sind dagegen, dass:

- nicht alle Materialien geeignet sind,
- Rundungen wegen der schichtweisen Herstellung nicht möglich sind,
- dünne Teile nicht herstellbar sind,
- die Technologie auf kleine Teile beschränkt ist.

2.2.4 CAx-Systeme

Im Produktlebenszyklus werden verschiedene rechnerunterstützte Software-Systeme eingesetzt, die zusammenfassend als »Computer Aided x (CAx)-Systeme« bezeichnet werden. Das »x« steht dabei als Platzhalter für eine Vielzahl von Akronymen, wie zum Beispiel CAD, CAE, CAP und CAM. Durch die CAx-Systeme werden zum einen Routineaufgaben unterstützt und zum anderen bieten sie neue Methoden und Verfahren für die Produktentwicklung. Zur Realisierung einer VPE werden verschiedene CAx-Systeme eingesetzt. Im Folgenden werden die wichtigsten CAx-Systeme näher beschrieben, die in Tabelle 2.1 aufgelistet sind.

Tabelle 2.1: Beispiele für CAx-Systeme

	englische Bezeichnung	deutsche Bezeichnung
CAD	Computer Aided Design	Rechnerunterstützte Konstruktion
CAE	Computer Aided Engineering	Rechnerunterstützung zur Berechnung, Analyse, Simulation und Optimierung
CAP	Computer Aided Planning	Rechnerunterstützte Arbeitsplanung
CAM	Computer Aided Manufacturing	Rechnerunterstützte Fertigung

CAD-Systeme: »Computer Aided Design (CAD)« bezeichnet die rechnerunterstützte Konstruktion (siehe [Pahl et al., 2007, S. 79 ff.] und [VDI 2209, 2009]). Es wird zwischen CAD-Systemen für die mechanische (M-CAD) und elektrische Konstruktion (E-CAD) unterschieden. In dieser Arbeit wird im Folgenden anstatt M-CAD nur CAD verwendet.

Die ersten CAD-Systeme wurden entwickelt, um die Erstellung von technischen Zeichnungen zu unterstützen. Sie konnten daher nur 2D-Modelle erzeugen und wurden als 2D-CAD-Systeme bezeichnet. Der Vorteil gegenüber der konventionellen Erstellung von Zeichnungen am Reißbrett ist, dass Änderungen und Vervielfältigungen der Zeichnungen einfacher erstellt werden können.

Mit den 3D-CAD-Systemen, die heutzutage vorwiegend eingesetzt werden, wurde eine völlig neue Vorgehensweise in der Produktentwicklung eingeführt. Die Zeichnungserstellung steht nicht mehr im Mittelpunkt, sondern Produkte werden direkt als virtuelles 3D-Modell konstruiert. Zusätzlich können weitere Eigenschaften des Produktes definiert werden, wie zum Beispiel Material- und Fertigungsinformationen sowie Toleranzen und Oberflächenrauhigkeiten. 3D-CAD-Systeme ermöglichen daher die digitale Beschreibung eines geplanten Produktes in einem rechnerinternen Modell. Ein mit einem CAD-System erstelltes Modell (3D-Modell inklusive weiterer Informationen) wird im Folgenden als »CAD-Modell« bezeichnet.

Die 3D-CAD-Systeme bilden die Kernkomponente zur Realisierung einer VPE, da die erzeugten CAD-Modelle für verschiedene Problemstellungen weiterverwendet werden können. Im CAD-System lassen sich teilautomatisch Dokumente beziehungsweise Modelle, wie zum Beispiel Zeichnungen und Stücklisten, ableiten. Andere CAx-Systeme können auf Basis der CAD-Modelle weitere Modelle, wie zum Beispiel Simulationsmodelle, NC-Programme und Arbeitspläne, erzeugen. Des Weiteren stellen CAD-Modelle die Grundlage für die in Abschnitt 2.2.2 und 2.2.3 beschriebenen Technologien »Virtual Reality« und »Rapid Prototyping« dar. Weiterhin können 3D-Modelle als aufbereitete fotorealistische Darstellungen für das Marketing oder als Abbildung in Handbüchern verwendet werden.

CAE-Systeme: Mit CAE-Systemen lassen sich Berechnungen, Analysen, Simulationen und Optimierungen rechnerunterstützt durchführen [Anderl u. Grabowski, 2007, S. 21]. Es können verschiedene Aspekte untersucht werden, wie zum Beispiel Mechanik (Verformungen, Spannungen, Bewegungen), Elektrik, Temperaturverhalten und Strömungen, aber auch Kombinationen davon. CAE-Systeme dienen vorwiegend zur Unterstützung der Konstruktion (Auslegung von Parametern, Optimierung, Verhalten simulieren, ...), werden aber auch im Zusammenhang mit Machbarkeitsanalysen oder zum Vergleich von Alternativen in den frühen Phasen der Produktentwicklung eingesetzt. Zur Untersuchung mit einem CAE-System muss zunächst ein geeignetes Modell erstellt werden. Dafür kann zum Teil ein vorhandenes CAD-Modell verwendet werden, welches die Gestalt und weitere Informationen über ein geplantes Produkt enthält.

Zwei wichtige Simulationsmethoden, die mit CAE-Systemen durchgeführt werden können, sind:

- **Finite-Elemente-Methode (FEM)-Simulation:** Untersuchung von verschiedenen physikalischen Problemstellungen (Verformungen, Spannungen, Wärmeübertragung, Akustik, Strömungen, ...) durch Zerlegung eines zu untersuchenden Körpers in eine endliche (»finite«) Anzahl kleiner Elemente (siehe [VDI 2211, 2003, S. 26 ff.] und [VDI 2209, 2009, S. 108 ff.])

- **Mehrkörpersystem (MKS)-Simulation:** Untersuchung des Bewegungsverhaltens komplexer Systeme, die aus gekoppelten beweglichen Teilen bestehen, wie zum Beispiel Industrieroboter oder Werkzeugmaschinen (siehe [VDI 2211, 2003, S. 31], [VDI 2209, 2009, S. 112 ff.], [Shabana, 2005] und Abschnitt 3.1)

CAP-Systeme: CAP-Systeme unterstützen die Arbeitsplanung, die sich damit befasst, wie das von der Konstruktion definierte Produkt gefertigt werden kann [Anderl u. Grabowski, 2007, S. 21]. Dafür sollen möglichst viele Informationen aus einem vorhandenen CAD-Modell gewonnen werden. Bei der Arbeitsplanung müssen unter anderem geeignete Fertigungsverfahren, Rohteilabmessungen und Fertigungsmittel ausgewählt werden. Als Ergebnis entsteht ein Arbeitsplan, der die einzelnen Schritte zur Fertigung inklusive Zeitvorgaben beschreibt. CAP-Systeme können zum Beispiel die Auswahl und Bewertung von Fertigungsverfahren und -mittel unterstützen.

CAM-Systeme: CAM-Systeme dienen insbesondere zur Erstellung von Steuerdaten (NC-Programme) für numerisch gesteuerte Werkzeugmaschinen [Anderl u. Grabowski, 2007, S. 22]. NC-Programme können vollständig oder teilweise aus CAD-Modellen abgeleitet werden. Außerdem lassen sich mit CAM-Systemen Fertigungsprozesse an Werkzeugmaschinen simulieren und damit die erstellten NC-Programme überprüfen.

2.3 Dateibasierter Datenaustausch

Die Realisierung einer VPE erfordert nach [Swienczek et al., 1998, S. 220] einen effizienten Datenaustausch zwischen einer Vielzahl an CAx-Systemen. Zwischen den CAx-Systemen sollen Produktdaten, in diesem Abschnitt im Folgenden als Daten bezeichnet, digital ausgetauscht werden. Der Grund dafür ist, dass einmal erzeugte Daten von anderen CAx-Systemen genutzt werden sollen. Eine erneute manuelle Eingabe der Daten, die zu Fehlern führen kann und durch die ein unnötiger Zeitaufwand entsteht, wird dadurch verhindert.

Ein allgemeines Problem beim Datenaustausch ist, dass jedes CAx-System auf einem eigenen Datenmodell basiert. Es können daher nur die Daten übertragen werden, die im empfangenden CAx-System abbildbar sind. Wenn es in einem CAD-System zum Beispiel möglich ist, das Material von Einzelteilen zu definieren und im empfangenden System nicht, dann können die Materialdaten nicht übertragen werden und gehen somit für die Weiterverarbeitung verloren. Die Anforderung an einen Datenaustausch kann daher nicht sein, dass alle Daten in ein anderes CAx-System übertragen werden sollen, sondern möglichst alle Daten, die sich im anderen CAx-System abbilden lassen beziehungsweise dort benötigt werden. Wichtig ist vor allem die fehlerfreie Übertragung von Daten sicherzustellen.

2.3.1 Dateibasierte Schnittstellen

Zum Austausch von Daten zwischen zwei CAx-Systemen müssen diese über eine Schnittstelle verbunden werden. Nach [Grabowski u. Anderl, 1990, S. 26] ist eine Schnittstelle:

> »... ein System von Bedingungen, Regeln und Vereinbarungen, das den Informationsaustausch zweier miteinander kommunizierender Systeme oder Systemkomponenten festlegt.«

Eine spezielle Form von Schnittstellen sind »dateibasierte Schnittstellen«. Dabei verwenden zwei CAx-Systeme Dateien zum Austausch von Daten. Ein CAx-System speichert die zu übertragenden Daten in einer Datei, die in einem anderen CAx-System importiert werden kann. Da sich diese Arbeit mit der Unterstützung des dateibasierten Datenaustauschs befasst, wird diese Variante zum Austausch von Daten zwischen CAx-Systemen im Folgenden näher erläutert.

2.3.2 Systemspezifische und systemneutrale Schnittstellen

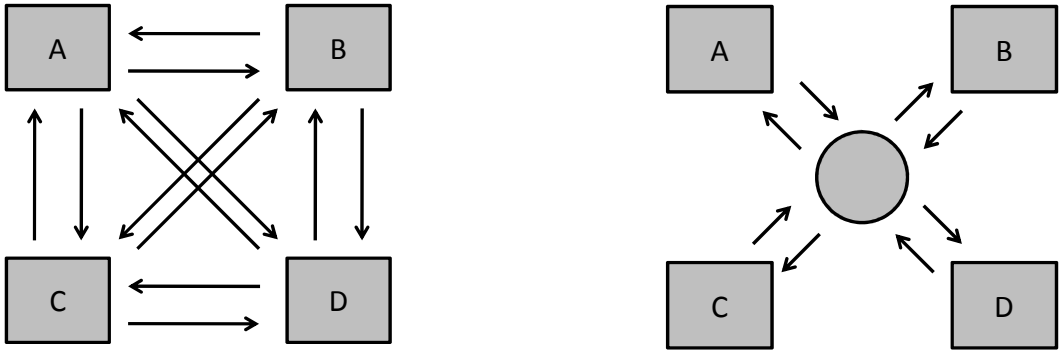
Zur Realisierung von dateibasierten Schnittstellen zwischen CAx-Systemen gibt es zwei unterschiedliche Ansätze. Es können »systemspezifische Schnittstellen« oder eine »systemneutrale Schnittstelle« verwendet werden. Ein Vergleich der beiden Ansätze wird zum Beispiel in [Grabowski u. Anderl, 1990, S. 34 f.], [Anderl u. Trippner, 2000, S. 11 f.] und [Vornholt et al., 2010, S. 12 f.] beschrieben. Beide Ansätze sind in Abbildung 2.4 dargestellt und sollen im Folgenden erläutert sowie verglichen werden. Das wesentliche Problem ist, dass CAx-Systeme unterschiedliche Datenmodelle verwenden. Um Daten von einem CAx-System zu einem anderen zu übertragen, ist eine Konvertierung zwischen den Datenmodellen der beiden CAx-Systeme erforderlich.

Bei der Verwendung von **systemspezifischen Schnittstellen** (Ansatz A) werden bei einer Anzahl von n CAx-Systemen $n \cdot (n-1)$ Konverter benötigt. Die Konverter werden auch als Direktkonverter bezeichnet, weil sie die Daten eines CAx-Systems direkt in das Datenmodell eines anderen CAx-Systems abbilden. Das Problem bei der direkten Konvertierung von jedem CAx-System zu allen anderen ist, dass die Anzahl der Konverter exponentiell mit der Anzahl der CAx-Systeme steigt. In der Regel müssen nicht alle CAx-Systeme mit allen anderen Daten austauschen, aber diese Annahme dient zur Veranschaulichung des allgemeinen Problems der direkten Konvertierung.

Bei der Verwendung einer **systemneutralen Schnittstelle** (Ansatz B) ist ein Format festgelegt, das für alle CAx-Systeme bekannt ist. Die systemspezifischen Daten von einem CAx-System können in dieses Format überführt und aus dem Format ausgelesen werden. Daher werden von jedem System nur zwei Konverter und damit insgesamt $2 \cdot n$ Konverter benötigt.

Beim Vergleich der beiden Ansätze lässt sich feststellen, dass ab einer Anzahl von vier CAx-Systemen für den Ansatz B weniger Konverter erforderlich sind. Je größer die Anzahl der CAx-Systeme ist, desto größer wird der Unterschied zwischen beiden Ansätzen. Außerdem entsteht für das Hinzufügen eines CAx-Systems bei Ansatz A ein höherer Aufwand als bei Ansatz B, denn es müssen zu jedem CAx-System zwei Konverter und damit $2 \cdot n$ Konverter zusätzlich programmiert werden. Bei Ansatz B sind es insgesamt nur zwei Konverter, egal wie viele CAx-Systeme bereits existieren.

Für den Ansatz A ergibt sich dadurch ein höherer Implementierungsaufwand gegenüber Ansatz B. Außerdem ist der Pflegeaufwand bei Ansatz A höher, da bei einer Änderung



(a) systemspezifische Schnittstellen (b) systemneutrale Schnittstelle

Abbildung 2.4: Möglichkeiten zur Realisierung des Datenaustauschs zwischen CAx-Systemen (angelehnt an [Grabowski u. Anderl, 1990, S. 34])

an einem der CAx-Systeme alle Konverter zu den anderen Systemen angepasst werden müssen. Der Vorteil von Ansatz A ist aber, dass mit Hilfe eines Direktkonverters der Datenaustausch zwischen zwei Systemen auf die unterschiedlichen Datenmodelle der Systeme abgestimmt werden kann. Im Gegensatz dazu lassen sich beim Ansatz B nur die Daten übertragen, die mit Hilfe der systemneutralen Schnittstelle abgebildet werden können, wodurch der Datenverlust im Vergleich zu Ansatz A höher ist. Da CAx-Systeme auf unterschiedlichen Datenmodellen basieren, können auch mit Direktkonvertern nicht alle Daten übertragen werden. In Abbildung 2.5 ist der Unterschied des Datenverlustes bei beiden Ansätzen dargestellt. Ein weiterer Nachteil von Ansatz B ist außerdem, dass bei der Änderung der systemneutralen Schnittstelle alle Konverter angepasst werden müssen. Alle beschriebenen Unterschiede sind in der Tabelle 2.2 zusammenfassend dargestellt.

Tabelle 2.2: Vergleich von systemspezifischen und -neutralen Schnittstellen

	systemspezifisch	systemneutral
Anzahl Konverter	$n \cdot (n-1)$	$2 \cdot n$
Aufwand für $(n+1)$-tes System	$2 \cdot n$	2
Implementierungsaufwand	hoch	gering
Pflegeaufwand	hoch	gering
Datenverluste	gering	hoch

Als Ergebnis des Vergleichs lässt sich festhalten, dass systemneutrale Schnittstellen vor allem bei einer großen Anzahl an CAx-Systemen sinnvoll sind, wobei ein höherer Datenverlust als bei Direktkonvertern in Kauf genommen werden muss.

2.3.3 Neutrale und native Dateiformate

Zur Realisierung von systemneutralen Schnittstellen wurden sogenannte »neutrale Dateiformate« entwickelt. Neutrale Dateiformate umfassen genormte Dateiformate und In-

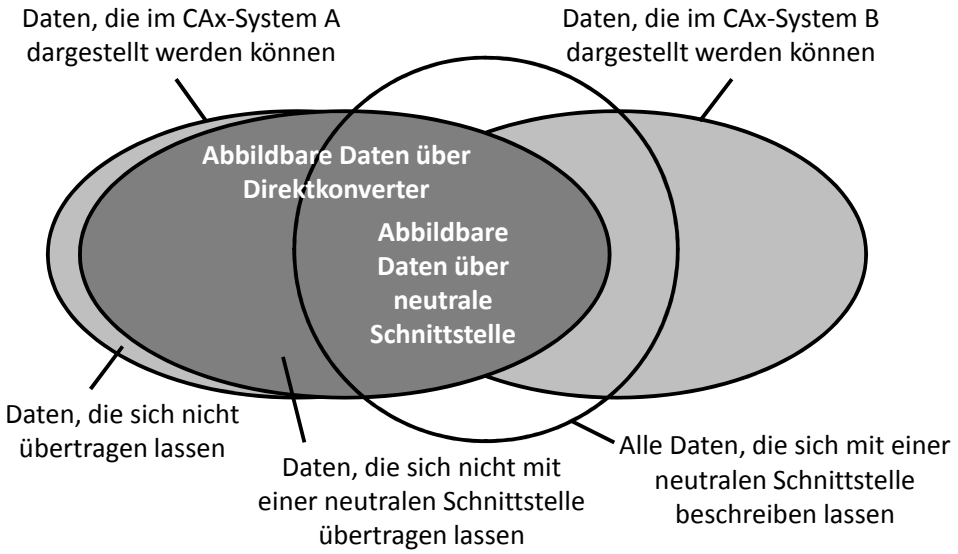


Abbildung 2.5: Datenverlust beim Datenaustausch (angelehnt an [Reim et al., 2007, S. 3])

dustriestandards, weshalb sie außerdem als »Standardformate« bezeichnet werden. Genormte Dateiformate stammen von nationalen (zum Beispiel DIN) oder internationalen Normungsinstituten (zum Beispiel ISO) und sind system- beziehungsweise herstellerunabhängig. Industriestandards sind Technologien, die allgemein benutzt werden und sehr verbreitet sind, aber nicht offen oder demokratisch verwaltet und entwickelt werden, beispielsweise die Programmiersprache »Java« [Lubell et al., 2004, S. 2]. Im Zusammenhang mit Dateiformaten sind damit weit verbreitete Dateiformate gemeint, die von einem CAx-System-Hersteller entwickelt werden, aber frei verfügbar sind.

Ein neutrales Dateiformat gibt eine einheitliche Struktur vor, wie Daten gespeichert werden können. Darauf aufbauend können von einem CAx-System zwei Konverter bereitgestellt werden, die das neutrale Dateiformat importieren und Daten in das neutrale Dateiformat exportieren können. Der Vorteil der neutralen Dateiformate besteht darin, dass die Konverter in den CAx-Systemen in der Regel schon implementiert sind. Alle CAx-Systeme, die ein neutrales Dateiformat unterstützen, können daher Daten austauschen, ohne dass ein Konverter implementiert werden muss. Beispiele für neutrale Dateiformate werden in Abschnitt 3.2 beschrieben.

Alle weiteren Dateiformate werden als »native Dateiformate« bezeichnet. Sie sind system- beziehungsweise herstellerabhängig und nicht frei verfügbar. Jedes CAx-System verwendet ein oder mehrere native Dateiformate, um die mit dem CAx-System erzeugten Daten zu speichern und bei Bedarf wieder einlesen zu können. Ein natives Dateiformat kann in der Regel nicht von einem CAx-System eines anderen Herstellers importiert werden. Zwischen CAx-Systemen eines Herstellers wird der Import aber normalerweise unterstützt. Wenn kein direkter Import möglich ist, muss ein zusätzlicher Konverter verwendet werden, welcher die direkte Konvertierung eines nativen Formates in ein anderes natives Format realisiert. Hierfür wird am Markt eine Vielzahl an kommerziellen Software-Systemen angeboten, beispielsweise »PolyTrans[1]«.

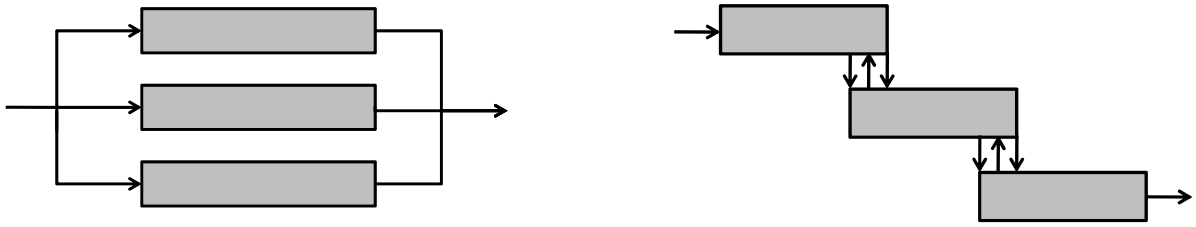
[1] `http://www.okino.com/conv/conv.htm?0` (zuletzt überprüft am 10.04.2011)

2.4 Prozessunterstützung in der Produktentwicklung

Da im Laufe der Arbeit auch auf die Unterstützung von Prozessen in der Produktentwicklung eingegangen wird, sollen in diesem Abschnitt einige allgemeine Aspekte zu diesem Thema erläutert werden.

2.4.1 Parallelisierung von Prozessen

Bei der Entwicklung von Produkten wird seit Ende der 70er Jahre versucht Prozesse zu parallelisieren, um dadurch Entwicklungszeit und -kosten einzusparen [Holmdahl, 2007, S. 23 ff.]. Dafür werden die in Abbildung 2.6 dargestellten Methoden »Concurrent Engineering« und »Simultaneous Engineering« eingesetzt (siehe zum Beispiel [Richter, 2009], [Anderl u. Trippner, 2000, S. 12 ff.] und [Vornholt et al., 2010, S. 10 f.]).



(a) Concurrent Engineering (b) Simultaneous Engineering

Abbildung 2.6: Möglichkeiten zur Parallelisierung von Prozessen (angelehnt an [Anderl u. Trippner, 2000, S. 14])

Concurrent Engineering: Beim Concurrent Engineering wird ein Prozess in mehrere Teilprozesse zerlegt, die unabhängig voneinander parallel bearbeitet werden können. Am Ende werden die Teilergebnisse zu einer Gesamtlösung zusammengeführt. Die Einteilung kann nach funktionalen (Fahrwerk, Antriebssystem, Bremssystem) oder strukturellen Aspekten (Produkt, Baugruppe, Einzelteil) sowie nach Domänen (Mechanik, Elektronik, Software) erfolgen.

Simultaneous Engineering: Beim Simultaneous Engineering werden unterschiedliche Prozesse überlappt und teilweise parallel ausgeführt. Sobald ausreichende Ergebnisse oder Teilergebnisse eines Prozesses vorliegen, kann mit einem weiteren Prozess parallel angefangen werden. Dadurch wird, zum einen durch die Parallelisierung und zum anderen durch eine frühzeitige Rückmeldung über mögliche Fehler, Zeit eingespart. Nicht nur innerhalb der Produktentwicklung, sondern auch zwischen der Phase der Produktentwicklung und -herstellung wird versucht, möglichst viele Prozesse mit Hilfe von »Simultaneous Engineering« zu parallelisieren. Daher ist eine klare Trennung zwischen diesen beiden Phasen, wie sie im allgemeinen Ablauf in Abschnitt 2.1.4 beschrieben wurde, in der praktischen Anwendung nicht mehr üblich.

Durch die Parallelisierung entstehen komplexe Prozesse, die gesteuert werden müssen. Auf Grund der stärkeren Zusammenarbeit wächst der Bedarf an Kommunikation und Abstimmung zwischen den beteiligten Personen. Zusätzlich müssen häufiger Daten ausgetauscht werden, weshalb ein reibungsloser Datenaustausch erforderlich ist.

2.4.2 Workflowmanagement-Systeme

Zur Unterstützung der Steuerung von komplexen Prozessen können »Workflowmanagement (WFM)-Systeme« eingesetzt werden. Bevor der Begriff WFM-System definiert und erläutert wird, werden zunächst die Begriffe »Geschäftsprozess« und »Workflow« definiert, weil sie für das Verständnis von Bedeutung sind.

- **Geschäftsprozess:** Ein Geschäftsprozess ist in Anlehnung an die Definition des Begriffs Prozess (siehe Abschnitt 2.1.2) die: »Folge von ... Aktivitäten im betrieblichen Geschehen, die zur Realisierung und Aufrechterhaltung der unternehmerischen Aufgaben und des laufenden Geschäfts beitragen, wobei eine ... Aktivität durch ein Ereignis oder mehrere Ereignisse gestartet und beendet wird. Die Einzelaktivitäten stehen in einem logischen Zusammenhang zueinander; sie sind inhaltlich abgeschlossen. [Krause et al., 2002, S. 265]«
- **Workflow:** »Die Automatisierung eines Geschäftsprozesses im Teil oder im Ganzen. Während des Workflows werden unter Beachtung von prozeduralen Regeln Dokumente, Informationen oder Aufgaben zum Zwecke der Bearbeitung von einem Teilnehmer zum nächsten weitergereicht. [Krause et al., 2002, S. 276]«
- **WFM-System:** »System, das Arbeitsabläufe unter Berücksichtigung von Ressourcen, Terminen und Kosten mit Hilfe von Software definiert, steuert und ausführt. [Krause et al., 2002, S. 276]«

WFM-Systeme bieten eine Unterstützung von Geschäftsprozessen, um den richtigen Personen, zur richtigen Zeit die richtigen Daten bereitzustellen. Dafür müssen die Prozesse zunächst im WFM-System definiert werden. Anschließend können Instanzen der Prozesse vom WFM-System gesteuert und ausgeführt werden. WFM-Systeme bieten sich vor allem zur Unterstützung von gut strukturierten, vorgeplanten und sich häufig wiederholenden Geschäftsprozessen an [Reichert u. Dadam, 2000].

Die Prozesse in der Produktentwicklung stellen auch Geschäftsprozesse dar. Im Gegensatz zu anderen Geschäftsprozessen, wie zum Beispiel in der Produktion und Logistik, lassen sich die Prozesse in der Produktentwicklung nach [Meerkamm et al., 2009, S. 8 f.] aber nicht immer vorausplanen. Der genaue Prozess ergibt sich oft erst im Laufe der Entwicklung. Außerdem ist die Entwicklung eines Produktes durch Kreativität und dynamische Abläufe (zum Beispiel Iterationen) geprägt. Durch eine Vorgabe des Entwicklungsprozesses würde zum einen der kreative Spielraum der Entwickler und zum anderen die dynamische Reaktionsfähigkeit eingeschränkt werden.

Es gibt aber Teilprozesse während der Produktentwicklung, die vorgeplant und damit automatisiert werden können, wie zum Beispiel Freigabe- und Änderungsprozesse (siehe

[Eigner u. Stelzer, 2009, S. 103 ff.]). Dafür werden in der Produktentwicklung in der Regel keine eigenständigen WFM-Systeme eingesetzt, sondern »Produktdatenmanagement (PDM)-Systeme« mit integrierten WFM-Komponenten [VDI 2219, 1993]. PDM bezeichnet nach [Eigner u. Stelzer, 2009, S. 31]:

> »... das Management von produktdefinierenden Daten ... in Verbindung mit der Abbildung und dem Management von technischen/organisatorischen Geschäftsprozessen ... «

PDM-Systeme dienen zur Umsetzung dieses Management-Ansatzes. Sie verwalten die Daten und Dateien (Produktdaten), die während der Produktentwicklung von CAx-Systemen und weiteren Software-Systemen erzeugt werden. Außerdem wird eine Verbindung zwischen den Produktdaten und den zur Erzeugung, Bearbeitung und Weiterleitung notwendigen Prozessen ermöglicht.

Aus organisatorischen Gründen ist es notwendig, dass in einem Unternehmen auch der gesamte Produktentwicklungsprozess im Groben geplant werden kann. Für die Unterstützung des gesamten Produktentwicklungsprozesses wurde im »Bayerischen Forschungsverbund FORFLOW für Prozess- und Workflowunterstützung zur Planung und Steuerung der Abläufe in der Produktentwicklung« [Meerkamm et al., 2009; Krehmer et al., 2010] ein sogenannter Prozessnavigator entwickelt. Der Prozessnavigator baut auf den Ansätzen des WFM auf und kann einen Produktentwickler in Abhängigkeit des Entwicklungsstandes und der Produktkomplexität durch die Entwicklungssituationen leiten und bei Entscheidungsfindungen unterstützen. Mit dem Prozessnavigator wird die notwendige flexible Prozessausführung unterstützt, um so dem Entwickler maximale Freiheiten bei gleichzeitig systematischer Unterstützung zu ermöglichen.

2.5 Zusammenfassung

In diesem Kapitel wurden die wesentlichen Grundlagen zum Verständnis der Arbeit beschrieben, die an dieser Stelle noch einmal zusammengefasst werden. Die Produktentwicklung beschreibt den gesamten Prozess von der ersten Idee bis zur Freigabe für die Fertigung, bei dem alle zur Nutzung und Herstellung des Produktes benötigten Unterlagen erstellt werden. Heutzutage wird in den Unternehmen versucht eine VPE zu realisieren. Zur Unterstützung der Produktentwicklung wird eine Vielzahl an rechnerunterstützten Systemen eingesetzt, die zusammenfassend als CAx-Systeme bezeichnet werden. Die Kernkomponente zur Realisierung einer VPE stellen CAD-Systeme dar, mit denen eine dreidimensionale Beschreibung eines geplanten Produktes möglich ist. Zusätzlich zu den geometrischen Informationen lassen sich weitere Informationen, wie zum Beispiel Material- und Fertigungsinformationen im CAD-System definieren. CAD-Systeme ermöglichen daher die digitale Beschreibung eines geplanten Produktes in einem rechnerinternen Modell. Die erstellten CAD-Modelle können für verschiedene Problemstellungen weiterverwendet werden.

Das Ziel einer VPE ist die durchgehende Rechnerunterstützung der Produktentwicklung und intensive Anwendung von Simulationen auf Basis von digitalen Modellen. Die Untersuchung an physischen Prototypen soll durch virtuelle ersetzt werden, um Kosten und Zeit

einzusparen. Außerdem sollen einmal erzeugte Daten digital weitergegeben werden. Dafür muss ein Datenaustausch zwischen verschiedenen CAx-Systemen ermöglicht werden. Es wurden zwei grundsätzliche Ansätze beschrieben, um Schnittstellen zwischen CAx-Systemen zu realisieren. Mit systemspezifischen Schnittstellen können mit Hilfe eines Direktkonverters Daten mit einem geringem Verlust in ein anderes CAx-System übertragen werden. Der Nachteil ist aber, dass bei einer Anzahl von n CAx-Systemen $n \cdot (n-1)$ Konverter benötigt werden. Bei der Verwendung einer neutralen Schnittstelle beziehungsweise von neutralen Dateiformaten kann die Anzahl auf $2 \cdot n$ Konverter reduziert werden.

Weiterhin können zur Unterstützung von Prozessen in der Produktentwicklung WFM-Systeme beziehungsweise WFM-Komponenten von PDM-Systemen eingesetzt werden. Das Problem ist aber, das sich nicht der gesamt Produktentwicklungsprozess unterstützen lässt, weil dieser durch Kreativität und dynamische Abläufe geprägt ist. Der genaue Prozess ergibt sich dadurch erst während der Entwicklung.

3 Analyse

Um die in der Problemdefinition in Abschnitt 1.1 beschriebenen Probleme beim dateibasierten Datenaustausch in der Produktentwicklung genauer zu untersuchen, wurden drei verschiedene Analysen durchgeführt. Die Ergebnisse dieser Analysen werden in diesem Kapitel beschrieben. Als erstes wurde der dateibasierte Datenaustausch zwischen verschiedenen CAx-Systemen bei einem Entwicklungsprozess von Industrierobotern untersucht, um Probleme, die in Verbindung mit dem Datenaustausch auftreten, zu ermitteln. Dafür wurden die Vorgehensweise zum Datenaustausch bei dem Entwicklungsprozess und die dabei erzeugten Daten im Detail untersucht. Die Ergebnisse dieser Analyse werden in Abschnitt 3.1 beschrieben.

Zwei Probleme, die sich bei der Analyse herausgestellt haben, wurden durch zwei vertiefende Analysen detailliert untersucht. Zum einen wurde ein Vergleich von Dateiformaten zum Austausch von CAD-Daten durchgeführt, um Unterschiede von Dateiformaten zu ermitteln. Zum anderen wurde eine Analyse zur Nachvollziehbarkeit des Datenaustauschs durchgeführt, um herauszufinden, welche Daten im Zusammenhang mit der Nachvollziehbarkeit von Bedeutung sind. Die Ergebnisse der beiden Analysen werden in Abschnitt 3.2 und 3.3 beschrieben.

3.1 Datenaustausch bei einem Entwicklungsprozess von Industrierobotern

Um die Vorgehensweise und die auftretenden Probleme beim dateibasierten Datenaustausch genauer zu untersuchen, wurde ein konkreter Prozess bei der Entwicklung von Industrierobotern analysiert, bei dem Daten zwischen verschiedenen CAx-Systemen in Form von Dateien ausgetauscht werden. Der Prozess wird am Fraunhofer-Institut für Fabrikbetrieb und -automatisierung IFF in Magdeburg (kurz Fraunhofer IFF) durchgeführt, um das kinematische und dynamische Verhalten von zu entwickelnden Industrierobotern mit Hilfe von Mehrkörpersystem-Simulationen zu analysieren.

Zur Untersetzung der Analyse wurden Diagramme mit der »Unified Modeling Language (UML)« in der Version 2.3[1] erstellt. Die UML ist eine grafische Modellierungssprache, die auf objektorientierten Konzepten aufbaut. Sie wurde verwendet, weil sie weit verbreitet ist und die Standardmodellierungssprache für die Analyse und den Entwurf von Software-Systemen darstellt [Oestereich, 2009]. Zum besseren Verständnis des analysierten Prozesses beziehungsweise von Mehrkörpersystem-Simulationen wird im folgenden Abschnitt zunächst erläutert was unter Mehrkörpersystemen zu verstehen ist.

[1] `http://www.omg.org/spec/UML/2.3/` (zuletzt überprüft am 10.04.2011)

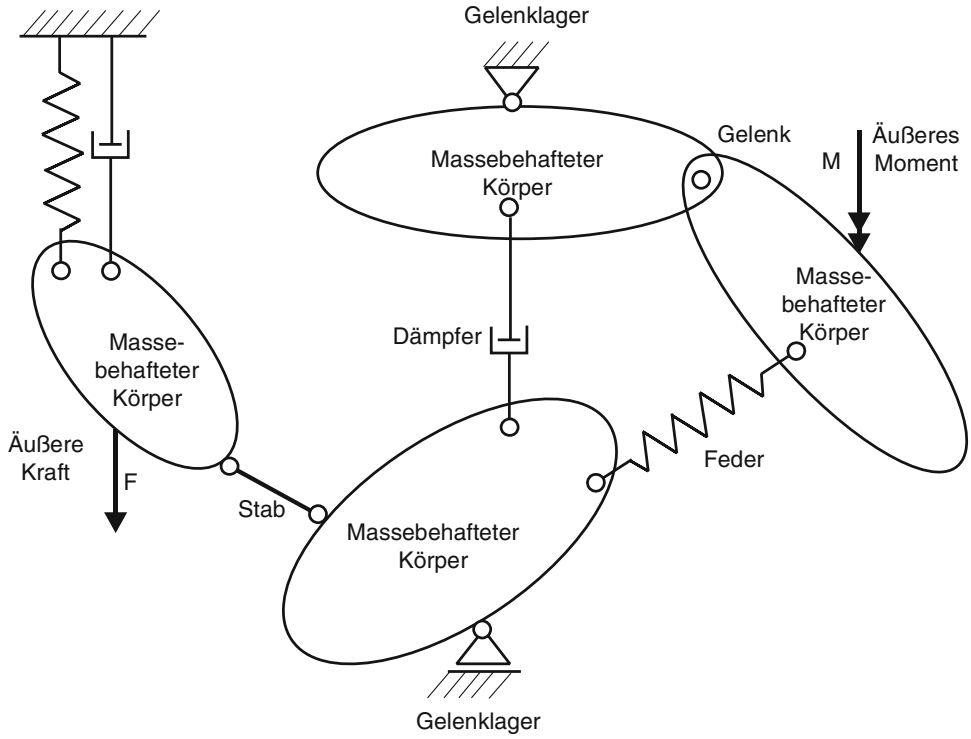


Abbildung 3.1: Modellvorstellung eines Mehrkörpersystems [Vajna et al., 2009, S. 287]

3.1.1 Mehrkörpersysteme

Mehrkörpersysteme (MKS) sind Modelle, die zur Untersuchung des kinematischen und dynamischen Verhaltens von komplexen technischen Produkten, wie zum Beispiel Maschinen, Fahrzeuge und Roboter, eingesetzt werden [Shabana, 2005]. Ein MKS besteht aus einer endlichen Anzahl von massebehafteten starren Körpern, die durch masselose Koppelelemente miteinander verbunden sind. Über diese Koppelelemente werden Kräfte und Momente zwischen den einzelnen Körpern übertragen, die an diskreten Punkten der Körper angreifen. Außerdem können auf die Körper auch von außen Kräfte und Momente wirken. Zur Veranschaulichung ist in Abbildung 3.1 ein Beispiel für ein MKS-Modell dargestellt.

In der Abbildung sind verschiedene Koppelelemente zwischen den massebehafteten Körpern dargestellt. Sie können nach [Vajna et al., 2009, S. 286] in zwei Gruppen eingeteilt werden. Durch die eine Gruppe von Koppelelementen (Gelenk, Führung, Lager) werden die Bewegungsmöglichkeiten, der durch sie verbundenen Körper, eingeschränkt und bei der anderen Gruppe (Feder, Dämpfer) nicht.

Nach [Roddeck, 2006, S. 75 ff.] werden die einschränkenden Koppelelemente als starr und die anderen als nicht starr bezeichnet. Bei Kopplungen, die nicht starr sind, wirken Kräfte zwischen den Körpern, die berücksichtigt werden müssen. Starre Kopplungen schränken die Anzahl der Freiheitsgrade ein, da zwischen den Koordinaten, welche die Lage der Körper beschreiben, feste Beziehungen, sogenannte Zwangsbedingungen, bestehen. Ein starrer Körper im Raum hat einen Freiheitsgrad von sechs, der sich aus drei Translations-

und drei Rotationsfreiheiten ergibt. Die Einschränkung der Freiheitsgrade durch starre Kopplungen wirkt sich auch auf den Freiheitsgrad des Gesamtsystems aus. Dieser entsteht aus $6N - r$, wobei N die Anzahl der Körper und r die Anzahl der Zwangsbedingungen ist.

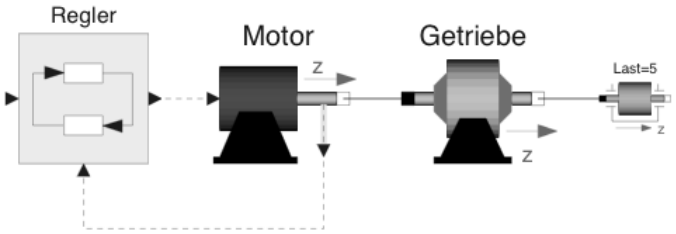
In einem MKS-Modell werden die Körper auf die zur Untersuchung des kinematischen und dynamischen Verhaltens wesentlichen Eigenschaften reduziert. Es müssen die Masse, die Lage des Schwerpunktes und die drei Trägheitsmomente in alle drei Drehrichtungen des Körpers definiert werden. Durch die Trägheitsmomente wird das Trägheitsverhalten des Körpers beschrieben, welches von der Gestalt des Körpers, der Massenverteilung und der Drehachse abhängt. Die genaue Gestalt des Körpers muss nur definiert werden, wenn Kollisionen berücksichtigt werden sollen.

Um ein MKS zu untersuchen, muss es in ein entsprechendes mathematisches Modell überführt werden. Für überschaubare Systeme kann dies auch ohne Rechnerunterstützung erfolgen. In der Praxis ist es aber auf Grund der Komplexität der MKS zu aufwändig. Daher gibt es CAE-Systeme (siehe Abschnitt 2.2.4) zur Durchführung von MKS-Simulationen. In einem CAE-System muss zunächst das MKS-Modell definiert werden. Anschließend wird ein mathematisches Modell, bestehend aus einer Menge von Bewegungsgleichungen in Form von Differentialgleichungen abgeleitet und mit geeigneten mathematischen Methoden über die Zeit integriert [VDI 2211, 2003, S. 31]. Als Ergebnis der MKS-Simulation erhält man die Bewegungsabläufe und die dabei an den Körpern wirkenden Kräfte und Momente [VDI 2209, 2009, S. 157].

3.1.2 Mehrkörpersystem-Simulationen mit Dymola

Am Fraunhofer IFF wird mit Hilfe von MKS-Simulationen das Bewegungsverhalten von Industrierobotern untersucht. Industrieroboter sind mechatronische Systeme (siehe Abschnitt 2.1.5). Bei den MKS-Simulationen sollen daher nicht nur die mechanischen Komponenten der Roboter, sondern auch ihre elektronischen berücksichtigt werden. Dafür wird das CAE-System »Dymola« eingesetzt, welches ein kommerzielles Modellierungs- und Simulationssystem für integrierte und komplexe Systeme ist. Es basiert auf der objektorientierten Beschreibungssprache »Modelica«, mit deren Hilfe sich physikalische Systeme beschreiben lassen, die aus Komponenten unterschiedlicher Ingenieurdisziplinen (Mechanik, Elektronik, Hydraulik, ...) bestehen (siehe [Elmqvist u. Mattsson, 1997; Elmqvist et al., 1999; Otter u. Elmqvist, 2000]).

Modelica ist kostenlos verfügbar und wird seit 1996 von der Forschergruppe »Modelica Association« entwickelt. Es stehen kostenlose sowie kommerzielle Bibliotheken zur Verfügung, die zum Aufbau von Modellen verwendet werden können. Mit Hilfe von Dymola kann ein Anwender komplexe Modelle grafisch erstellen und mit Modelica-Code beschreiben. Zur Veranschaulichung der Modellierung ist in Abbildung 3.2 ein einfaches grafisches Modell in Dymola mit dem zugehörigen Modelica-Code dargestellt. Das Modell stellt einen Antriebsstrang dar und besteht aus einem Motor, einem Getriebe, einer Last und einem Regelungssystem. Auf Basis eines mit Modelica beschriebenen Modells kann mit Dymola eine Simulation durchgeführt werden. Dymola kann dabei die Ergebnisse, zum Beispiel in



(a) Grafische Darstellung in Dymola

```
model Antriebsstrang
  Control Regler;
  Motor   Motor;
  GearBox Getriebe;
  Shaft   Last (J=5);
equation
  connect(Regler.out, Motor.in);
  connect(Motor.out , Regler.in2);
  connect(Motor.n   , Getriebe.p);
  connect(Getriebe.n, Last.p);
end Antriebsstrang;
```

(b) Zugehöriger Modelica-Code

Abbildung 3.2: Modellierung in Dymola am Beispiel eines Antriebstrangs (angelehnt an [Otter u. Elmqvist, 2000, S. 3])

Form von Diagrammen oder Animationen, visualisieren und kann anschließend auch zur Auswertung der Simulation verwendet werden [Otter u. Schweiger, 2004].

Bei der Entwicklung von Industrierobotern werden CAD-Modelle erstellt, welche Daten enthalten, die für eine MKS-Simulation verwendet werden können. Daher ist es sinnvoll ein MKS-Modell nicht vollständig in Dymola zu erstellen, sondern weitgehend aus einem bestehenden CAD-Modell abzuleiten. Das Problem ist, dass Dymola keine Möglichkeit bietet, Daten für ein MKS-Modell aus einem CAD-Modell abzuleiten. Vom Fraunhofer IFF wurde daher das Software-System »RobotMax« entwickelt. Damit können unter anderem CAD-Modelle in MKS-Modelle konvertiert und als Modelica-Code für Dymola exportiert werden. Zusätzlich können mit RobotMax elektronische Komponenten, wie zum Beispiel Motoren, aus einer vorhandenen Bibliothek hinzugefügt werden. Damit entstehen mechatronische MKS-Modelle, die mit Dymola untersucht werden können.

Die Entwicklung von RobotMax und die Vorgehensweise zur Überführung von CAD-Modellen in Modelica-Modelle wird in [Juhász u. Schmucker, 2008] und [Juhász, 2008] erläutert. Ähnliche Ansätze für die teilautomatische Übertragung von CAD-Modellen in Modelica-Modelle werden in [Engelson et al., 2003; Bhattacharya et al., 2006] beschrieben. In den folgenden Abschnitten wird der analysierte Prozess im Detail erläutert.

3.1.3 Ablauf des Prozesses

Zunächst soll ein grober Überblick über den Ablauf des Prozesses vermittelt werden. Für die MKS-Simulation muss ein MKS-Modell erstellt werden. Dies passiert in zwei Schritten. Als erstes wird ein CAD-Modell eines Roboters für RobotMax aufbereitet. Neben der Geometrie werden zur Erstellung eines MKS-Modells noch weitere Daten benötigt, wie zum Beispiel das Gewicht der Einzelteile. Die Geometrie des Roboters und die weiteren Daten werden an das CAx-System RobotMax übergeben. Als nächstes entsteht aus den CAD-Daten in RobotMax ein mechanisches MKS-Modell, welches mit elektronischen Komponenten zu einem mechatronischen MKS-Modell erweitert wird. Das mechatronische MKS-Modell kann anschließend von RobotMax in Form von Modelica-Dateien exportiert und an Dymola übergeben werden. Die MKS-Simulation kann in Dymola oder mit einem weiteren CAE-System »Matlab« durchgeführt werden.

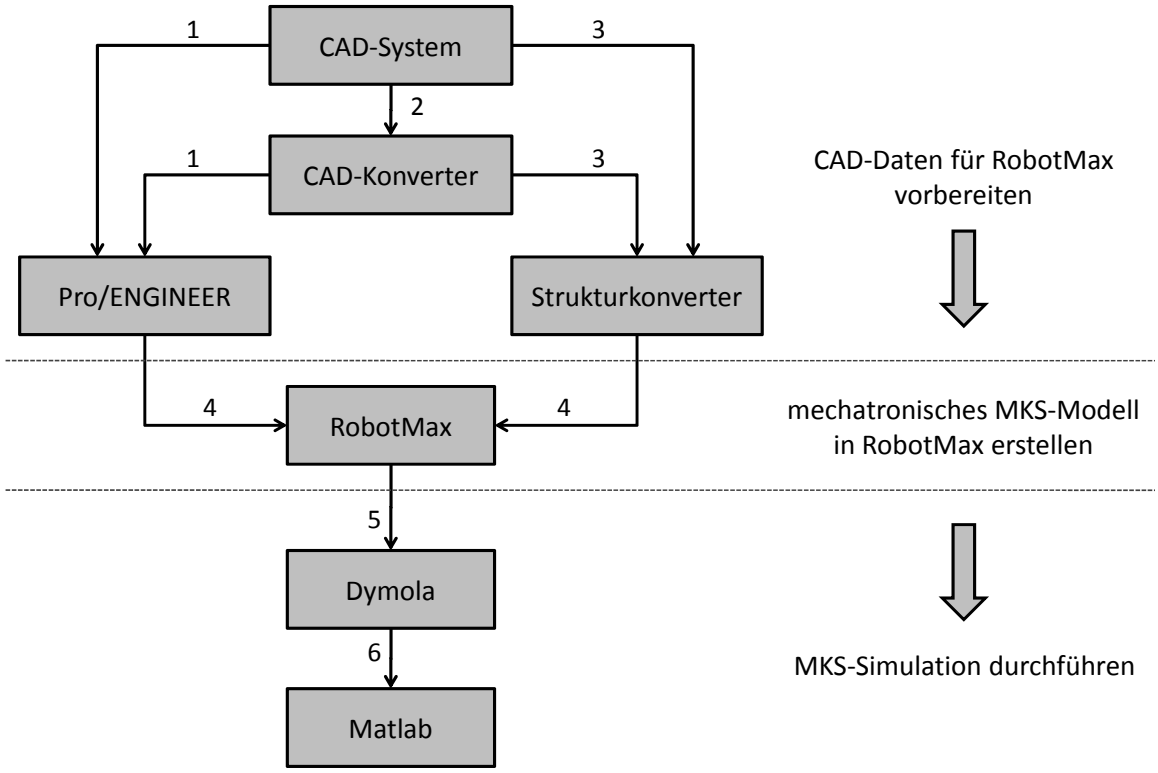


Abbildung 3.3: Beim Prozess am Fraunhofer IFF eingesetzte CAx-Systeme

Tabelle 3.1: Beschreibung der Schnittstellen zwischen den beim Prozess am Fraunhofer IFF eingesetzten CAx-Systemen

Nr.	Dateiformat	Dateianzahl	Inhalt
1	in Pro/ENGINEER importierbare CAD-Formate	1 oder mehrere	3D-Modell
2	vom CAD-Konverter konvertierbare CAD-Formate	1 oder mehrere	3D-Modell
3	STEP	1	3D-Modell
4	VRML	mehrere	3D-Modell
	XML	1	zusätzliche CAD-Daten zur Erstellung des MKS-Modells (Produktstruktur, Masse der Einzelteile, ...)
5	Modelica	mehrere	MKS-Modell
6	C-Code	1	MKS-Modell

Die drei wesentlichen Schritte beziehungsweise Aktivitäten, die beim Prozess durchgeführt werden, sind auf der rechten Seite in Abbildung 3.3 dargestellt. Parallel dazu befinden sich auf der linken Seite die bei den Aktivitäten verwendeten CAx-Systeme. Die Zahlen in der Abbildung bezeichnen die Schnittstellen, welche in der Tabelle 3.1 beschrieben sind. STEP (Standard for the Exchange of Product Model Data) und VRML (Virtual Reality Modeling Language) bezeichnen zwei Dateiformate, welche zum Austausch von CAD-Daten verwendet werden können. Die beiden Formate werden in Abschnitt 3.2.2 beziehungsweise 3.2.6 näher erläutert. Im Folgenden werden die drei Aktivitäten beschrieben.

Aktivität »CAD-Daten für RobotMax vorbereiten«

In der ersten Phase müssen die CAD-Daten für RobotMax vorbereitet werden. RobotMax benötigt mehrere VRML-Dateien und eine XML-Datei. Die VRML-Dateien enthalten die geometrischen Daten der Einzelteile des Roboters. Für den Aufbau eines MKS-Modells in RobotMax werden noch zusätzliche Daten benötigt, welche durch eine XML-Datei übergeben werden. Sie enthält folgende Daten:

- Produktstruktur
- für jedes Einzelteil:
  - Masse
  - Trägheitstensor mit Trägheitsmomenten
  - Position und Orientierung des Schwerpunktes
- Kopplungen
  - Name und ID der beiden Einzelteile
  - Kopplungstyp
  - Position und Orientierung der Kopplung

Die Produktstruktur wird durch eine verschachtelte Beschreibung von Baugruppen und Einzelteilen in der XML-Datei abgebildet. Zu den Einzelteilen werden weitere Daten angegeben. Die Masse, der Trägheitstensor und die Lage des Schwerpunktes ergeben sich durch die Festlegung des Materials der Einzelteile im CAD-System. Die Position im Koordinatensystem wird durch einen (x,y,z)-Vektor und die Orientierung, das heißt die Lage im Koordinatensystem, wird durch eine Matrix definiert. Zusätzlich werden die Kopplungen beziehungsweise Verbindungen zwischen den Einzelteilen beschrieben. Diese resultieren aus den im CAD-System beim Zusammenbau der Einzelteile definierten Zwangsbedingungen (»Constraints«). In der XML-Datei bezieht sich jede Kopplung auf zwei Einzelteile, die durch ihren Namen und eine ID definiert werden. Durch einen Kopplungstyp wird die Art der Verbindung beschrieben. Weiterhin werden durch die Position und Orientierung festgelegt, an welcher Stelle die Einzelteile verbunden sind.

Der Ablauf der Bereitstellung der Daten für RobotMax ist im Aktivitätsdiagramm in Abbildung 3.4 dargestellt. Es existieren zwei verschiedene Möglichkeiten, um die Daten bereitzustellen:

- Zum einen über das CAD-System »Pro/ENGINEER Wildfire (kurz Pro/E)«
- Zum anderen mit einem vom Fraunhofer IFF entwickelten Software-System »Strukturkonverter«

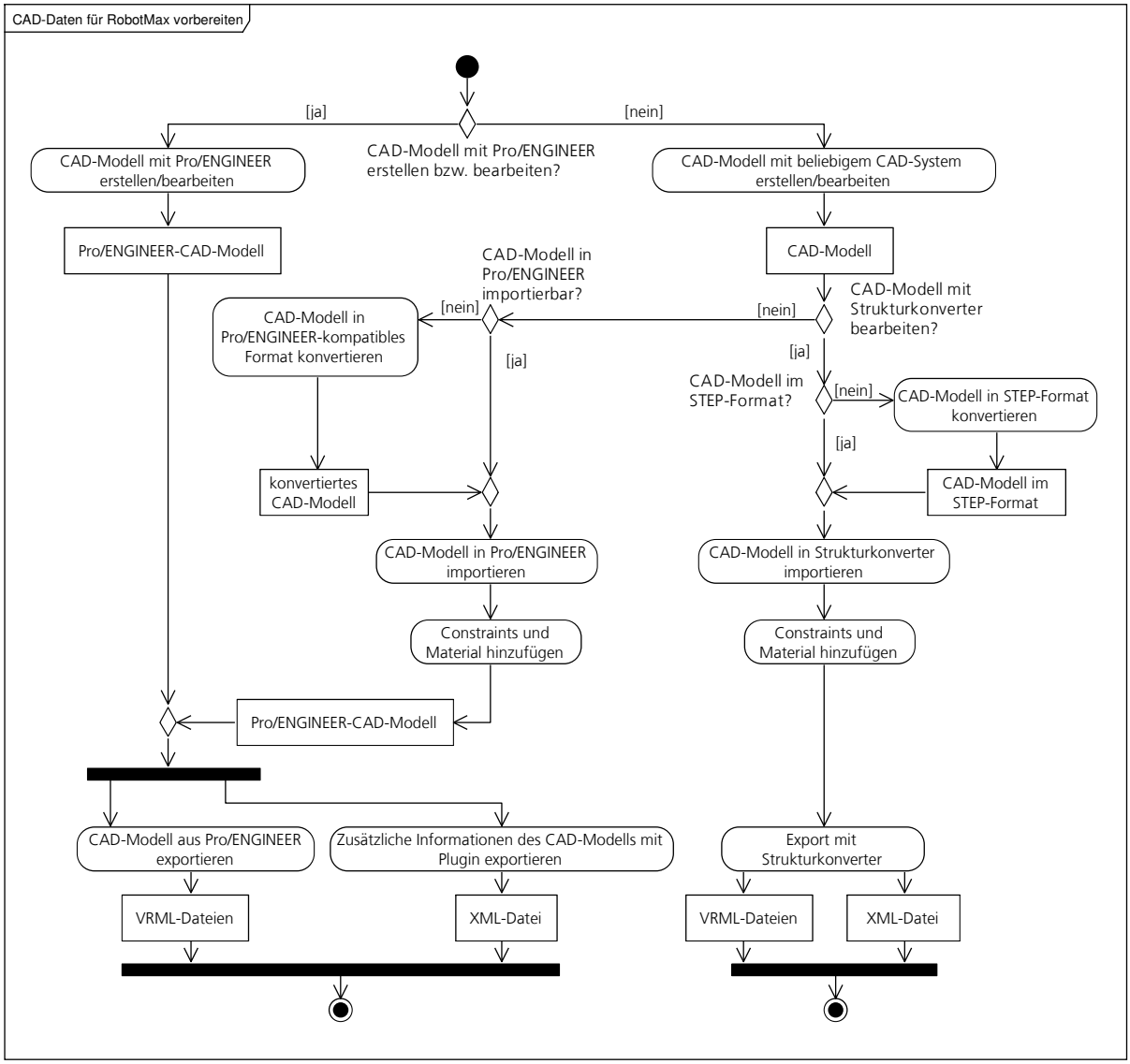


Abbildung 3.4: Aktivitätsdiagramm »CAD-Daten für RobotMax vorbereiten«

Bei beiden Varianten muss zunächst ein CAD-Modell erstellt werden. Dies kann mit Pro/E oder einem beliebigen CAD-System erfolgen. Ein in Pro/E erstelltes CAD-Modell kann in Form von mehreren VRML-Dateien exportiert werden. Für den Export der XML-Datei wird ein spezielles »Plugin« benötigt, das vom Fraunhofer IFF entwickelt wurde.

Das in einem beliebigen CAD-System erstellte CAD-Modell kann entweder in Pro/E oder im Strukturkonverter bearbeitet werden. Zur Bearbeitung in Pro/E muss das CAD-Modell in einem Format vorliegen, dass von Pro/E importiert werden kann. Ansonsten muss es mit Hilfe eines CAD-Konverters in ein unterstütztes Format gebracht werden. Falls im ursprünglichen CAD-System Informationen über die Verbindungen des Zusammenbaus und Materialeigenschaften definiert wurden, gehen diese beim Import in Pro/E aber verloren. Nach dem Import sind in Pro/E daher nur die geometrischen Informationen vorhanden. Die fehlenden Informationen müssen in Pro/E wieder hinzugefügt werden. Wenn das

CAD-Modell inklusive der zusätzlichen Informationen in Pro/E vorhanden ist, kann der Export der VRML-Dateien und XML-Datei erfolgen.

Der Strukturkonverter kann CAD-Modelle im STEP-Format importieren. Von einem beliebigen CAD-System muss entweder ein Export in das STEP-Format erfolgen oder ein CAD-Modell exportiert werden, welches anschließend mit einem CAD-Konverter in das STEP-Format konvertiert wird. In einer STEP-Datei können eine Menge von Baugruppen und Einzelteilen enthalten sein, die hierarchisch strukturiert sind. Eine Information über die Beschränkung der Freiheitsgrade ist nicht vorhanden. Diese wird aber zur Erstellung eines MKS-Modells benötigt. Mit Hilfe des Strukturkonverters lassen sich diese zusätzlichen Informationen hinzufügen. Anschließend können die für RobotMax benötigten VRML-Dateien und die XML-Datei exportiert werden. Der Vorteil des Strukturkonverters ist, dass die zusätzlichen Informationen auch von jemandem hinzugefügt werden können, der sich mit Pro/E nicht auskennt.

Aktivität »mechatronisches MKS-Modell in RobotMax erstellen«

Nachdem die CAD-Daten für RobotMax vorbereitet wurden, kann in RobotMax ein mechatronisches MKS-Modell erstellt werden. Der Ablauf ist im Aktivitätsdiagramm in Abbildung 3.5 dargestellt. Als erstes werden die erzeugten VRML-Dateien und die XML-Datei importiert. In RobotMax entsteht dann zunächst ein »mechanisches MKS-Modell«. Zum Hinzufügen von elektronischen Komponenten stellt RobotMax eine Bibliothek bereit, aus der Motoren ausgewählt werden können.

Als Ergebnis entsteht ein »mechatronisches MKS-Modell«, welches mechanische und elektronische Komponenten enthält. Das mechatronische MKS-Modell kann in Form von mehreren Modelica-Dateien exportiert werden. Das mechanische MKS-Modell wird zusätzlich gespeichert. So können bei Bedarf andere elektronische Komponenten hinzugefügt werden, ohne dass ein erneuter Import von CAD-Daten in RobotMax erfolgen muss.

Aktivität »MKS-Simulation durchführen«

Nachdem ein mechatronisches MKS-Modell mit RobotMax erstellt wurde, kann eine MKS-Simulation durchgeführt werden. Der Ablauf ist im Aktivitätsdiagramm in Abbildung 3.6 dargestellt. Als erstes muss das mit RobotMax erstellte mechatronische MKS-Modell in Dymola importiert werden. Zur Durchführung von MKS-Simulationen kann neben Dymola auch Matlab verwendet werden. Matlab kann keine Modelica-Dateien importieren, aber »C-Dateien«, das heißt Dateien mit Code in der Programmiersprache »C«. Von Dymola kann das MKS-Modell als »C-Datei« für Matlab exportiert werden. Dieser kompliziert erscheinende Weg zur Durchführung einer MKS-Simulation ist durchaus sinnvoll. In [Richert et al., 2004] wird die Modellbildung und Simulation unter Dymola und Matlab verglichen. Das Ergebnis ist, dass sich Dymola besser für die Modellbildung eignet und Matlab seine Stärken bei der Auswertung von Simulationen mit Hilfe von umfangreichen Analysefunktionen hat.

Bevor eine MKS-Simulation durchgeführt werden kann, müssen bei beiden Varianten Einstellungen getroffen werden, wie zum Beispiel das Abbruchkriterium der MKS-Simulation

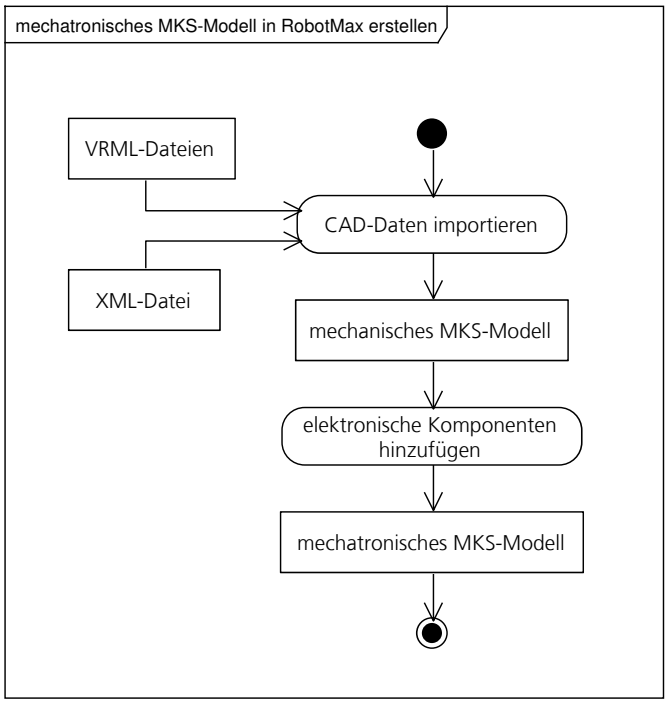


Abbildung 3.5: Aktivitätsdiagramm »mechatronisches MKS-Modell in RobotMax erstellen«

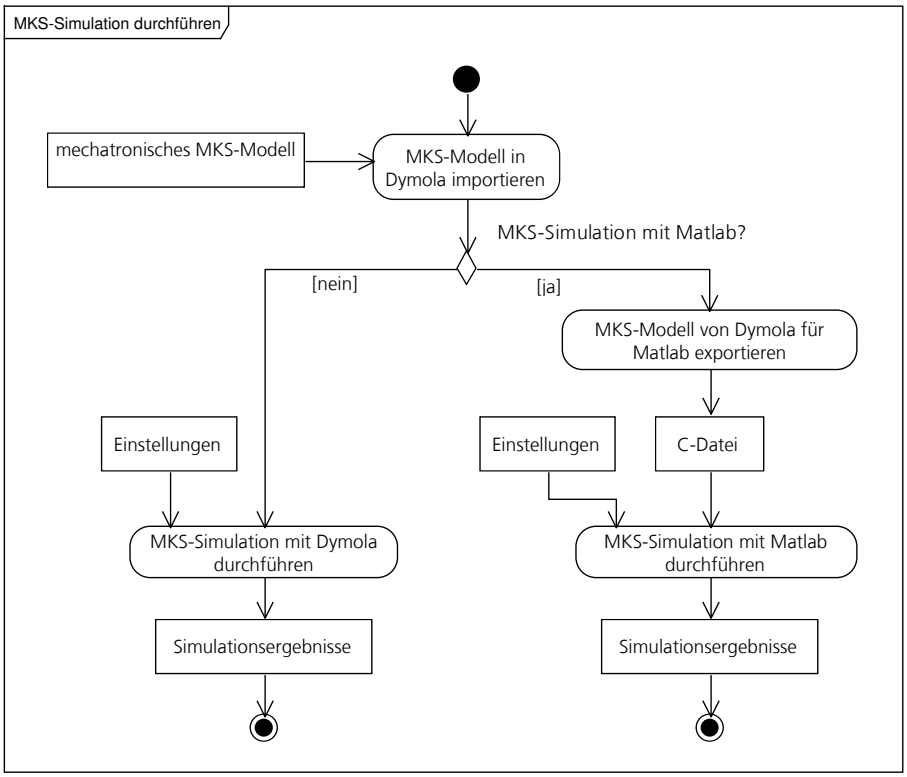


Abbildung 3.6: Aktivitätsdiagramm »MKS-Simulation durchführen«

sowie Art und Umfang der gewünschten Ausgabe der Ergebnisse. Als Resultat entsteht nach der Durchführung der MKS-Simulation eine Menge von Simulationsergebnissen.

3.1.4 Analyse der erzeugten Daten

Nachdem der Ablauf des Prozesses beschrieben wurde, soll in diesem Abschnitt auf die dabei erzeugten Daten eingegangen werden. In Abbildung 3.7 ist ein Klassendiagramm dargestellt, welches die Daten und ihre Beziehungen zueinander beschreibt. Die Klasse »CAD-Daten für RobotMax« beschreibt die CAD-Daten, die an RobotMax übergeben werden müssen, nämlich verschiedene VRML-Dateien und eine XML-Datei. Die zwei unterschiedlichen Möglichkeiten für die Bereitstellung sind durch die spezialisierten Klassen »Strukturkonverter Export« und »Pro/ENGINEER Export« dargestellt. Der Export vom Strukturkonverter basiert auf einem CAD-Modell im STEP-Format, welches gegebenenfalls aus einem weiteren CAD-Modell durch einen Konverter entstanden ist. Ein CAD-Modell wird je nach CAD-Format und Umfang durch eine oder mehrere CAD-Dateien repräsentiert. Für ein CAD-Modell im STEP-Format gibt es zum Beispiel eine »STEP-Datei«. Der Export der CAD-Daten für RobotMax über Pro/E basiert auf einem CAD-Modell im nativen Pro/E-Format, welches in »ASM-« und »PRT-Dateien« abgespeichert wird. Es kann gegebenenfalls auf einem vorhandenen CAD-Modell aufbauen, welches in Pro/E importiert wurde.

Aus den CAD-Daten für RobotMax entsteht zunächst ein mechanisches MKS-Modell, welches in einer oder mehreren Modelica-Dateien abgespeichert werden kann. Aus dem mechanischen MKS-Modell können unterschiedliche mechatronische MKS-Modelle entstehen, indem mit Hilfe von RobotMax unterschiedliche Motoren zum mechanischen Modell hinzugefügt oder die Parameter der Motoren verändert werden. Am mechanischen Modell und an den CAD-Daten ändert sich in diesem Fall nichts.

Mit einem mechatronischen MKS-Modell können verschiedene MKS-Simulationen durchgeführt werden, die sich in den Einstellungen für die Simulation unterscheiden. Ein mechatronisches MKS-Modell kann bei einer Simulation mit Matlab auch noch in einer zusätzlichen C-Datei gespeichert sein. Aus einer durchgeführten MKS-Simulation entstehen Simulationsergebnisse, die in verschiedenen Dateien abgespeichert werden.

Zur Verwaltung der beim Prozess anfallenden Daten und Dateien existiert kein Software-System. Die entstehenden Dateien werden in unterschiedlichen Ordnern auf einem Netzlaufwerk abgelegt.

3.1.5 Schlussfolgerungen

Im Zusammenhang mit dem dateibasierten Datenaustausch treten bei dem analysierten Prozess einige Probleme auf. Ein Problem ist, dass zu den erzeugten Dateien keine beschreibenden Daten (Metadaten) abgelegt werden. Informationen darüber, wie die Dateien entstanden sind, gehen somit verloren. Dadurch kann nicht nachvollzogen werden, mit welchem CAx-System beziehungsweise mit welcher Version eines CAx-Systems eine Datei

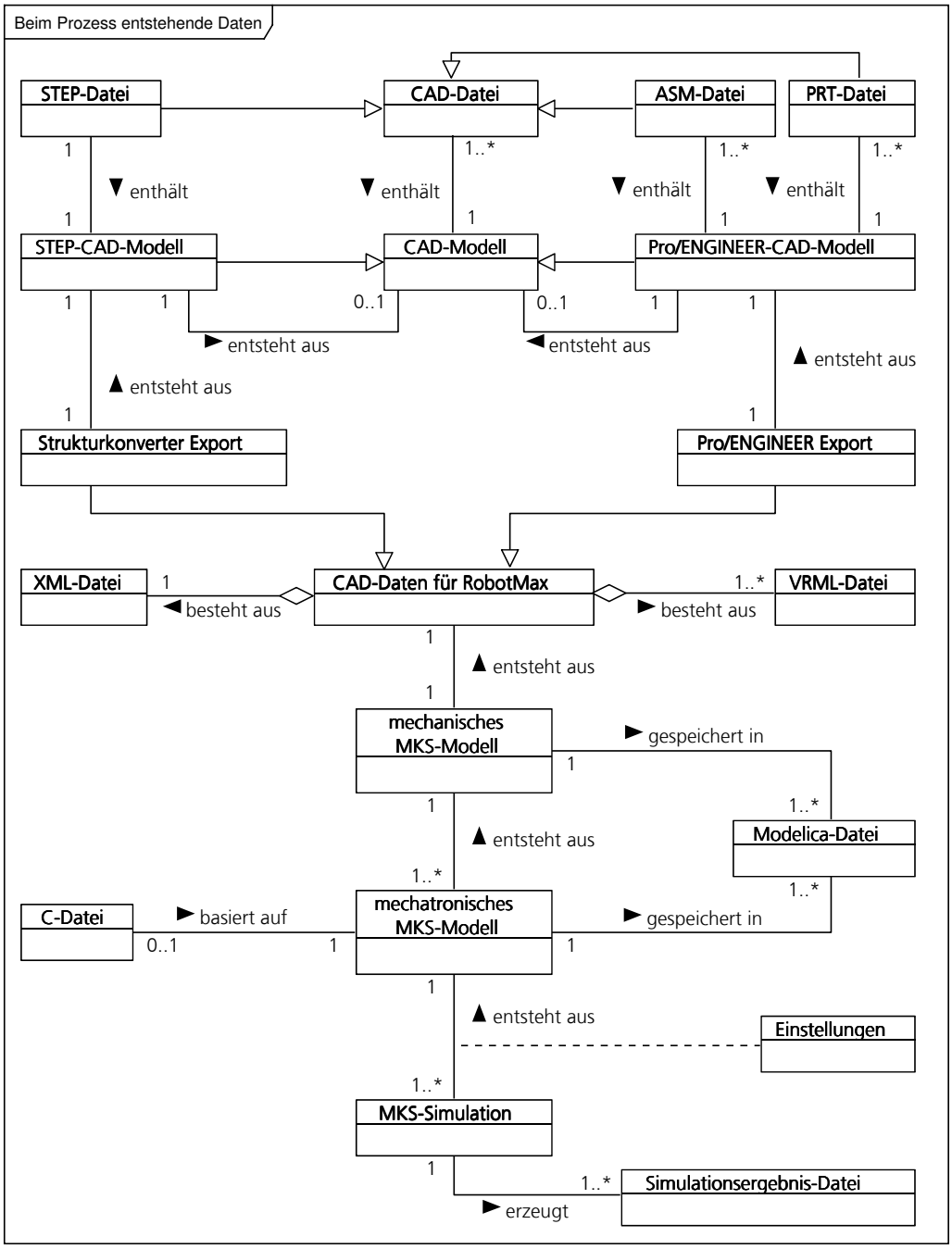


Abbildung 3.7: Klassendiagramm »Beim Prozess entstehende Daten«

erzeugt wurde. Des Weiteren können beim Export von Dateien aus einem CAx-System in der Regel noch Einstellungen getroffen werden, die sich auf den Inhalt der Dateien auswirken. Welche Einstellungen getroffen wurden, wird aber nicht gespeichert. Es wäre daher hilfreich, zusätzliche Daten über den Datenaustausch zu den Dateien mit abzulegen, um die Nachvollziehbarkeit des Datenaustauschs zu verbessern.

Ein weiteres Problem ist, dass es zum Teil alternative Möglichkeiten für den Datenaustausch gibt. Die für RobotMax benötigten Daten können von Pro/E oder dem Struktur-

konverter bereitgestellt werden. Solche alternativen Möglichkeiten können den Anwendern Probleme breiten, da für sie die Unterschiede beziehungsweise die Vor- und Nachteile der Varianten oft nicht ersichtlich sind. Weitere Alternativen ergeben sich, wenn Daten in verschiedenen Dateiformaten übertragen werden können. Dies ist beim analysierten Prozess zum Beispiel beim Austausch von einem beliebigen CAD-System zu Pro/E der Fall. Auch hier ist es für die Anwender oft schwierig zu wissen, welche Daten von welchem Dateiformat übertragen werden können und wo die Vor- und Nachteile liegen. Zusammengefasst ergeben sich alternative Möglichkeiten durch:

- verschiedene CAx-Systeme, die Daten für ein CAx-System bereitstellen können
- verschiedene Dateiformate, die für den Datenaustausch verwendet werden können
- Kombination der beiden Varianten

Weiterhin kann es manchmal notwendig sein, Daten von einem CAx-System zu einem anderen in unterschiedlichen Dateiformaten zu übertragen. Der Grund dafür ist, dass zum Teil durch ein Dateiformat nicht alle benötigten Daten übertragen werden können. Durch verschiedene Dateiformate sollen insgesamt mehr Daten übertragen werden. Beim analysierten Prozess ist dies bei der Bereitstellung der Daten für RobotMax der Fall. Von Pro/E oder dem Strukturkonverter muss ein Export von VRML-Dateien und zusätzlich ein Export einer XML-Datei erfolgen, um die benötigten Daten für RobotMax bereitzustellen. Im Gegensatz zu einem Dateiaustausch, bei dem Dateien in einem Dateiformat von einem CAx-System zu einem anderen übertragen werden, soll ein Datenaustausch, bei dem unterschiedliche Dateiformate benötigt werden, im Folgenden als »gekoppelter Datenaustausch« bezeichnet werden. Es kann auch vorkommen, dass ein CAx-System Daten von mehreren CAx-Systemen benötigt. Der Datenaustausch von mehreren CAx-Systemen zu einem CAx-System wird auch als »gekoppelter Datenaustausch« bezeichnet. Zusammengefasst ergeben sich Kopplungen dadurch, dass:

- Daten von verschiedenen CAx-Systemen übertragen werden müssen
- Daten in mindestens zwei Dateiformaten ausgetauscht werden müssen
- Kombination der beiden Varianten

Für die Anwender in einem Unternehmen wäre es hilfreich, wenn Informationen zu den Alternativen und Kopplungen nachgeschlagen werden könnten.

Die wesentlichen Probleme beim analysierten Prozess sind zum einen die Nachvollziehbarkeit beim Datenaustausch und zum anderen Alternativen und Kopplungen. Zur Nachvollziehbarkeit des Datenaustauschs sind unter anderem Einstellungen beim Import und Export wichtig, die am Beispiel eines CAx-Systems genauer untersucht wurden. Die Ergebnisse dieser Analyse werden in Abschnitt 3.3 beschrieben.

Die Alternativen und Kopplungen ergeben sich neben verschiedenen CAx-Systemen auch durch verschiedene Dateiformate. Um die Unterschiede zwischen Dateiformaten sowie Vor- und Nachteile zur ermitteln, wurde eine weitere Analyse durchgeführt, deren Ergebnisse im folgenden Abschnitt beschrieben werden. Da es eine Vielzahl an Dateiformaten im Zusammenhang mit CAx-Systemen gibt, wurde die Analyse auf Dateiformate zum Austausch von CAD-Daten eingeschränkt. Der Grund dafür ist, dass der Austausch von CAD-Daten in der Produktentwicklung am wichtigsten ist, weil CAD-Systeme die Kernkomponente zur Realisierung einer VPE bilden. In einem CAD-System werden die wesentlichen

Produkteigenschaften definiert und die erzeugten CAD-Daten können von verschiedenen CAx-Systemen weiterverwendet werden (siehe Abschnitt 2.2.4). Der durchgängige digitale Austausch von erzeugten CAD-Daten ist daher besonders wichtig.

3.2 Dateiformate zum Austausch von CAD-Daten

Beim Austausch von CAD-Daten gibt es unterschiedliche Anforderungen an den Umfang der Daten, die übertragen werden sollen. Wenn CAD-Daten in einem anderen CAD-System weiterbearbeitet werden sollen, ist es wichtig, so viele CAD-Daten wie möglich zu übertragen. In anderen Fällen wird nur ein Teil der Daten eines CAD-Modells im empfangenden CAx-System benötigt.

Dass CAD-Daten in einem weiteren CAD-System bearbeitet werden sollen, kann zum Beispiel auftreten, wenn ein CAD-System in einem Unternehmen durch ein neues ersetzt werden soll. Die mit dem alten CAD-System erzeugten Daten sollen bei Bedarf im neuen CAD-System bearbeitet werden können. Durch einen Austausch mit neutralen Dateiformaten ist der Datenverlust aber hoch (siehe Abschnitt 2.3.2). Außerdem kann es zu Übertragungsfehlern, wie zum Beispiel Lücken im 3D-Modell kommen [Cyon Research, 2006, S. 8]. Daher werden oft direkte Konvertierungen benötigt. In [Reim et al., 2007] wird ein Beispiel für ein Konvertierungsprojekt zwischen zwei CAD-Systemen beschrieben. Das Ziel des Projektes war es, so viele Daten wie möglich vom nativen Format des einen CAD-Systems zum anderen zu übertragen. Hierfür wurde jeweils ein unterschiedliches neutrales Dateiformat für die Übertragung von 2D- und 3D-Modellen verwendet. Zusätzlich wurde noch ein Direktkonverter eingesetzt, um Daten zu übertragen, die nicht mit den neutralen Dateiformaten abgebildet werden konnten.

Ein Beispiel, bei dem nur ein Teil der Daten übertragen wird, ist in [Swienczek et al., 1998] beschrieben. Dort werden unterschiedliche CAD-Systeme in der Konstruktion eingesetzt, um verschiedene Stärken der Systeme zu nutzen. In einem der CAD-Systeme müssen aber gesamtheitliche Analysen, wie zum Beispiel Kollisionsanalysen, durchgeführt werden. Hierfür müssen Daten über die Gestalt der CAD-Modelle ausgetauscht werden.

Neben dem Austausch von CAD-Daten zwischen CAD-Systemen können CAD-Daten auch von vielen weiteren CAx-Systemen verwendet werden. Von Bedeutung ist dabei vor allem die Gestalt des CAD-Modells, aber auch weitere CAD-Daten, die in einem CAD-System festgelegt werden können, wie zum Beispiel Materialinformationen und fertigungstechnische Parameter. Welche Daten übertragen werden können hängt davon ab, welche Möglichkeiten zur Abbildung von Daten das verwendete Dateiformat bietet. Um Unterschiede zwischen Dateiformaten zum Austausch von CAD-Daten zu ermitteln, wurden einige neutrale Dateiformate verglichen. Der Grund dafür, dass neutrale Dateiformate und keine nativen verglichen wurden, ist, dass zum Austausch von CAD-Daten in der Regel neutrale Dateiformate eingesetzt werden, weil von den CAx-System-Herstellern bereits Konverter für den Import und Export implementiert sind (siehe Abschnitt 2.3.3). Vor allem CAD-Systeme unterstützen viele neutrale Dateiformate, weil die CAD-System-Hersteller gezwungen sind, diese Interoperabilität zu gewährleisten, da CAD-Systeme die Kernkomponente zur Realisierung einer VPE bilden.

Beim Vergleich der neutralen Dateiformate wurden zum einen die Formate STEP und VRML untersucht, da sie beim analysierten Entwicklungsprozess, der in Abschnitt 3.1.3 beschrieben wurde, verwendet werden. Zum anderen wurden weitere Formate auf der Basis von verschiedenen Quellen, in denen neutrale Dateiformate verglichen werden, ausgewählt (siehe [Grabowski u. Anderl, 1990, S. 48 ff.], [VDI 2209, 2009, S. 119 ff.], [VDI 2211, 2003, S. 50 ff.], [Dyla, 2002, S. 7 ff.], [Anderl u. Grabowski, 2007, S. 24] und [Schuhmann, 2001, S. 17 ff.]). Die Formate IGES, DXF und STL werden beschrieben, da sie in den Quellen am häufigsten vorkommen. Zusätzlich wird noch das Format JT beschrieben, da es sehr verbreitet ist und sich zur Zeit im ISO-Standardisierungsprozess befindet. Zunächst werden die Formate beschrieben und im Anschluss folgt in Abschnitt 3.2.7 ein Vergleich der Formate.

3.2.1 Initial Graphics Exchange Specification (IGES)

IGES ist das erste neutrale Dateiformat zum Austausch von CAD-Daten, welches ab 1979 entwickelt wurde. Im Januar 1980 erschien die erste Version in Form eines Reports des amerikanischen »National Bureau of Standards« [Nagel et al., 1980]. Ein Jahr später wurde diese Version als US-amerikanische Norm »ANSI Y14.26M« vom »American National Standards Institute« veröffentlicht. Damals waren 2D-CAD-Systeme vorherrschend, weshalb zunächst nur 2D-Modelle mit IGES ausgetauscht werden konnten. In späteren Versionen kamen weitere Elemente hinzu, so dass auch 3D-Modelle ausgetauscht werden können. IGES wird seit 1996 auf Grund der Entwicklung von STEP (siehe Abschnitt 3.2.2) nicht mehr weiterentwickelt. Die aktuelle und letzte Version ist »5.3«. Die meisten der heutigen CAD-Systeme und viele weitere CAx-Systeme unterstützen immer noch das IGES-Format. Eine IGES-Datei kann im ASCII- oder Binär-Format erzeugt werden.

3.2.2 Standard for the Exchange of Product Model Data (STEP)

Der Begriff STEP steht als Arbeitstitel für die internationale Normenreihe »ISO 10303 - Product Data Representation and Exchange« [ISO 10303-1, 1994], die sich mit der standardisierten Beschreibung von Produktdaten für deren Austausch befasst (siehe [Anderl u. Trippner, 2000], [Kramer u. Xu, 2009] und [Fowler, 1995]). Die Entwicklung von STEP war die konsequente Fortsetzung der Standardisierungsbestrebungen zum Austausch von Produktbeschreibungen, die mit IGES Ende der 70er Jahre begann. In den 80er Jahren folgten weitere Austauschformate in Form von nationalen Normen (SET, VDA-FS) und Industriestandards (CAD*I, PDDI, PDES), die alle auf den Austausch von Geometriedaten beschränkt waren [Grabowski u. Anderl, 1990, S. 88 ff.]. Mit STEP als internationale Norm sollte einerseits ein Ersatz für die vielen Austauschformate geschaffen werden. Andererseits war nicht mehr nur der Austausch von Geometriedaten das Ziel, sondern von allen Produktdaten, die im Lebenszyklus eines Produktes entstehen [Pratt, 2001]. Die Entwicklung von STEP begann 1984 und nach 10 Jahren erschien 1994 die erste Veröffentlichung von STEP mit einigen grundlegenden Teilen der Normenreihe »ISO 10303«.

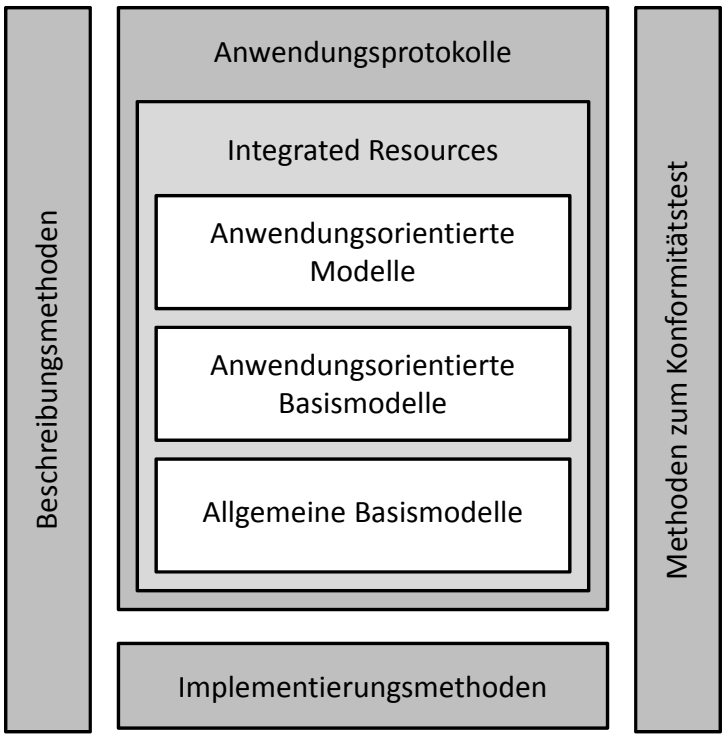


Abbildung 3.8: Aufbau von STEP (angelehnt an [Dyla, 2002, S. 24])

Bis heute sind nach und nach weitere Teile veröffentlicht worden und die Entwicklung ist noch nicht beendet.

Der Aufbau von STEP ermöglicht die ständige Weiterentwicklung, da einzelne Teile hinzugefügt werden können, ohne andere Teile zu beeinflussen. Im Wesentlichen besteht STEP aus Modellen zur Beschreibung von Produktdaten und weiteren Methoden, welche Regeln und Hilfsmittel zur Definition von Modellen umfassen. STEP kann daher als eine Art Baukasten aufgefasst werden, mit dem anwendungsspezifische Produktdatenmodelle (»Anwendungsprotokolle«) unter Verwendung von Grundbausteinen (»Integrated Resources«) nach definierten Regeln und genormten Methoden beschrieben werden können. Der grundlegende Aufbau von STEP ist in Abbildung 3.8 dargestellt und wird im Folgenden erläutert.

Beschreibungsmethoden

In der »10er-Reihe« (ISO 10303-1x) sind die grundlegenden Beschreibungsmethoden definiert. Als Grundlage für STEP wurde die formale Beschreibungssprache »EXPRESS« entwickelt [ISO 10303-11, 2004], welche objektorientierte und prozedurale Konzepte vereinigt und zur Beschreibung von konsistenten, widerspruchsfreien und semantisch eindeutigen Datenmodellen dient. Alle Datenmodelle in STEP sind durch »EXPRESS-Schemata« beschrieben und können mit Hilfe von »EXPRESS-G« grafisch dargestellt werden.

Implementierungsmethoden

Ein mit »EXPRESS« beschriebenes Datenmodell ist unabhängig von einer konkreten Implementierung. In der »20er-Reihe« (ISO 10303-2x) sind zwei grundsätzliche Möglichkeiten zur Implementierung beschrieben:

- Zum einen kann STEP als neutrales Dateiformat zum Austausch von Dateien verwendet werden. Dafür müssen in einem CAx-System zwei Konverter implementiert werden, die auf Basis eines in »EXPRESS« beschriebenen Datenmodells eine »STEP-Datei« erzeugen und lesen können. Dadurch können aus den nativen Daten eines CAx-Systems STEP-Dateien im ASCII- [ISO 10303-21, 2002] oder im XML-Format [ISO 10303-28, 2007] für den dateibasierten Datenaustausch erstellt werden.
- Zum anderen gibt es die Möglichkeit eine »STEP-Datenbank« [Han et al., 2002; Kern u. Bøhn, 1997] in Anlehnung an ein »EXPRESS-Schema« zu erstellen. Ein CAx-System kann über ein in STEP genormtes »Application Programming Interface (API)« auf die »STEP-Daten« in der Datenbank zugreifen [ISO 10303-22, 1998]. Das API wird als »Standard Data Access Interface (SDAI)« bezeichnet und ist programmiersprachenunabhängig mit »EXPRESS« beschrieben. Es existieren aber spezielle SDAI für die Programmiersprachen »C«, »C++« und »Java«.

Da die Beschreibungssprache »EXPRESS« für viele Programmierer unbekannt ist, wurde eine Unterstützung der zwei verbreiteten und standardisierten Technologien »Extensible Markup Language (XML)« und »Unified Modeling Language (UML)« in STEP geschaffen [Lubell et al., 2004]. So können die Vorteile beider Technologien im Zusammenhang mit STEP genutzt werden. Neben der Beschreibung von STEP-Daten im XML-Format ist es möglich »EXPRESS-Schemata« in XML zu definieren [ISO 10303-28, 2007]. Außerdem können »EXPRESS-Schemata« auch als UML-Diagramme dargestellt werden [ISO 10303-25, 2005].

Methoden zum Konformitätstest

In der »30er-Reihe« (ISO 10303-3x) werden allgemeine Methoden zur Prüfung der Konformität von Implementierungen gegenüber einer Spezifikation festgelegt. Dies umfasst allgemeine Konformitätskriterien, Testverfahren, Vorgehensweisen zur Testdurchführung und die allgemeine Darstellung der Testdaten.

Integrated Resources

Die »Integrated Resources« stellen die Grundbausteine von STEP dar und setzen sich aus verschiedenen Teilmodellen zusammen:

- **Allgemeine Basismodelle:** Die in der »40/50er-Reihe« beschriebenen »Allgemeinen Basismodelle« bilden die Grundlage für alle weiteren darauf aufbauenden Datenmodelle im Rahmen von STEP. Sie beschreiben anwendungsunabhängige Grundelemente, wie zum Beispiel allgemeine Objekte zur Produktbeschreibung (Namen, Texte, Zeit- und Datumsangaben, Maßeinheiten, ...), Geometrie und Topologie, Produktstruktur und -konfiguration oder Materialeigenschaften.

- **Anwendungsorientierte Basismodelle:** Die in der »100er-Reihe« beschriebenen »Anwendungsorientierten Basismodelle« verwenden Objekte der »Allgemeinen Basismodelle« und verfeinern diese. Das Objekt Gerade kann zum Beispiel zur Beschreibung einer Maßlinie für Zeichnungen verwendet werden. Es entstehen komplexere Basismodelle für bestimmte Anwendungszwecke, wie zum Beispiel Modelle zur Beschreibung von technischen Zeichnungen, zur Abbildung von elektrischen Funktionen und zur Darstellung von kinematischen Strukturen.
- **Anwendungsorientierte Modelle:** Aus den Basismodellen wurden zu Beginn der Entwicklung von STEP die Anwendungsprotokolle definiert. Sie sind aber komplex, überschneiden sich zum Teil und sind nicht gut aufeinander abgestimmt [Feeney, 2002]. Dies führte zur Entwicklung von weiteren »Anwendungsorientierten Modellen«, die in den Reihen »400/500/1000« beschrieben sind. Diese fassen Teilmodelle, wie zum Beispiel verschiedene Geometriemodelle, zusammen, die in verschiedenen Anwendungsprotokollen verwendet werden können.

Anwendungsprotokolle

In der »200er-Reihe« sind auf Basis der Modelle der »Integrated Resources« »Anwendungsprotokolle (AP)« definiert, welche branchen- beziehungsweise anwendungsspezifische Produktdatenmodelle beschreiben. Dafür wurden die für den Anwendungszweck benötigten Basismodelle ausgewählt und durch Spezialisierungen erweitert. Jedes AP definiert außerdem eine Menge von sogenannten »Conformance Classes« [Pratt, 2001]. Damit werden Untermengen des kompletten AP festgelegt.

Die AP bilden die Grundlage für eine Implementierung mit STEP. Ein Konverter muss aufbauend auf einem Schema, dass von einem AP oder einer »Conformance Class« eines AP vorgegeben wird, implementiert werden.

Beispiele für Anwendungsprotokolle sind:

- AP 203 (Configuration Controlled Design) [ISO 10303-203, 2005]
- AP 209 (Composite and metallic structural analysis and related design) [ISO 10303-209, 2001]
- AP 214 (Core Data for Automotive Mechanical Design and Processes) [ISO 10303-214, 2010]

Die Anwendungsprotokolle »AP 203« und »AP 214« werden von den meisten CAD-Systemen für den Austausch von 3D-Modellen unterstützt [VDI 2209, 2009, S. 119]. Dafür sind in den CAD-Systemen Konverter zum Importieren und Exportieren von STEP-Dateien implementiert, welche Daten in der vom »AP 203« beziehungsweise »AP 214« vorgegebenen Struktur enthalten. Das »AP 209« dient zum Austausch von Daten zwischen CAD- und CAE-Systemen, mit denen FEM-Simulationen durchgeführt werden können.

Das »AP 214« deckt die speziellen Anforderungen der Automobilindustrie an ein Produktmodell ab. Es legt ein Datenmodell für die mechanische Sicht auf den Produktlebenszyklus

in der Fahrzeugentwicklung von der Produktdefinition über Styling, Konstruktion, Prototyperstellung, Produktionsplanung bis hin zur Qualitätskontrolle fest [Swienczek et al., 1998, S. 221]. Dazu gehören geometrische und nicht-geometrische Daten des mechanischen Anteils von Bauteilen, Baugruppen, Produkten und Betriebsmitteln. Die »Conformance Class 1« bildet eine Untermenge mit den für die Praxis wichtigsten Objekten des »AP 214«, welche sich in die Gruppen »Produktdatenmanagement«, »Elementstruktur«, »Drahtmodell«, »Flächenmodell« und »Volumenmodell« gliedern.

Fazit

STEP beschreibt nicht ein einziges neutrales Dateiformat, sondern mit Hilfe der verschiedenen AP eine Reihe von anwendungsspezifischen neutralen Dateiformaten. Die Zahl erhöht sich noch durch die in den AP definierten »Conformance Classes«, welche Untermengen der AP beschreiben, die auch als neutrale Dateiformate verwendet werden können. Dadurch ergibt sich auch ein Nachteil von STEP, denn durch die Vielzahl an möglichen neutralen Dateiformaten werden auch verschiedene Konverter benötigt. Welche Daten durch STEP übertragen werden können, hängt vom verwendeten Anwendungsprotokoll ab. Neben geometrischen (2D- und 3D-Modelle) und nichtgeometrischen CAD-Daten (zum Beispiel Material- und Fertigungsinformationen, Produktstrukturen) können mit STEP auch andere Daten als CAD-Daten übertragen werden, wie zum Beispiel FEM-Modelle zwischen CAE-Systemen.

3.2.3 Jupiter Tessellation (JT)

Das JT-Format wurde ursprünglich Mitte der 90er Jahre von der Firma »Engineering Animation, Inc.« entwickelt [Cyon Research, 2006]. Diese wurde 1999 von der Firma »UGS« aufgekauft, die wiederum 2007 von der »Siemens AG« akquiriert und in »Siemens PLM Software« umbenannt wurde. Das JT-Format ist sehr verbreitet und kann daher als Industriestandard bezeichnet werden. Außerdem wurde von »Siemens PLM Software« und dem »ProSTEP iViP Verein«[1] der formelle Prozess eingeleitet, in dessen Rahmen das JT-Format zu einem internationalen ISO-Standard werden soll [Sendler, 2009]. Die Version »8.1b« ist bei der ISO bereits als sogenannte »Public available specification« [ISO/PAS 14306, 2009] verfügbar. Dies ist eine Vorstufe zum ISO-Standard. Die erste Version des JT-ISO-Standards ist nach [Vettermann, 2010] für Ende 2011 geplant. Die Dokumentation der aktuellen Version »9.5« ist bei »Siemens PLM Software« frei verfügbar[2]. Anwendungsfälle für das JT-Format, vor allem in der Automobilindustrie, werden zur Zeit vom »ProSTEP iViP Verein« und dem »Verband der Automobilindustrie« in der Projektgruppe

[1] Der »ProSTEP iViP Verein« ist eine internationale Branchengemeinschaft von Unternehmen aus der Automobil- sowie Luft-und Raumfahrtindustrie, CAx-System-Hersteller und Forschungseinrichtungen. Das Ziel des Vereins ist es, die aus einem weltweiten Entwicklungsverbund resultierenden Aufgaben, wie zum Beispiel Systemintegration, Produktdatenstandardisierung und Prozessmanagement, für die Fertigungsindustrie zu lösen. Dafür werden von den Mitgliedern gemeinsam Lösungen entwickelt und Standardisierungen vorangetrieben. (`http://prostep.org/` (zuletzt überprüft am 10.04.2011))

[2] `http://www.plm.automation.siemens.com/en_us/Images/JT_v95_File_Format_Reference_Rev-A_tcm1023-111987.pdf` (zuletzt überprüft am 10.04.2011)

»JT Workflow Forum« untersucht [Dotzauer u. Martin, 2010]. Als Ergebnis sollen Anforderungen für die Erweiterung von CAx-Systemen zur Unterstützung des JT-Formates abgeleitet werden.

In einer JT-Datei können unterschiedliche Repräsentationen von 3D-Modellen zusammen abgelegt werden. Zum einen tessellierte Dreiecksflächen, wie beim STL-Format (siehe Abschnitt 3.2.5) in verschiedenen Detailierungsstufen. Durch die Detaillierungsstufen wird die Visualisierung von 3D-Modellen unterstützt. Wenn der Betrachter weit entfernt ist, wird eine vereinfachte Darstellung gewählt, um die Performance zu erhöhen. Zum anderen können 3D-Modelle als parametrisiertes Volumenmodell abgelegt werden, welches die Körper und Flächen exakt abbildet. Im Vergleich zu anderen neutralen Formaten verfügt das JT-Format über eine hohe Kompressionsrate zur Speicherung der 3D-Modelle. Dadurch sind JT-Dateien klein, können schnell ausgetauscht werden und große Baugruppen lassen sich gut visualisieren [Deisinger, 2010].

Neben der Geometrie ist das JT-Format auch für die Übergabe an CAP- und CAM-Systeme attraktiv, weil es zusätzliche Fertigungsinformationen, sogenannte »Product Manufacturing Information (PMI)«, wie zum Beispiel Bemaßungen und Oberflächenbeschaffenheit, aufnehmen kann. Diese müssen in einem CAD-System definiert und können beim Export im JT-Format mit abgelegt werden. Auch weitere Informationen wie Licht, Farben, Texturen, Material und andere Eigenschaften lassen sich definieren, wodurch sich das JT-Format auch gut für Visualisierungszwecke, zum Beispiel in VR-Systemen (siehe Abschnitt 2.2.2), eignet.

Insgesamt ist das JT-Format flexibel einsetzbar, weil unterschiedliche Daten abgelegt werden können. Eine JT-Datei enthält aber immer nur die benötigten Daten, die beim Export ausgewählt wurden, so dass die Dateigröße so gering wie möglich gehalten werden kann.

Vergleich von JT mit STEP

In Zukunft wird das JT-Format neben STEP ein weiterer internationaler Standard zum Austausch von Produktdaten sein. Das Ziel ist aber nicht, STEP durch JT zu ersetzten, sondern beide Formate parallel zu verwenden, weil sie jeweils Vor- und Nachteile bieten. Beim »ProSTEP iViP Verein« wird neben JT zur Zeit auch an einem neuen AP für STEP (»AP 242«) gearbeitet, welches die AP 203 und 214 zusammenführen soll [Vettermann, 2010]. Dabei wird versucht JT und STEP aufeinander abzustimmen und für unterschiedliche Anwendungsfälle auszurichten. JT soll vor allem als »leichtgewichtiges Format« für Visualisierungszwecke und Simulationen dienen, wofür nur ein Teil der in einem CAD-System definierten Produktdaten mit Hilfe des JT-Formates ausgetauscht wird. STEP soll für den Austausch von Baugruppen mit einer Vielzahl von zusätzlichen Daten eingesetzt werden und somit möglichst viele der definierten Produktdaten übertragen.

Ein Vergleich zwischen JT und STEP wird in [Ranger, 2007] beschrieben und ist zusammengefasst in Tabelle 3.2 dargestellt. STEP eignet sich gut, um eine Vielzahl von Daten auszutauschen. So können neben CAD-Daten verschiedene Daten im Lebenszyklus eines Produktes ausgetauscht werden, wie zum Beispiel Simulationsdaten. Bei JT ist der Umfang auf die in einem CAD-System definierten Geometriedaten und einige zusätzliche

Tabelle 3.2: Vergleich von STEP und JT

Eigenschaft	STEP	JT
Beschreibungsumfang der auszutauschenden Daten	hoch	niedrig
Eignung für Visualisierungszwecke	niedrig	hoch
Dateigröße	hoch	niedrig
Komplexität	hoch	niedrig
Umfang der Dokumentation	hoch	niedrig
kostenlos verfügbar	nein	ja

Informationen für Visualisierungszwecke und Fertigungsinformationen beschränkt. Für viele Anwendungsfälle ist dies aber ausreichend.

Für Visualisierungszwecke bietet JT eine bessere Unterstützung gegenüber STEP, da sich zum Beispiel Informationen wie Licht und Texturen definieren lassen. Weiterhin können durch die hohe Kompressionsrate von JT auch große Baugruppen effizient visualisiert werden. Zusätzlich werden durch die hohe Kompressionsrate kleinere Dateigrößen im Vergleich zu STEP erreicht. Ein weiterer Nachteil von STEP ist die hohe Komplexität und der daraus resultierende Umfang der Dokumentation, der eine Implementierung erschwert. Die Dokumentation der aktuellen JT-Version 9.5 umfasst 452 Seiten, bei STEP sind es dagegen mehrere 1000 Seiten. Des Weiteren ist JT kostenlos verfügbar und die STEP-Dokumentationen müssen bei der ISO gekauft werden.

3.2.4 Drawing Interchange Format (DXF)

Das DXF-Format ist ein von der Firma »Autodesk« entwickeltes proprietäres und kein genormtes neutrales Dateiformat, sondern hat sich durch die weite Verbreitung zu einem Industriestandard entwickelt. Zu jeder Version des CAD-Systems »AutoCAD« wird das DXF-Format an dessen Anforderungen angepasst. Die Dokumentationen der Versionen des DXF-Formats sind frei verfügbar[1]. Das DXF-Format ist übersichtlich und wird von vielen CAD-Systemen als Export- und Import-Format unterstützt. Es lassen sich 2D- und 3D-Modelle im ASCII-Format austauschen, wobei das DXF-Format hauptsächlich für den Austausch von 2D-Modellen verwendet wird.

3.2.5 Surface Tessellation Language (STL)

Das STL-Format wurde 1989 von der Firma »3D Systems« definiert und dient zur Beschreibung der Oberfläche von 3D-Modellen mit Hilfe von Dreiecksfacetten, was auch als »Tessellation« bezeichnet wird. Für jede Dreiecksfacette werden drei Eckpunkte und eine Flächennormale definiert. Die Genauigkeit eines 3D-Modells im STL-Format ist abhängig von der Anzahl der Dreiecke, die sich außerdem auf die Größe einer STL-Datei auswirken.

[1] `http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=12272454&linkID=10809853` (zuletzt überprüft am 10.04.2011)

Diese kann im ASCII- oder Binär-Format vorliegen, wobei die Größe der Datei im Binär-Format kleiner ist. Zusätzlich zu den Dreiecken können keine weiteren Daten abgelegt werden. Das STL-Format ist nicht standardisiert, aber auf Grund der weiten Verbreitung beim Austausch zwischen CAD- und CAM-Systemen, insbesondere zur Übergabe von Daten an Rapid-Prototyping-Anlagen (siehe Abschnitt 2.2.3), ein Industriestandard.

3.2.6 Virtual Reality Modeling Language (VRML)

Das VRML-Format ist eine Beschreibungssprache für 3D-Szenen, deren Geometrien, Ausleuchtungen und Animationen. Das Format wurde ursprünglich als 3D-Standard für das Internet entwickelt und 1997 als ISO-Standard veröffentlicht [ISO 14772-1, 1997]. Im Jahr 2004 wurde es aber durch das auf XML basierende X3D-Format [ISO 19775-1, 2004] abgelöst. In VRML werden 3D-Modelle wie beim STL-Format mit Hilfe von Dreiecksfacetten beschrieben. Im Vergleich dazu können aber noch weitere Daten abgelegt werden, die zur Unterstützung der Visualisierung dienen. Außerdem ist es wie beim JT-Format möglich, verschiedene Detaillierungsstufen der 3D-Modelle zu beschreiben. In der Produktentwicklung wird das VRML-Format vor allem zum Austausch zwischen CAD- und VR-Systemen verwendet.

3.2.7 Schlussfolgerungen

Im Laufe der Zeit wurde eine Vielzahl an neutralen Dateiformaten zum Austausch von CAD-Daten entwickelt, von denen einige beschrieben wurden. Mit Hilfe der Literaturquellen, auf die bei der Beschreibung der einzelnen Formate verwiesen wurde, wurden die wesentlichen Unterschiede zwischen den Formaten herausgearbeitet. In der Tabelle 3.3 ist als Ergebnis der Analyse ein Vergleich der beschriebenen neutralen Dateiformate dargestellt. Es wird beschrieben, welche neutralen Dateiformate genormt und welche Industriestandards sind. Außerdem ist dargestellt, ob das ASCII-, XML- oder ein Binär-Format zur Speicherung von Daten verwendet wird. Der Vorteil von ASCII- und XML-Dateien ist, dass sie mit jedem Texteditor geöffnet werden können. Binär-Dateien können im Vergleich zu ASCII und XML aber besser komprimiert werden.

Tabelle 3.3: Vergleich von neutralen Dateiformaten

Eigenschaft	IGES	STEP	JT	DXF	STL	VRML
Norm (n) / Industriestandard (i)	n	n	i	i	i	n
ASCII (a) / Binär (b) / XML (x)	a + b	a + x	b	a	a + b	a
Inhalt						
2D-Modelle	✓	✓	–	✓	–	–
3D-Modelle (facettiert)	✓	✓	✓	–	✓	✓
3D-Modelle (parametrisch)	✓	✓	✓	✓	–	–
Nichtgeometrische CAD-Daten	✓	✓	✓	✓	–	✓

Weiterhin ist dargestellt, welche Daten mit welchem Dateiformat ausgetauscht werden können. Dafür wurden vier wesentliche Unterscheidungsmerkmale festgelegt. 2D-Modelle können mit IGES, STEP und DXF übertragen werden. Bei 3D-Modellen wird eine Unterscheidung in facettierte und parametrische vorgenommen. Bei der parametrischen Methode kann die Oberfläche eines Modells mathematisch korrekt abgebildet werden. Im Gegensatz dazu wird bei der facettierten Methode die Oberfläche nur annähernd genau mit Hilfe von Dreiecksflächen beschrieben. Des Weiteren können, außer beim STL-Format, nichtgeometrische CAD-Daten, wie zum Beispiel Material- und Fertigungsinformationen, übertragen werden. Dabei sind je nach Format unterschiedliche CAD-Daten übertragbar.

In [Dyla, 2002, S. 9] und [VDI 2209, 2009, S. 120] werden die Formate IGES, STEP, DXF und weitere neutrale Dateiformate, die in dieser Arbeit nicht beschrieben wurden, detaillierter verglichen. Dabei wird bei den 2D- und 3D-Modelle auf einzelne Elemente eingegangen, die übertragen werden können, wie zum Beispiel Punkte, Flächen und Farben. Weiterhin werden die nichtgeometrischen CAD-Daten, die mit den Formaten übertragen werden können, genauer verglichen.

Im Zusammenhang mit STEP und JT wurde bei der Beschreibung der Formate auch auf aktuelle Entwicklungen verwiesen. Dies zeigt, dass die Entwicklung von neutralen Dateiformaten zum Austausch von CAD-Daten nicht abgeschlossen ist. Der Grund dafür ist, dass sich im Laufe der Zeit die Anforderungen an neutrale Dateiformate ändern, beispielsweise durch die Weiterentwicklung von CAx-Systemen oder das Entstehen von neuen Technologien. Ein Beispiel für eine aktuelle Entwicklung ist neben STEP und JT das neutrale Dateiformat »AutomationML«, dass zur Unterstützung des Datenaustauschs in der Anlagenplanung entwickelt wird [Lüder et al., 2010]. AutomationML verbindet mehrere XML-basierte Standards zu einem neuen Standard. Neben CAD-Daten können weitere Daten, wie zum Beispiel die kinematische Struktur und das automatisierungstechnische Verhalten einer Anlage, beschrieben werden [Brecher et al., 2010, S. 273]. Die Zielstellung der Entwicklung von AutomationML ist die Unterstützung der »Digitalen Fabrik« [Westkämper u. Niemann, 2009; VDI 4449, 2008] durch ein einheitliches Dateiformat. Da die Entwicklung der ersten Version erst vor kurzem abgeschlossen wurde, konnte das Format AutomationML in dieser Arbeit nicht im Detail analysiert werden.

Der Vergleich der neutralen Dateiformate hat gezeigt, dass in Abhängigkeit der Formate unterschiedliche Daten übertragen werden können. Zusätzlich gibt es eine ständige Weiter- und Neuentwicklung von neutralen Dateiformaten. Für die Mitarbeiter in einem Unternehmen ist es schwierig, alle Dateiformate im Detail zu kennen und sich ständig über neue oder weiterentwickelte Dateiformate zu informieren. Für einen reibungslosen Datenaustausch ist es aber von Bedeutung, dass die passenden Dateiformate verwendet werden. Daher würde sich eine Unterstützungsmöglichkeit für die Mitarbeiter anbieten, mit der die Unterschiede sowie Vor- und Nachteile von Dateiformaten nachgeschlagen werden können.

Neben den Problemen, welche durch unterschiedliche Dateiformate entstehen, wurden in Abschnitt 3.1.5 Probleme im Zusammenhang mit der Nachvollziehbarkeit festgestellt. Um diese Probleme genauer zu untersuchen, wurde eine weitere Analyse durchgeführt, deren Ergebnisse im folgenden Abschnitt beschrieben werden.

(a) Produktstruktur in Pro/E (b) 3D-Modell

Abbildung 3.9: Produktstruktur und 3D-Modell eines vereinfachten Industrieroboters

3.3 Nachvollziehbarkeit des Datenaustauschs

Die Analyse zur Nachvollziehbarkeit des Datenaustauschs wurde am Beispiel des CAD-Systems »Pro/ENGINEER Wildfire (kurz Pro/E)« in der Version 5.0 durchgeführt. Es wurde der Export und Import von 3D-Modellen in Pro/E mit Hilfe von neutralen Dateiformaten untersucht. Pro/E kann 42 verschiedene Formate exportieren und 50 importieren. Darunter sind viele neutrale sowie einige native Dateiformate, wie zum Beispiel das Format »CATProduct« des CAD-Systems »CATIA V5«. Für die Untersuchung wurden die neutralen Dateiformate verwendet, die im vorherigen Abschnitt beschrieben wurden. Ein Schwerpunkt der Untersuchung war, welche Einstellungen beim Export und Import möglich sind und welche Auswirkung diese auf die erzeugten Daten haben. Des Weiteren wurde analysiert, welche Einstellungen und Daten über den Prozess im Nachhinein noch anhand der erzeugten Daten nachvollzogen werden können und welche Informationen verloren gehen.

Zur Untersuchung wurde ein 3D-Modell eines vereinfachten Industrieroboters verwendet, das in Abbildung 3.9 (b) dargestellt ist. Pro/E verwendet zwei unterschiedliche native Dateiformate zur Speicherung von CAD-Daten. Zum einen das Format »PRT« (Abkürzung von Part) zur Speicherung von Einzelteilen und zum anderen »ASM« (Abkürzung von Assembly) zur Speicherung von Baugruppen. Eine Baugruppe kann mehrere Einzelteile und/oder Baugruppen enthalten. Ein Produkt besteht aus einer Menge von Baugruppen und Einzelteilen, die eine Struktur bilden, welche als »Produktstruktur« bezeichnet wird. Die Produktstruktur des Roboters ist in Abbildung 3.9 (a) dargestellt. Das Produkt selber wird in Pro/E auch durch eine Baugruppe repräsentiert. Der Roboter ist als oberste Baugruppe der Produktstruktur in der Datei »RV_E2.ASM« gespeichert und besteht aus vier Einzelteilen (BASE, ARM3, ARM4, WRIST) sowie zwei Baugruppen (ARM1, ARM2), die jeweils aus drei weiteren Einzelteilen bestehen. In einer Produktstruktur können Einzelteile und Baugruppen mehrfach vorkommen. Beim Roboter sind zum Beispiel die Einzelteile »ARM1_AUSSEN.PRT« und »ARM2_AUSSEN.PRT« zweifach verbaut.

Tabelle 3.4: Vergleich der Exportmöglichkeiten von neutralen Dateiformaten in Pro/E

Merkmale	IGES	STEP	JT	STL	VRML
3D-Modelltyp/-Elemente auswählbar					
Drahtmodell	✓	✓	–	–	–
Flächenmodell	✓	✓	–	–	–
Volumenmodell	✓	✓	–	–	–
Facetten	✓	✓	✓	–	–
Bezugskurven und -punkte	✓	✓	✓	–	–
Schalen	✓	✓	–	–	–
weitere Einstellungen					
Konfigurationsdatei	✓	✓	–	–	–
Qualität	–	–	–	–	✓
Auswahl der Teile	–	–	–	✓	–
Anzahl Dateien					
1 Datei	✓	✓	–	✓	–
mehrere Dateien	✓	–	✓	–	✓
ASCII (a) / Binär (b) / XML (x)					
theoretisch möglich	a + b	a + x	b	a + b	a
von Pro/E unterstützt	a	a	b	a + b	a
Logdatei wird erstellt	✓	✓	✓	–	–

3.3.1 Export

Als erstes wird der Vergleich der untersuchten Exportmöglichkeiten von Pro/E beschrieben. Die Ergebnisse sind in Tabelle 3.4 dargestellt. Das DXF-Format wird von Pro/E nicht für den Export unterstützt. Daher konnte es nur im Zusammenhang mit dem Import untersucht werden. Um einen Export in Pro/E durchzuführen, muss zunächst ein Dateiformat ausgewählt werden. Im Anschluss öffnet sich ein Dialog, der in Abhängigkeit des ausgewählten Formates bestimmte Einstellungen ermöglicht beziehungsweise erfordert. Bei den Formaten IGES, STEP und JT können die 3D-Modelltypen und -Elemente ausgewählt werden, die exportiert werden sollen, wobei verschiedene Kombinationen möglich sind. Bei STL und VRML wird die Auswahl von Pro/E automatisch getroffen, weil die Formate keine unterschiedlichen Möglichkeiten bieten. Je nach Format sind noch weitere Einstellungen möglich. Bei IGES und STEP können Konfigurationsdateien erstellt werden, um zusätzliche Einstellungen zu definieren. Bei STEP kann dadurch zum Beispiel das zu verwendende Anwendungsprotokoll festgelegt werden. Pro/E unterstützt die Anwendungsprotokolle 202, 203, 209 und 214. Die Konfigurationsdateien werden beim Export mit abgespeichert. Beim STL-Format können die Teile ausgewählt werden, die exportiert werden sollen und beim VRML-Format kann die Qualität des zu erzeugenden 3D-Modells in drei Stufen (hoch, mittel, niedrig) festgelegt werden. Nachdem alle Einstellungen getroffen wurden, kann der Export erfolgen.

Als Ergebnis entstehen eine oder mehrere Dateien im ausgewählten Format und gegebenenfalls eine Konfigurationsdatei sowie eine Logdatei, welche Informationen über den

durchgeführten Export enthält. Die Anzahl der Dateien, die das 3D-Modell im neutralen Dateiformat abbilden, hängt vom ausgewählten Format ab. Beim IGES-Format kann eingestellt werden, ob eine oder mehrere Dateien erzeugt werden sollen. Zur Speicherung von Dateien gibt es bei neutralen Dateiformaten die Formate ASCII, Binär und XML (siehe Tabelle 3.3). Von den möglichen Formaten werden aber von Pro/E nicht alle unterstützt. Bei IGES und STEP können nur Dateien im ASCII-Format erzeugt werden. Bei den Formaten IGES, STEP und JT entsteht beim Export zusätzlich eine Logdatei. Beim JT-Format enthält sie nur das Erzeugungsdatum und die Zeit, die für die Konvertierung benötigt wurde. Bei den Formaten STEP und IGES ist sie umfangreicher und enthält unter anderem Informationen über die verwendete IGES Version beziehungsweise das STEP AP, das Erzeugungsdatum und die IDs der ursprünglichen und erzeugten Datei. Damit wird in den Logdateien aber nur ein Teil der festgelegten Einstellungen abgespeichert.

3.3.2 Import

Der Vergleich der Importmöglichkeiten von neutralen Dateiformaten in Pro/E ist in Tabelle 3.5 dargestellt. In Pro/E gibt es verschiedene Typen von Modellen, die erzeugt werden können (zum Beispiel Einzelteil, Baugruppe und Zeichnung). Beim Import von einigen neutralen Dateiformaten kann der Typ ausgewählt werden, der in Pro/E erzeugt werden soll. Damit die Daten korrekt eingelesen werden, muss der passende Typ ausgewählt werden. Wenn zum Beispiel ein Volumenmodell im IGES-Format importiert und als Typ Zeichnung gewählt wird, erhält man eine leere Zeichnung. Die zu importierende Daten enthält aber keine Informationen über ihren Inhalt. Daher bleibt nur die Möglichkeit alle Typen zu testen, wenn der Inhalt nicht bekannt ist. Bei den Formaten JT und STL kann der Typ automatisch von Pro/E bestimmt werden.

Durch die Definition von »Import-Profilen« kann bei einigen Formaten der Import zusätzlich beeinflusst werden. Die Import-Profile sind in Pro/E gespeichert, aber nach dem Import kann nicht nachvollzogen werden, welches Import-Profil gewählt wurde.

Bei den Formaten IGES, STEP und JT kann eine Logdatei erstellt werden, die Informationen über den Import enthält. Es werden aber wie beim Export nicht alle Einstellungen abgespeichert, so dass der Importprozess nicht vollständig nachvollzogen werden kann. Das Problem ist, dass die Einstellungen beim Import sich auf die in Pro/E erzeugten

Tabelle 3.5: Vergleich der Importmöglichkeiten von neutralen Dateiformaten in Pro/E

Merkmale	IGES	STEP	JT	DXF	STL	VRML
Typ in Pro/E						
Einzelteil	✓	✓	–	✓	–	✓
Baugruppe	✓	✓	–	✓	–	✓
Zeichnung	✓	✓	–	✓	–	–
weitere	✓	✓	–	✓	–	–
Import-Profil auswählbar	✓	✓	✓	–	–	–
Logdatei kann erstellt werden	✓	✓	✓	–	–	–

nativen Daten auswirken. Eine exportierte Datei kann dadurch unterschiedlich im importierenden CAx-System eingelesen werden, so dass der Datenaustausch auch von den Einstellungen beim Import abhängt. Das heißt, durch falsche Einstellungen kann der Import misslingen oder ungewünschte beziehungsweise fehlerhafte Daten erzeugen.

3.3.3 Schlussfolgerungen

Ein Ergebnis der Analyse ist, dass sich die Einstellungen beim Import und Export in Abhängigkeit des dabei verwendeten Dateiformats unterscheiden.

Weiterhin lässt sich als Ergebnis der untersuchten Exportmöglichkeiten festhalten, dass die Einstellungen bei keinem der Formate vollständig mit abgelegt werden. Zum Teil werden aber Log- und Konfigurationsdateien erzeugt, die einen Teil der Einstellungen beinhalten. Der genaue Ablauf eines Exportprozesses ist daher im Nachhinein nicht nachvollziehbar. Durch die Log- und Konfigurationsdateien ergibt sich bei IGES und STEP ein Vorteil gegenüber den anderen Formaten. Diese sollten zu den erzeugten Dateien mit abgelegt und nicht gelöscht werden, um die Nachvollziehbarkeit zu verbessern. Im Hinblick auf eine vollständige Nachvollziehbarkeit sollten aber möglichst alle Einstellungen gespeichert werden.

Beim den untersuchten Importmöglichkeiten ist das Ergebnis ähnlich wie beim Export. Die Einstellungen lassen sich nicht vollständig speichern. Nur ein Teil der Einstellungen wird in Form von Logdateien abgelegt. Diese Dateien sollten nicht gelöscht werden, um die Nachvollziehbarkeit eines durchgeführten Datenaustauschprozesses zu verbessern. Außerdem wurde bei der Analyse festgestellt, dass beim Import zusätzliche Informationen über den Inhalt der Dateien hilfreich sind, um die richtigen Einstellungen zum Import vornehmen zu können.

Da die Analyse mit »Pro/ENGINEER Wildfire 5.0«, das heißt mit einem spezifischen CAx-System, durchgeführt wurde, ist es schwierig allgemeine Erkenntnisse abzuleiten. Es ist aber davon auszugehen, dass bei anderen CAD-Systemen die Einstellungen beim Import und Export ähnlich sind. Außerdem wird es in der Regel bei CAx-Systemen Einstellungen beim Import und Export geben. Daher lässt sich aus der Analyse ein allgemeiner Ablauf eines Datenaustauschprozesses ableiten, der in Abbildung 3.10 dargestellt ist. Vom CAx-System »A« sollen Daten exportiert werden. Dafür sind gegebenenfalls Einstellungen notwendig, die zum Teil in Dateien abgespeichert werden können. Nach dem Exportieren entstehen eine oder mehrere Dateien, welche die Daten enthalten, die ausgetauscht werden sollen. Diese Dateien werden vom CAx-System »B« importiert, wobei gegebenenfalls auch Einstellungen notwendig sind. Diese können zum Teil in Dateien abgespeichert werden. Zwischen den nativen Dateien in beiden CAx-Systemen und den auszutauschenden Dateien entsteht eine indirekte Beziehung.

3.4 Zusammenfassung

In diesem Kapitel wurden die Ergebnisse von drei durchgeführten Analysen erläutert. Die wesentlichen Ergebnisse sollen an dieser Stelle zusammengefasst werden. Als erstes wurde

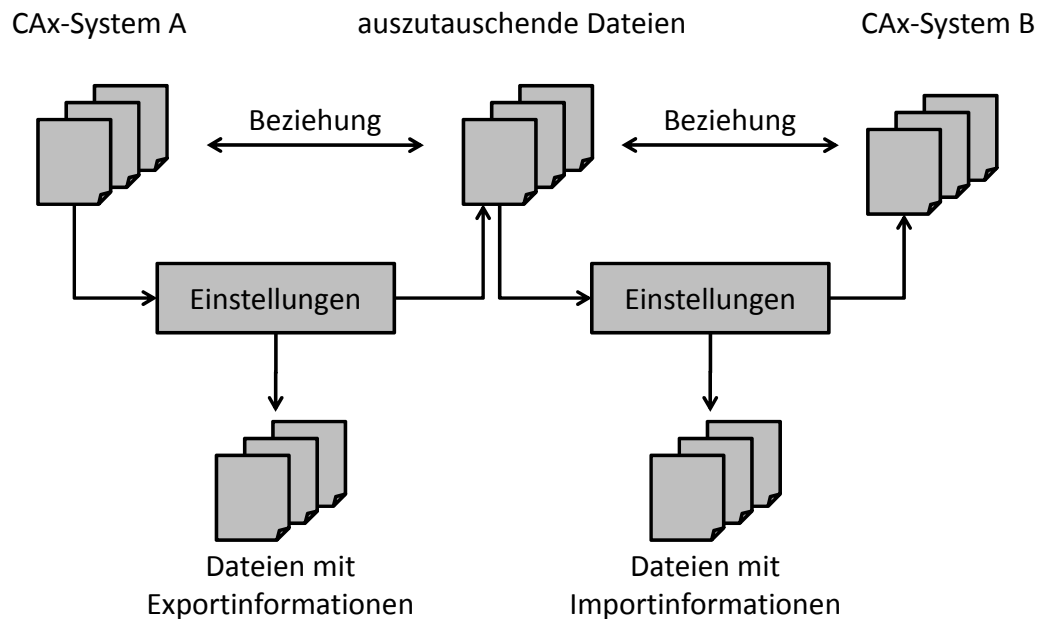


Abbildung 3.10: Ablauf eines Datenaustauschprozesses

der dateibasierte Datenaustausch zwischen verschiedenen CAx-Systemen bei einem Entwicklungsprozess von Industrierobotern untersucht. Aus der Analyse der Vorgehensweise beim Datenaustausch und den dabei erzeugten Dateien wurden zwei wesentliche Probleme festgestellt:

- Alternativen und Kopplungen beim Datenaustausch
- Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen

Alternativen bezeichnen verschiedene Möglichkeiten, um Daten auszutauschen. Kopplungen entstehen, wenn beim Datenaustausch Daten von verschiedenen CAx-Systemen oder in unterschiedlichen Dateiformaten übertragen werden. Neben Alternativen und Kopplungen ist die Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen ein weiteres Problem. Es werden lediglich die erzeugten Dateien auf einem Netzlaufwerk verwaltet. Weitere Daten darüber, wie die Dateien entstanden sind, werden nicht gespeichert. Im Nachhinein kann dadurch nicht nachvollzogen werden, mit welchem CAx-System beziehungsweise mit welcher Version eines CAx-Systems eine Datei erzeugt wurde. Außerdem können Einstellungen, die beim Export und Import getroffen wurden, nicht nachvollzogen werden. Mit Hilfe von zwei vertiefenden Analysen wurden diese beiden Probleme detailliert untersucht. Die Ergebnisse beider Analysen wurden in diesem Kapitel beschrieben.

Da sich Alternativen und Kopplungen neben CAx-Systemen durch unterschiedliche Dateiformate ergeben, wurde zum einen eine Auswahl von neutralen Dateiformaten zum Austausch von CAD-Daten beschrieben und anschließend verglichen. Dabei hat sich herausgestellt, dass im Laufe der Zeit eine Vielzahl an neutralen Dateiformaten entwickelt wurde, welche unterschiedliche Daten übertragen können. Die Entwicklung ist aber nicht abgeschlossen, sondern wird ständig weitergeführt, da sich die Anforderungen an Dateiformate im Laufe der Zeit ändern. Für die Mitarbeiter in einem Unternehmen ist es daher schwierig, alle Dateiformate im Detail zu kennen und sich ständig über neue oder weiter-

entwickelte Dateiformate zu informieren. Für einen reibungslosen Datenaustausch ist es aber von Bedeutung, dass die richtigen Dateiformate verwendet werden.

Zum anderen erfolgte eine Analyse zur Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen am Beispiel des CAD-Systems »Pro/ENGINEER Wildfire«. Es wurde der Import und Export von verschiedenen neutralen Dateiformaten untersucht, wobei festgestellt wurde, dass im Nachhinein der genaue Ablauf der Import- und Exportprozesse nicht nachvollziehbar ist. Einige durchgeführte Einstellungen werden in Log- und Konfigurationsdateien gespeichert. Dies ist aber nicht bei allen Dateiformaten der Fall. Als ein weiteres Ergebnis der Analyse wurde ein allgemeiner Ablauf für Datenaustauschprozesse und die dabei anfallenden Daten abgeleitet.

4 Konzept

Mit Hilfe der im vorangehenden Kapitel beschriebenen Analysen wurden die in der Problemdefinition in Abschnitt 1.1 beschriebenen Probleme, die im Zusammenhang mit dem dateibasierten Datenaustausch auftreten, im Detail untersucht. Ausgehend von den dabei erarbeiteten Ergebnissen wurde ein Konzept für ein Software-System zur Unterstützung des dateibasierten Datenaustauschs zwischen CAx-Systemen in der Produktentwicklung ausgearbeitet, dass in diesem Kapitel erläutert wird. Der Schwerpunkt des Konzeptes lag auf der Entwicklung eines Datenmodells, welches die zur Unterstützung notwendigen Daten und ihre Beziehungen untereinander beschreibt. Um das Konzept übersichtlich zu gestalten, wurden zwei Teilkonzepte entwickelt:

- Zum einen soll mit dem Teilkonzept **»Systemverwaltung«**, welches in Abschnitt 4.1 beschrieben wird, die Verwaltung der Systemlandschaft eines Unternehmens, das heißt die Verwaltung von CAx-Systemen und deren Schnittstellen, ermöglicht werden, um dadurch den Überblick über die Systemlandschaft zu verbessern.
- Zum anderen sollen mit Hilfe des Teilkonzepts **»Prozessverwaltung«**, welches in Abschnitt 4.2 beschrieben wird, Prozessketten inklusive Alternativen und Kopplungen definiert und durchgeführte Datenaustauschprozesse verwaltet werden können, um die Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen zu ermöglichen.

Für die beiden Teilkonzepte werden jeweils die spezifizierten Anforderungen, ein entwickeltes Datenmodell und Anwendungsfälle erläutert. Zur Untersetzung wird wie bei der Analyse in Abschnitt 3.1 die Modellierungssprache UML in der Version 2.3 verwendet. Die Datenmodelle sind als Klassendiagramme dargestellt. Weiterhin werden Anwendungsfall- und Aktivitätsdiagramme zur Beschreibung der notwendigen Funktionen des zu realisierenden Software-Systems verwendet.

Da zwischen den Datenmodellen beider Teilkonzepte Verbindungen bestehen, ergibt sich ein Gesamtdatenmodell, welches zum Abschluss dieses Kapitels beschrieben wird. Als erstes wird im folgenden Abschnitt das Teilkonzept »Systemverwaltung« erläutert.

4.1 Systemverwaltung

Für die Anwender eines Unternehmens ist es schwierig, den Überblick über die Systemlandschaft zu behalten. Es werden in der Regel viele CAx-Systeme bei der Produktentwicklung eingesetzt. Über die Zeit können neue CAx-Systeme hinzukommen, alte CAx-Systeme ersetzt werden oder es werden bestehende CAx-Systeme durch Softwareupdates aktualisiert. Dadurch ergeben sich neue Schnittstellen und auch Vorgehensweisen für den Datenaustausch. In diesem Abschnitt wird ein Teilkonzept erläutert, dass eine Möglichkeit zur Verwaltung der Systemlandschaft, das heißt zur Verwaltung der CAx-Systeme

und Schnittstellen zwischen den CAx-Systemen, beschreibt. Dafür werden im folgenden Abschnitt zunächst die Anforderungen an die Systemverwaltung beschrieben.

4.1.1 Anforderungen

Zur Verwaltung der Systemlandschaft müssen die vorhandenen CAx-Systeme eines Unternehmens beschrieben werden können. Für die Anwender ist es wichtig zu wissen, welche Export- und Importmöglichkeiten ein CAx-System bietet, das heißt, welche Dateiformate es importieren und exportieren kann. Wie in der Analyse in Abschnitt 3.3 festgestellt wurde, sind dabei je nach Dateiformat unterschiedliche Einstellungen möglich. Diese Einstellungen sollten für die Anwender beschrieben werden, damit sie wissen, was sie jeweils bewirken. Weiterhin sollen Schnittstellen zwischen CAx-Systemen beschrieben werden können, wobei auch ein »gekoppelter Datenaustausch« (siehe Abschnitt 3.1.5) berücksichtigt werden soll. Des Weiteren sollen Änderungen an der Systemlandschaft über die Zeit möglich sein. Alte Zustände der Systemlandschaft sollen aber nicht verloren gehen, so dass die Systemlandschaft zu unterschiedlichen Zeitpunkten nachvollziehbar bleibt.

Aus den textuell beschriebenen Anforderungen an die Systemverwaltung können Anforderungen an ein Datenmodell abgeleitet werden, welches die für die Systemverwaltung notwendigen Daten beschreibt. Folgende Anforderungen werden an das Datenmodell gestellt:

- Beschreibung von CAx-Systemen
- Beschreibung von Export- und Importmöglichkeiten der CAx-Systeme
- Beschreibung von Einstellungen bei den Export- und Importmöglichkeiten
- Beschreibung von Schnittstellen zwischen CAx-Systemen (inklusive Kopplungen)
- Nachvollziehbarkeit der Systemlandschaft über die Zeit ermöglichen (Historie)

Auf der Grundlage dieser Anforderungen wurde ein Datenmodell für die Systemverwaltung erstellt, welches in Abschnitt 4.1.2 beschrieben wird.

Weiterhin lassen sich aus den textuell beschriebenen Anforderungen Funktionen ableiten, welche die Systemverwaltung den Anwendern zur Verfügung stellen muss. Diese sollen den Anwendern einerseits ermöglichen, Daten in der Systemverwaltung anzulegen. Andererseits können verschiedene Funktionen auf Basis der angelegten Daten realisiert werden. Folgende funktionale Anforderungen werden an die Systemverwaltung gestellt:

- Daten in der Systemverwaltung anlegen (CAx-Systeme, Schnittstellen, ...)
- Systemlandschaft zu einem gewünschten Zeitpunkt anzeigen
- Export- und Importmöglichkeiten eines CAx-Systems anzeigen
- Schnittstellen zwischen CAx-Systemen anzeigen

Die funktionalen Anforderungen werden in Abschnitt 4.1.3 durch Anwendungsfälle näher beschrieben.

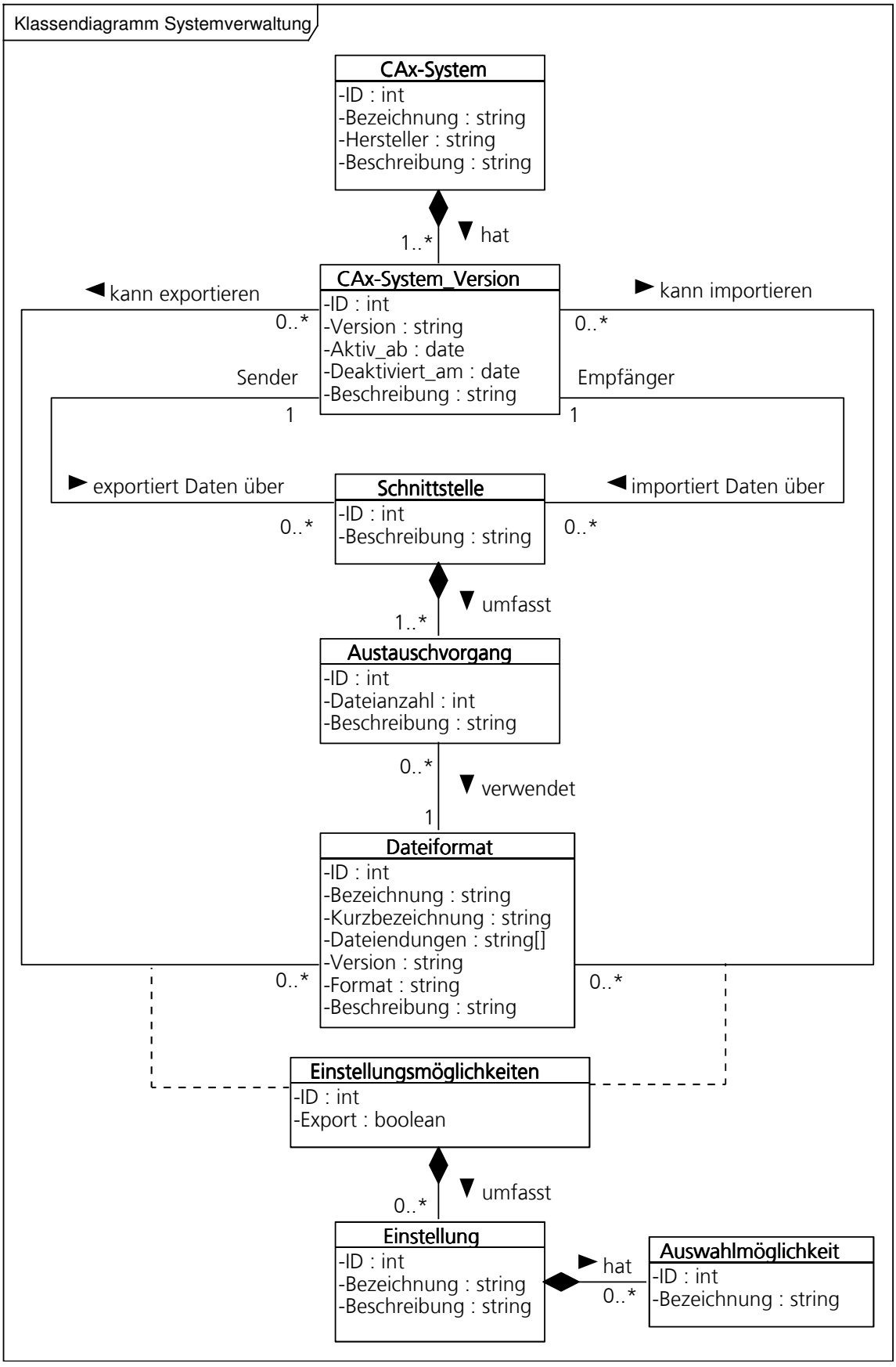


Abbildung 4.1: Klassendiagramm »Systemverwaltung«

4.1.2 Datenmodell

Als Grundlage für die Systemverwaltung wurde ein Datenmodell entwickelt, das in Abbildung 4.1 in Form eines Klassendiagramms dargestellt ist. Im Folgenden werden die Klassen und ihre Beziehungen zueinander erläutert.

Von einem CAx-System können über die Zeit verschiedene Versionen verwendet werden. In Abhängigkeit von einer Version gibt es spezifische Eigenschaften. Mit der Klasse **»CAx-System«** können die unveränderlichen Eigenschaften von verschiedenen Versionen eines CAx-Systems beschrieben werden. Dies sind die Bezeichnung, der Hersteller und eine versionsübergreifende Beschreibung, welche zum Beispiel zur Erläuterung des wesentlichen Verwendungszwecks eines CAx-Systems genutzt werden kann.

Mit der Klasse **»CAx-System_Version«** lassen sich die Versionen eines CAx-Systems mit ihren spezifischen Eigenschaften näher beschreiben. Mit dem Attribut »Version« lässt sich die Bezeichnung der Version, wie zum Beispiel »1.0«, »2.1« oder »2011b«, festlegen. In Abhängigkeit einer Version können unterschiedliche Dateiformate exportiert beziehungsweise importiert werden. Um die Anforderung zu erfüllen, dass die Änderungen an der Systemlandschaft über die Zeit nachvollziehbar bleiben, wird der Zeitraum, in dem eine CAx-System_Version verwendet wird, gespeichert. Mit dem Attribut »Aktiv_ab« wird das Datum festgelegt, ab dem eine Version verwendet werden kann beziehungsweise konnte. Mit dem Attribut »Deaktiviert_am« wird der Zeitpunkt festgelegt, ab dem die Version nicht mehr eingesetzt werden kann beziehungsweise eingesetzt wurde. Mit Hilfe des Attributs »Beschreibung« können zum Beispiel die Besonderheiten einer Version oder wesentliche Änderungen zu den vorangehenden Versionen erläutert werden.

Die Klasse **»Dateiformat«** repräsentiert eine Version eines nativen oder neutralen Dateiformates. Mit dem Attribut »Version« kann die Bezeichnung der Version festgelegt werden. Außerdem können die Bezeichnung des Dateiformates und zusätzlich eine Kurzbezeichnung angegeben werden, die oft ein Akronym der Bezeichnung ist (zum Beispiel IGES, STEP, DXF). Weiterhin können die möglichen Dateiendungen eines Dateiformates festgelegt werden. Wie in Abschnitt 3.2 beschrieben wurde, gibt es unterschiedliche Formate, in denen die Daten beschrieben werden können. Beispiele für diese Formate sind das ASCII-, XML- und Binär-Format. Das bei einem Dateiformat verwendete Format kann durch das Attribut »Format« festgelegt werden. Wenn verschiedene Formate unterstützt werden, müssen mehrere Objekte der Klasse »Dateiformat« angelegt werden. Durch das Attribut »Beschreibung« können zum Beispiel die Besonderheiten eines Dateiformates und welche Daten sich übertragen lassen erläutert werden.

Für jede CAx-System_Version können die Dateiformate definiert werden, die exportiert beziehungsweise importiert werden können. Wie bei der Analyse in Abschnitt 3.3 festgestellt wurde, gibt es in Abhängigkeit einer CAx-System_Version für ein Dateiformat unterschiedliche Einstellungsmöglichkeiten. Diese können durch die Klasse **»Einstellungsmöglichkeiten«** definiert werden. Eine Einstellungsmöglichkeit enthält das Attribut »Export«, womit sich festlegen lässt, ob sich die Zuweisung eines Dateiformates zu einer CAx-System_Version auf den Export oder Import bezieht. Die Unterscheidung ist wichtig, da es beim Export und Import unterschiedliche Einstellungsmöglichkeiten gibt. Wenn ein Dateiformat von einer CAx-System_Version exportiert und importiert werden

kann, muss es daher zweimal zugewiesen werden. Für jede Zuweisung können unterschiedliche Einstellungsmöglichkeiten beschrieben werden.

Als Einstellungsmöglichkeiten können eine, mehrere oder keine **»Einstellungen«** definiert werden. Ein Beispiel für eine Einstellung ist bei Pro/E in der Version 5.0 für das IGES-Format die Anzahl an Dateien, die exportiert werden sollen (siehe Abschnitt 3.3.1). Für die Bezeichnung der Einstellung würde sich zum Beispiel »Anzahl Dateien« anbieten. Mit Hilfe des Attributes »Beschreibung« kann erläutert werden, wofür eine Einstellung notwendig ist beziehungsweise welche Auswirkungen sich durch sie ergeben. Zum Teil gibt es Einstellungen, für die ein Wert eingegeben werden muss. Es kann aber auch vorkommen, dass verschiedene Auswahlmöglichkeiten bei einer Einstellung vorgegeben sind. Mit der Klasse **»Auswahlmöglichkeiten«** können daher zu einer Einstellung eine Menge von Auswahlmöglichkeiten festgelegt werden. Bei der Anzahl der Dateien für das IGES-Format kann in Pro/E zwischen einer oder mehreren Dateien ausgewählt werden. Diese beiden Alternativen könnten als Auswahlmöglichkeiten zur Einstellung »Anzahl Dateien« definiert werden.

Mit der Klasse **»Schnittstellen«** lassen sich Schnittstellen zwischen zwei CAx-System_-Versionen beschreiben. Dabei ist eine CAx-System_Version der Sender, das heißt, sie exportiert Daten für eine andere CAx-System_Version, die als Empfänger bezeichnet wird.

Wie in den Anforderungen beschrieben wurde, soll eine Schnittstelle auch einen »gekoppelten Datenaustausch« unterstützen. Damit ist gemeint, dass Daten von einer CAx-System_Version in verschiedenen Dateiformaten exportiert werden, um dadurch insgesamt mehr Daten zu einer anderen CAx-System_Version übertragen zu können. Dies ist zum Beispiel bei dem in Abschnitt 3.1 analysierten Entwicklungsprozess der Fall. Aus dem CAD-System Pro/E müssen VRML-Dateien und eine XML-Datei exportiert werden, um die für RobotMax benötigten Daten bereitzustellen. Der Austausch von Dateien, welche dasselbe Dateiformat haben, wird im Folgenden als Austauschvorgang bezeichnet.

Der Zusammenhang zwischen einer Schnittstelle und den Austauschvorgängen ist in Abbildung 4.2 dargestellt. Eine Schnittstelle beschreibt eine Verbindungsmöglichkeit zwischen zwei CAx-System_Versionen, um Daten für einen bestimmten Zweck zu übertragen. Dabei kann es zum Teil notwendig sein, Daten in verschiedenen Dateiformaten zu übertragen,

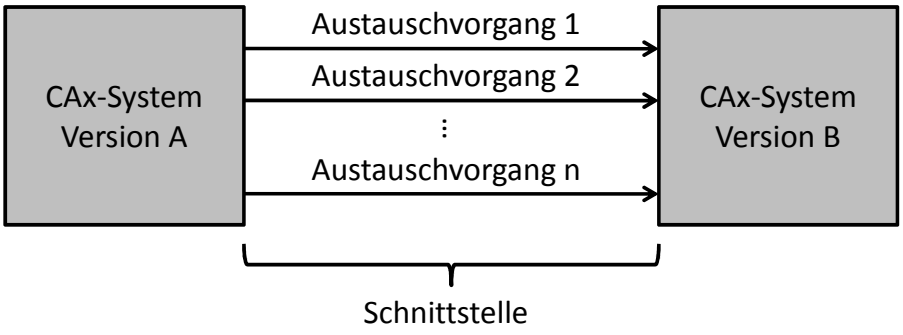


Abbildung 4.2: Zusammenhang zwischen einer Schnittstelle und Austauschvorgängen

da Dateiformate unterschiedliche Daten abbilden können (siehe Abschnitt 3.2.7). Der Datenaustausch in einem Dateiformat repräsentiert jeweils einen Austauschvorgang. Eine Schnittstelle kann eine oder mehrere Austauschvorgänge umfassen. Außerdem kann es mehrere Schnittstellen zwischen zwei CAx-System_Versionen geben.

Mit der Klasse **»Austauschvorgang«** lässt sich ein Austauschvorgang beschreiben. Es muss ein Dateiformat festgelegt werden, mit welchem sich die Dateien bei einem Austauschvorgang übertragen lassen. Dabei muss beachtet werden, dass die beiden beteiligten CAx-System_Versionen das Dateiformat unterstützten. Darauf wird im folgenden Abschnitt bei der Beschreibung des Anwendungsfalls »Schnittstelle definieren« genauer eingegangen. Mit dem Attribut »Dateianzahl« kann festgelegt werden, wie viele Dateien beim Datenaustausch übertragen werden. Wenn die Anzahl nicht feststeht und damit beliebig ist, soll kein Wert angegeben werden. Mit dem Attribut »Beschreibung« kann zum Beispiel erläutert werden, was beim Austausch zu beachten ist oder welche Einstellungen wichtig sind. Bei der Schnittstelle gibt es auch ein Attribut »Beschreibung«, dass beispielsweise zur Erläuterung von Vor- und Nachteilen gegenüber anderen Schnittstellen verwendet werden kann.

Damit wurde das für die Systemverwaltung entwickelte Datenmodell erläutert. Im folgenden Abschnitt werden die funktionalen Anforderungen, die in Abschnitt 4.1.1 spezifiziert wurden, durch Anwendungsfälle näher beschrieben.

4.1.3 Anwendungsfälle

In Abbildung 4.3 ist ein Anwendungsfalldiagramm dargestellt, welches die AWF und ihre Beziehungen zueinander beschreibt. Die AWF repräsentieren mögliche Interaktionen mit der zu entwickelnden Systemverwaltung. Zur Verbesserung der Übersicht wurden die AWF in zwei Gruppen unterteilt:

- Zum einen AWF, welche sich auf das Anlegen von Daten beziehen (Konfiguration)
- Zum anderen AWF, welche Funktionen auf Basis der Daten realisieren (Anwendung)

Zur weiteren Untersetzung wurde für jeden AWF ein Aktivitätsdiagramm erstellt. Im Folgenden werden aber nur zwei komplexe AWF inklusive der zugehörigen Aktivitätsdiagramme näher erläutert. Im Anhang B.1 befinden sich die Aktivitätsdiagramme von allen weiteren AWF. Außerdem sind in der Tabelle A.1 im Anhang A alle AWF der Systemverwaltung beschrieben.

AWF »Export- oder Importmöglichkeit einer CAx-System_Version definieren«

Beim AWF »Export- oder Importmöglichkeit einer CAx-System_Version definieren« soll zu einer bestehenden CAx-System_Version ein Dateiformat zugeordnet werden, dass entweder exportiert oder importiert werden kann. Der Ablauf des AWF ist in einem Aktivitätsdiagramm in Abbildung 4.4 dargestellt.

Als erstes muss die CAx-System_Version ausgewählt werden, zu der ein Dateiformat zugeordnet werden soll. Anschließend gibt es zwei Möglichkeiten. Entweder wurde das

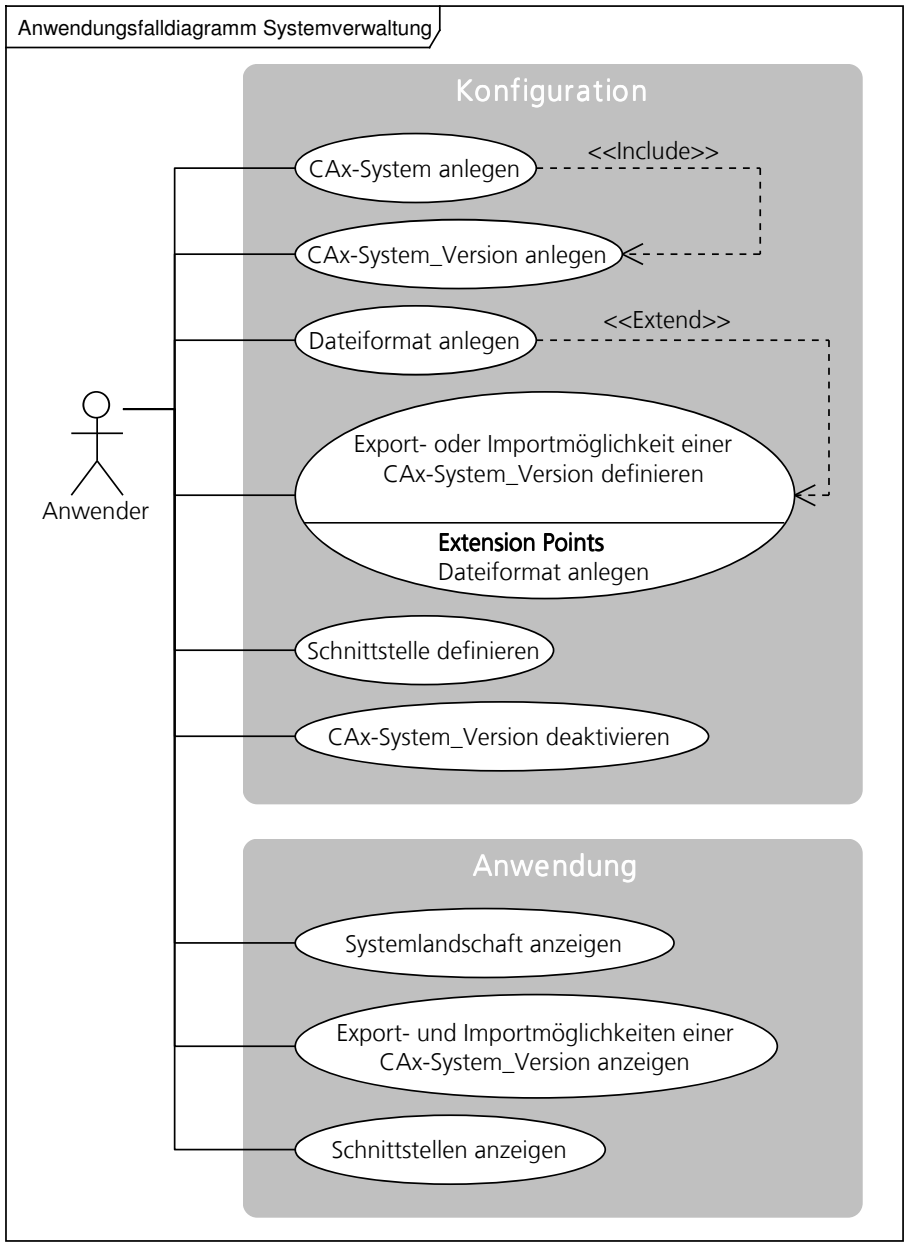


Abbildung 4.3: Anwendungsfalldiagramm »Systemverwaltung«

Dateiformat bereits angelegt, dann kann es aus den bestehenden Dateiformaten ausgewählt werden oder ein neues Dateiformat muss angelegt werden. Das Anlegen eines neuen Dateiformates wird durch den AWF »Dateiformat anlegen« beschrieben, der in den AWF »Export- oder Importmöglichkeit einer CAx-System_Version definieren« eingebunden ist (siehe Anwendungsfalldiagramm in Abbildung 4.3). Der Ablauf des AWF »Dateiformat anlegen« ist im Anhang in einem Aktivitätsdiagramm in Abbildung B.3 dargestellt.

Nachdem das Dateiformat festgelegt wurde, können noch die Einstellungsmöglichkeiten definiert werden, welche die CAx-System_Version in Abhängigkeit des Dateiformates bietet. Diese umfassen eine Menge von Einstellungen und gegebenenfalls Auswahlmöglichkeiten.

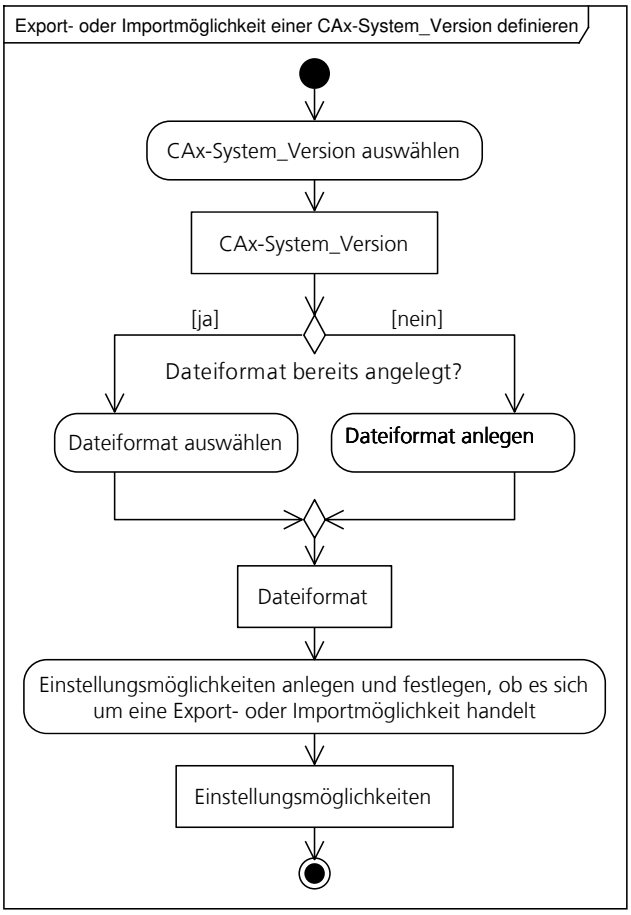


Abbildung 4.4: Aktivitätsdiagramm zum Anwendungsfall »Export- oder Importmöglichkeit einer CAx-System_Version definieren«

AWF »Schnittstelle definieren«

Der AWF »Schnittstelle definieren« dient zum Anlegen einer Schnittstelle zwischen zwei CAx-System_Versionen. Der Ablauf des AWF ist in einem Aktivitätsdiagramm in Abbildung 4.5 dargestellt.

Zur Definition einer Schnittstelle müssen zunächst zwei CAx-System_Versionen bestimmt werden, zwischen denen die Schnittstelle definiert werden soll. Zum einen der Sender und zum anderen der Empfänger. Auf Basis der ausgewählten CAx-System_Versionen können die Dateiformate bestimmt werden, welche zum Austausch verwendet werden können. Dies sind alle Dateiformate, die vom Sender exportiert und gleichzeitig vom Empfänger importiert werden können. Wenn kein Dateiformat existiert, ist das Anlegen der Schnittstelle nicht möglich. In diesem Fall müssen erst die entsprechenden Dateiformate zu den CAx-System_Versionen mit dem AWF »Export- oder Importmöglichkeit einer CAx-System_Version definieren« zugeordnet werden.

Wenn es mindestens ein Dateiformat gibt, können die Dateiformate ausgewählt werden, die über die Schnittstelle übertragen werden sollen. Zu jedem Dateiformat muss anschließend ein Austauschvorgang angelegt werden, wofür jeweils die Dateianzahl und eine Be-

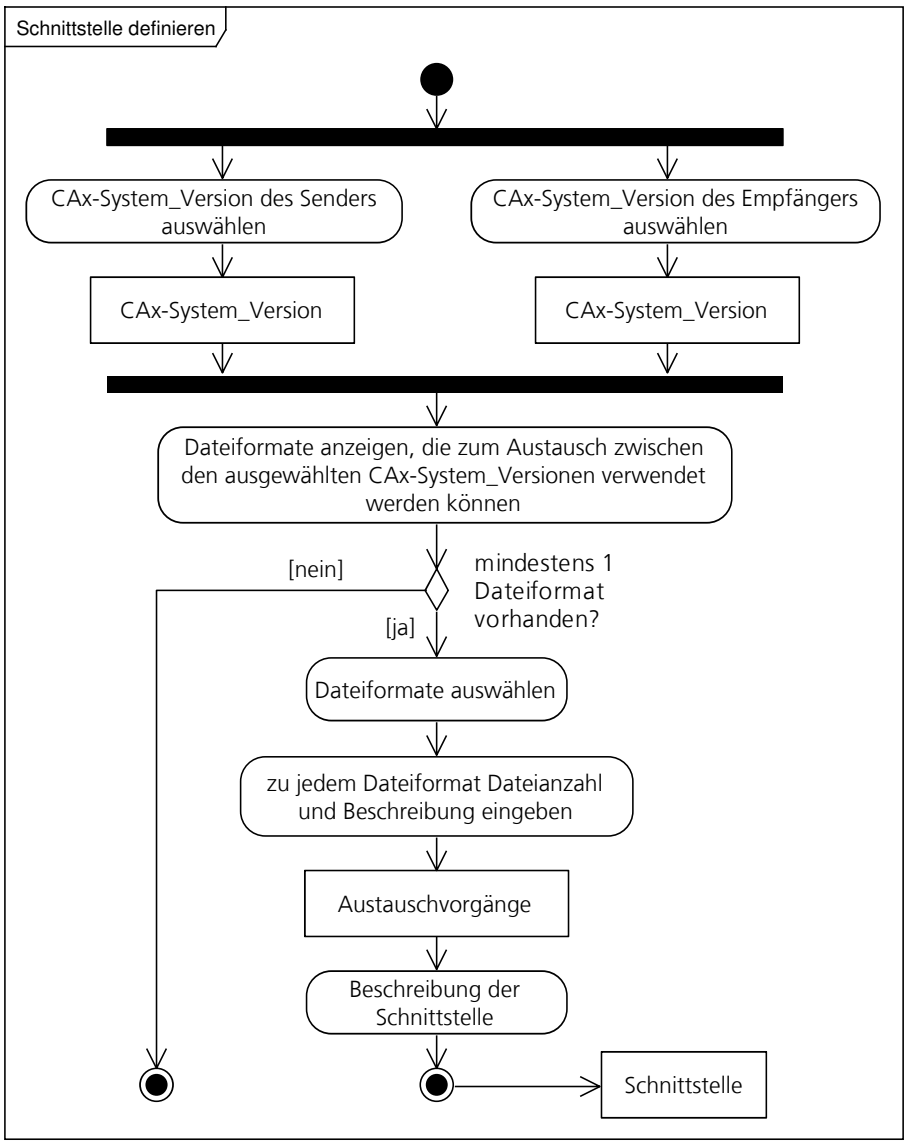


Abbildung 4.5: Aktivitätsdiagramm zum Anwendungsfall »Schnittstelle definieren«

schreibung eingegeben werden müssen. Wenn die Dateianzahl nicht bei jedem Datenaustausch gleich ist, wird kein Wert eingegeben. Zum Abschluss wird noch eine Beschreibung der Schnittstelle eingegeben.

4.1.4 Vorgehensweise zum Anlegen von Daten

Mit dem Datenmodell und den Anwendungsfällen wurde das Teilkonzept zur Realisierung der Systemverwaltung definiert. In diesem Abschnitt wird erläutert, in welcher Reihenfolge die Daten eines Unternehmens mit Hilfe der Systemverwaltung angelegt werden sollten. Dadurch soll das beschriebene Teilkonzept verständlicher werden. Zunächst wird ein allgemeiner Ablauf zum Anlegen der Daten beschrieben, der in Abbildung 4.6 dargestellt ist. Anschließend wird der Ablauf anhand eines Beispiels näher erläutert.

Allgemeiner Ablauf zum Anlegen von Daten in der Systemverwaltung

Zunächst sollten alle im Unternehmen verwendeten CAx-Systeme und ihre Versionen bestimmt und angelegt werden. Zur weiteren Beschreibung der CAx-Systeme beziehungsweise der Versionen der CAx-Systeme sollten alle Export- und Importmöglichkeiten definiert werden. In diesem Zusammenhang werden alle Dateiformate angelegt, die von den CAx-System_Versionen importiert und exportiert werden können. Außerdem werden die Einstellungsmöglichkeiten für den Export und Import von Dateiformaten im Bezug zu den CAx-System_Versionen festgelegt. Dadurch sind die Voraussetzungen für das Definieren und Beschreiben der Schnittstellen zwischen den CAx-System_Versionen geschaffen.

Beispiel für das Anlegen von Daten in der Systemverwaltung

Der allgemeine Ablauf wird im Folgenden anhand eines Beispiels erläutert. Für das Beispiel wird die Systemlandschaft des in Abschnitt 3.1 beschriebenen Prozesses verwendet (siehe Abbildung 3.3 und Tabelle 3.1). Die Beschreibung der vollständigen Systemlandschaft würde zu komplex werden, daher soll nur eine Schnittstelle zwischen zwei CAx-Systemen beschrieben werden. Als Beispiel wurde die komplexe Schnittstelle zwischen Pro/E und RobotMax gewählt, bei der mehrere VRML-Dateien und eine XML-Datei übertragen werden und damit zwei Austauschvorgänge stattfinden. Zur Untersetzung des Beispiels wurde ein Objektdiagramm erstellt, welches in Abbildung 4.7 dargestellt ist. Es beschreibt die beim Beispiel anfallenden Objekte der Klassen der Systemverwaltung und Beziehungen zwischen diesen.

Wie beim allgemeinen Ablauf beschrieben wurde, sollte ein Unternehmen als erstes alle vorhandenen CAx-Systeme und Versionen anlegen. Im Beispiel wurden diese beiden CAx-System_Versionen angelegt:

- Pro/ENGINEER Wildfire 5.0 vom Hersteller Parametric Technology Corporation
- RobotMax 2.0 vom Fraunhofer IFF entwickelt

Dafür wurde jeweils eine Instanz der Klasse »CAx-System« und »CAx-System_Version« mit den entsprechenden Eigenschaften angelegt (AWF »CAx-System anlegen«).

Als Nächstes sollte ein Unternehmen die Export- und Importmöglichkeiten aller CAx-System_Versionen definieren (AWF »Export- oder Importmöglichkeit einer CAx-System_Version definieren«). Im Beispiel wurden für die beiden CAx-System_Versionen nur die Möglichkeiten definiert, welche für die Schnittstelle zwischen beiden Versionen erforderlich sind. Pro/E kann VRML-Dateien und die für RobotMax benötigte XML-Datei exportieren. RobotMax kann beide Formate importieren.

Wenn als erstes die Export- und Importmöglichkeiten für Pro/E definiert werden, dann sind die Dateiformate noch nicht angelegt, können aber in diesem Zusammenhang mit dem AWF »Dateiformat anlegen« definiert werden (siehe Abbildung 4.4). Es entstehen zwei Instanzen der Klasse »Dateiformat«, welche das verwendete VRML- und XML-Format beschreiben. Bei VRML werden die genaue Bezeichnung (Virtual Reality Modeling Language), die Version (1.0), die Dateiendungen (wrl, wrml) und das Format, in dem die Daten innerhalb der Datei dargestellt werden (ASCII), definiert. Bei der XML-Datei für

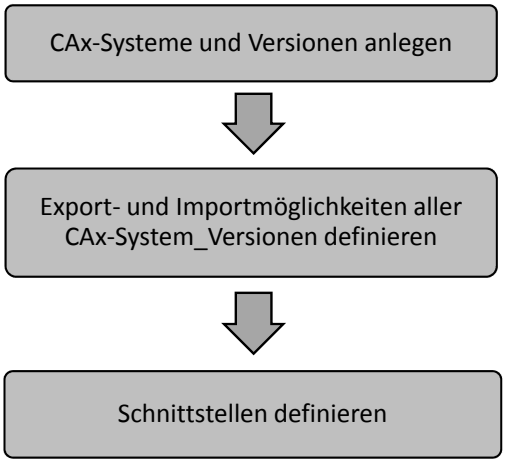


Abbildung 4.6: Vorgehensweise zum Anlegen der Daten in der »Systemverwaltung«

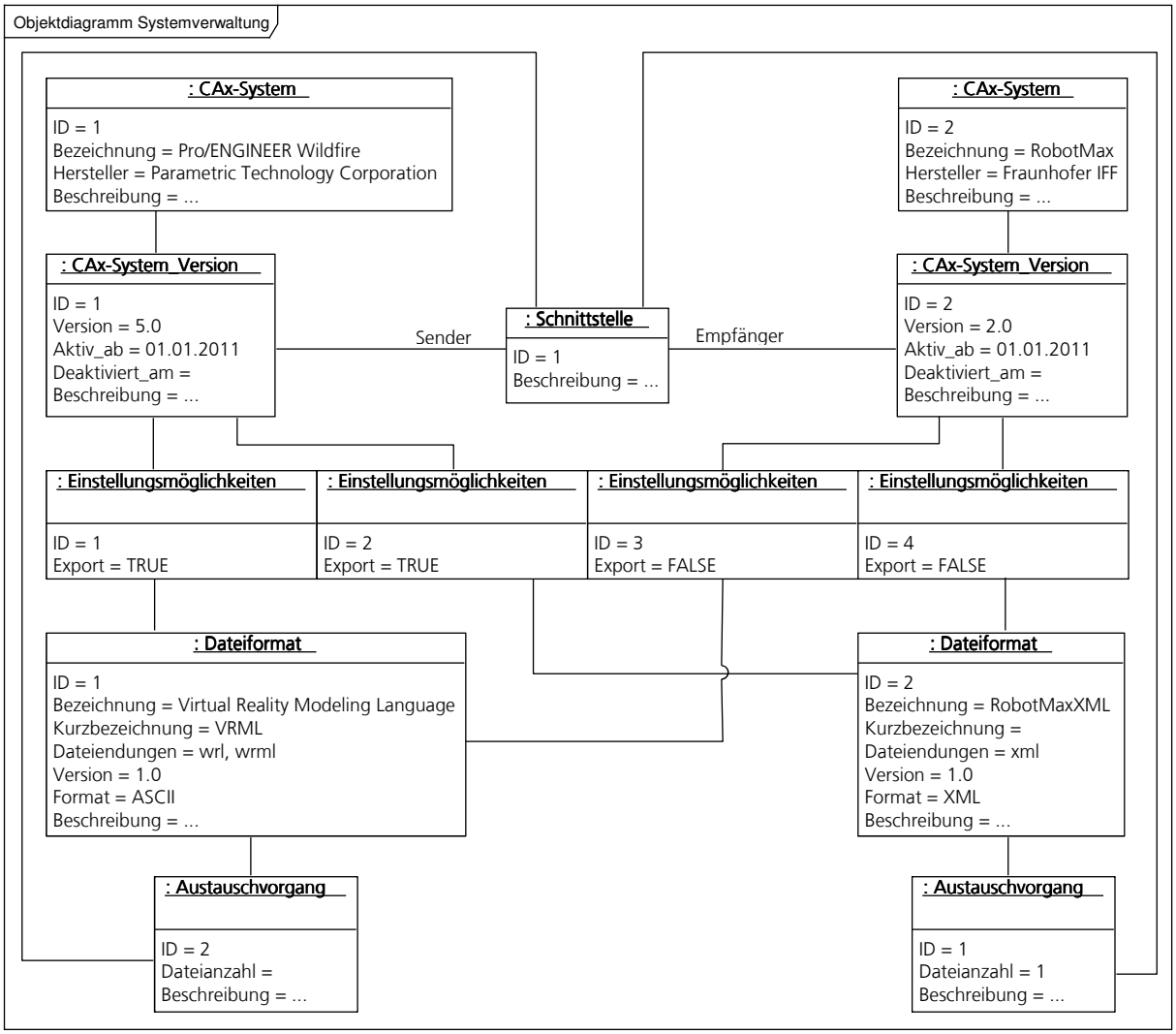


Abbildung 4.7: Objektdiagramm »Systemverwaltung«

RobotMax werden die Bezeichnung (RobotMaxXML), die Dateiendung (xml), die Version (1.0) und das Format (XML) definiert.

Beide Formate werden bei Pro/E als Exportmöglichkeit angelegt, da Daten von Pro/E zu RobotMax übertragen werden sollen. Die Instanzen der Dateiformate werden dabei nicht direkt zur Pro/E Version zugeordnet, sondern sind über ein weiteres Objekt »Einstellungsmöglichkeiten« verbunden. Dies beschreibt zum einen mit dem Attribut »Export«, ob es sich um eine Export- oder Importmöglichkeit handelt. Bei Pro/E wurde daher das Attribut bei beiden Einstellungsmöglichkeiten auf »TRUE« gesetzt. Zum anderen können Einstellungen beschrieben werden, die beim Export und Import durchgeführt werden können. Dies wurde bei dem Beispiel weggelassen, da es sonst zu umfangreich geworden wäre. Zu den Einstellungsmöglichkeiten könnten Objekte der Klasse »Einstellung« und zu diesen Objekte der Klasse »Auswahlmöglichkeit« zugeordnet werden.

Bei der Definition der Export- und Importmöglichkeiten für RobotMax sind die Dateiformate bereits angelegt und können daher ausgewählt werden (siehe Abbildung 4.4). Beide Dateiformate lassen sich zu RobotMax über zwei weitere Instanzen der Klasse »Einstellungsmöglichkeiten« zuordnen. Dabei wird das Attribut »Export« auf »FALSE« gesetzt, weil RobotMax beide Dateiformate importieren kann. Damit wurden für Pro/E und RobotMax die Export- und Importmöglichkeiten definiert, welche zum Anlegen der Schnittstelle zwischen beiden CAx-System_Versionen benötigt werden.

Im nächsten Schritt sollte ein Unternehmen alle Schnittstellen definieren (AWF »Schnittstelle definieren«). Im Beispiel wird das Anlegen einer Schnittstelle zwischen Pro/E und RobotMax beschrieben. Zunächst müssen die CAx-System_Versionen des Senders und Empfängers ausgewählt werden (siehe Abbildung 4.5). Der Sender ist Pro/E, da die Daten von dort exportiert werden. Nachdem beide Versionen ausgewählt wurden, werden die Dateiformate angezeigt, die von beiden unterstützt werden, das heißt, von Pro/E exportiert und von RobotMax importiert werden können. Dies trifft im Beispiel auf die beiden angelegten Dateiformate zu. Diese werden ausgewählt und für jedes Dateiformat wird ein eigener Austauschvorgang definiert. Dadurch entstehen zwei Objekte der Klasse »Austauschvorgang«. Bei dem Austauschvorgang der VRML-Dateien kann keine Dateianzahl festgelegt werden, weil diese von der Anzahl der Einzelteile des 3D-Modells in Pro/E abhängt. Bei der XML-Datei wird aber immer nur eine Datei ausgetauscht. Daher kann dort als Dateianzahl »1« festgelegt werden.

Die Vorgehensweise des beschriebenen Beispiels würde für alle CAx-Systeme, Versionen und Schnittstellen durchgeführt werden müssen, um die Systemlandschaft eines Unternehmens vollständig zu beschreiben. Im Anschluss kann ein Anwender die AWF, die im Anwendungsfalldiagramm in Abbildung 4.3 in der Gruppe »Anwendung« dargestellt sind, verwenden, um Informationen über die Systemlandschaft nachschlagen zu können.

Damit wurde das Teilkonzept »Systemverwaltung« erläutert und es folgt die Beschreibung des zweiten Teilkonzepts »Prozessverwaltung«.

4.2 Prozessverwaltung

Bei den Ergebnissen der Analyse in Kapitel 3 erwies sich als ein Problem der Überblick über die Systemlandschaft. Durch die im vorangehenden Abschnitt beschriebene Systemverwaltung kann der Überblick über die Systemlandschaft unterstützt werden. Zwei weitere Probleme, die ermittelt wurden, beziehen sich auf die Durchführung von Datenaustauschprozessen:

- Alternativen und Kopplungen beim Datenaustausch
- Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen

Mit dem in diesem Abschnitt beschriebenen Konzept für eine Prozessverwaltung sollen die Anwender eines Unternehmens bei der Durchführung von Datenaustauschprozessen und anschließend bei der Nachvollziehbarkeit der durchgeführten Datenaustauschprozesse unterstützt werden. Dafür werden im folgenden Abschnitt zunächst die Anforderungen an die Prozessverwaltung beschrieben.

4.2.1 Anforderungen

Mit der Prozessverwaltung sollen Prozessketten definiert werden können, welche Abläufe beziehungsweise Teilprozesse in der Produktentwicklung beschreiben, an denen eine Menge von CAx-Systemen beteiligt sind und zwischen denen Daten ausgetauscht werden müssen. Ein Beispiel dafür ist der Prozess vom Fraunhofer IFF, der in Abschnitt 3.1 analysiert wurde. Eine Prozesskette soll aus einer beliebigen Anzahl an Prozessschritten bestehen, die jeweils mit Hilfe eines CAx-Systems durchgeführt werden. Bei der Verknüpfung von Prozessschritten findet ein Datenaustausch zwischen den beteiligten CAx-Systemen statt. Bei der Beschreibung von Datenaustauschprozessen zwischen zwei Prozessschritten sollen Alternativen und Kopplungen berücksichtigt werden (siehe Abschnitt 3.1.5). Alternativen sind unterschiedliche Möglichkeiten, die es geben kann, um Daten für ein CAx-System bereitzustellen beziehungsweise unterschiedliche Möglichkeiten für den Import in einem CAx-System. Kopplungen entstehen, wenn ein CAx-System von verschiedenen anderen CAx-Systemen und/oder in verschiedenen Dateiformaten Daten benötigt.

Das Ziel der Prozessverwaltung ist daher keine allgemeine Verwaltung von Geschäftsprozessen eines Unternehmens, denn dafür gibt es bereits Workflowmanagement-Systeme beziehungsweise Workflowmanagement-Komponenten in PDM-Systemen (siehe Abschnitt 2.4.2). Das Ziel der Prozessverwaltung ist die Unterstützung von Prozessen, die beim dateibasierten Datenaustausch zwischen CAx-Systemen durchgeführt werden.

Durch die Definition von Prozessketten wird in der Prozessverwaltung die Vorgehensweise für bestimmte Abläufe festgelegt. Für die Anwender wird dadurch beschrieben, wie sie vorgehen sollen. Bei der Definition der Abläufe sollen Alternativen möglich sein. Die Anwender können dann jeweils die Alternative auswählen, welche sich für einen konkreten Ablauf am besten eignet. Für die Entscheidungsfindung sollen den Anwendern Informationen über den Prozess sowie über die Austauschmöglichkeiten und Schnittstellen zwischen CAx-Systemen zur Verfügung gestellt werden.

Neben der Definition von Prozessketten soll die Prozessverwaltung auch die Nachvollziehbarkeit von durchgeführten Prozessketten und den dabei durchgeführten Datenaustauschprozessen ermöglichen. Ein Anwender soll eine durchgeführte Prozesskette beschreiben können. Dafür ist es wichtig, dass der Anwender eingeben kann, für welche Alternativen er sich entschieden hat, um den konkreten Ablauf festzulegen, der durchgeführt wurde. Anschließend soll der Anwender den durchgeführten Prozess beschreiben können, wobei alle durchgeführten Datenaustauschprozesse einzeln beschrieben werden sollen. Welche Daten zur Nachvollziehbarkeit eines Datenaustausch nötig sind, wurde in Abschnitt 3.3 analysiert. Zum Beispiel soll der Anwender Einstellungen, die beim Export und Import in den CAx-Systemen festgelegt wurden, speichern können. Außerdem soll eine Verbindung zwischen den Daten des durchgeführten Prozesses und den erzeugten Dateien erstellt werden können, damit nachvollzogen werden kann, wie die Dateien entstanden sind.

Aus den textuell beschriebenen Anforderungen an die Prozessverwaltung lassen sich Anforderungen an ein Datenmodell ableiten, welches die für die Prozessverwaltung notwendigen Daten beschreibt. Folgende Anforderungen werden an das Datenmodell gestellt:

- Beschreibung von Prozessketten und -schritten
- Beschreibung von Alternativen und Kopplungen zum Datenaustausch
- Beschreibung von durchgeführten Prozessketten
- Beschreibung von durchgeführten Datenaustauschprozessen

Auf Grundlage dieser Anforderungen wurde ein Datenmodell für die Prozessverwaltung erstellt, welches in Abschnitt 4.2.2 beschrieben wird.

Weiterhin lassen sich aus den textuell beschriebenen Anforderungen Funktionen ableiten, welche die Prozessverwaltung den Anwendern zur Verfügung stellen muss. Diese sollen den Anwendern einerseits ermöglichen, Daten in der Prozessverwaltung anzulegen. Andererseits können verschiedene Funktionen auf Basis der angelegten Daten realisiert werden, welche den Anwendern bereitgestellt werden sollen. Folgende funktionale Anforderungen werden an die Prozessverwaltung gestellt:

- Daten für die Prozessverwaltung anlegen können (Prozessketten, -schritte, ...)
- Durchgeführte Prozesskette beschreiben
- Angelegte Prozesskette anzeigen
- Durchgeführte Prozesskette anzeigen

Die funktionalen Anforderungen werden in Abschnitt 4.1.3 durch Anwendungsfälle näher beschrieben.

4.2.2 Datenmodell

Als Grundlage für die Prozessverwaltung wurde ein Datenmodell entwickelt, das in Abbildung 4.8 in Form eines Klassendiagramms dargestellt ist. Um das Klassendiagramm übersichtlich zu gestalten, wurden die Klassen unterschiedlich eingefärbt. In der rechten oberen Ecke befindet sich eine Legende, welche die verwendeten Farben beschreibt. Es wurden folgende drei Gruppen gebildet:

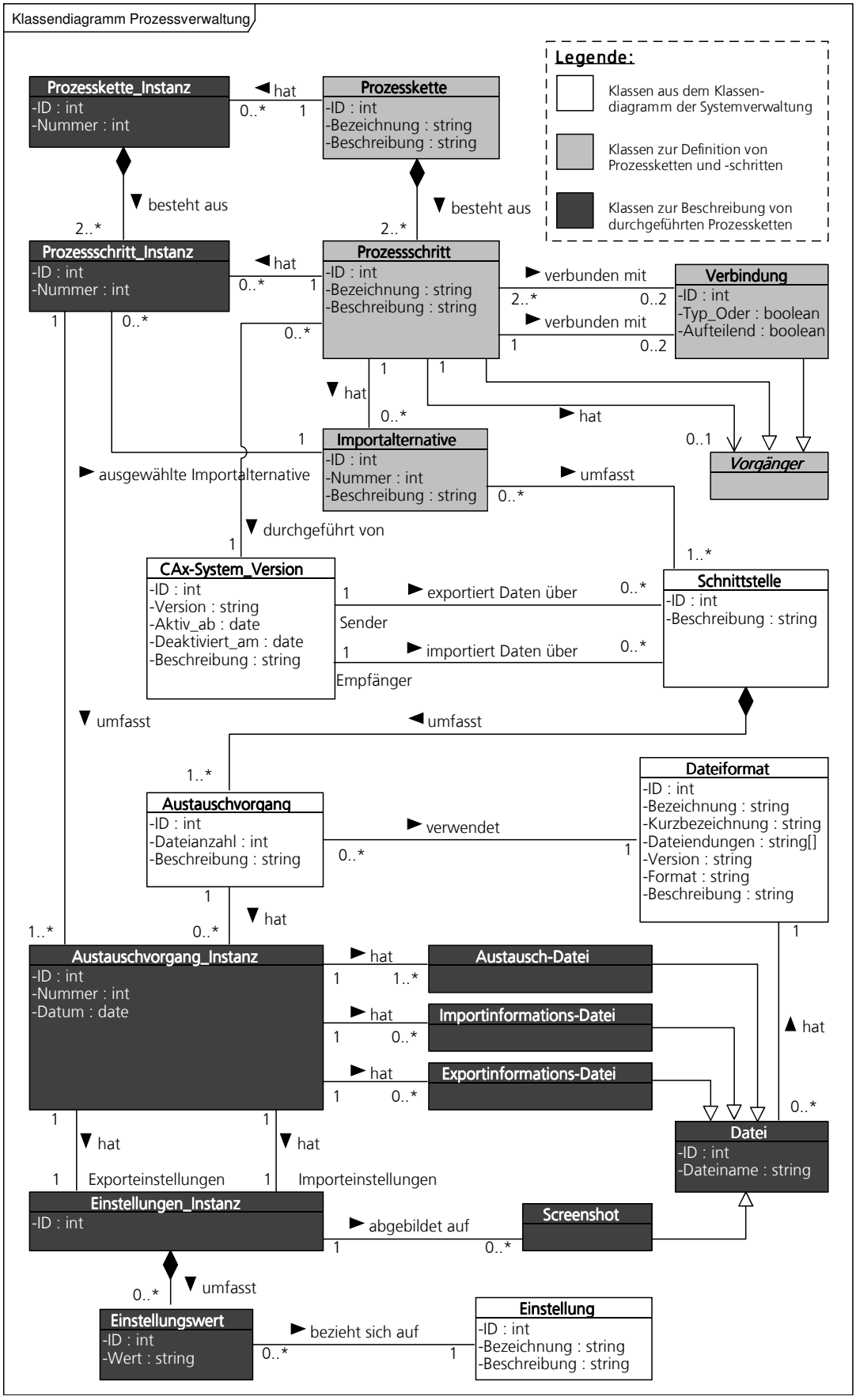


Abbildung 4.8: Klassendiagramm »Prozessverwaltung«

- **Weiße Klassen:** stammen aus dem Datenmodell der Systemverwaltung (siehe Abbildung 4.1.2)
- **Hellgraue Klassen:** werden zur Definition von Prozessketten und -schritten benötigt
- **Dunkelgraue Klassen:** werden zur Beschreibung von durchgeführten Prozessketten benötigt

Die weißen Klassen wurden bereits im Teilkonzept Systemverwaltung definiert und beschrieben. In der Systemverwaltung werden alle CAx-Systeme und Versionen sowie Schnittstellen zwischen diesen unabhängig von den Prozessen beschrieben. Die Prozessverwaltung baut auf diesen Klassen auf und setzt die Schnittstellen und CAx-Systeme in Bezug zu Prozessschritten. Dabei werden in Abhängigkeit eines Prozessschrittes nur die CAx-Systeme und Schnittstellen ausgewählt, die sich für diesen Prozessschritt am besten eignen. Dies kann zusätzlich für die Anwender beschrieben werden.

Im Folgenden werden die neuen Klassen der Prozessverwaltung und ihre Beziehungen untereinander sowie zu den Klassen der Systemverwaltung erläutert. Als erstes sollen die Klassen zur Definition von Prozessketten und -schritten (hellgraue Klassen) beschrieben werden, weil die Klassen zur Beschreibung von durchgeführten Prozessketten (dunkelgraue Klassen) auf diesen aufbauen.

Klassen zur Definition von Prozessketten und -schritten

Die Klasse **»Prozesskette«** dient zur Beschreibung von Abläufen, die aus einer Menge von Prozessschritten bestehen. Mit dem Attribut »Bezeichnung« kann ein Name für die Prozesskette festgelegt werden. Außerdem kann durch eine Beschreibung zum Beispiel der Verwendungszweck der Prozesskette erläutert werden.

Die einzelnen Schritte einer Prozesskette können durch die Klasse **»Prozessschritt«** beschrieben werden. Eine Bezeichnung und Beschreibung können mit den gleichnamigen Attributen festgelegt werden. Außerdem wird die CAx-System_Version zugeordnet, welche zur Durchführung des Prozessschritts verwendet werden soll.

Weiterhin soll eine Prozesskette alternative und gekoppelte Abläufe enthalten können. Um dies zu ermöglichen, muss der Ablauf an einer gewünschten Stelle in mehrere Abläufe aufgeteilt und an einer weiteren Stelle wieder zusammengeführt werden können. Dafür sollen die aus der UML bekannten exklusiven »Oder-Verbindungen« für die Beschreibung von alternativen Abläufen sowie »Und-Verbindungen« zur Beschreibung von gekoppelten Abläufen verwendet werden. Mit der Klasse **»Verbindung«** kann eine »Oder-« oder eine »Und-Verbindung« definiert werden. Das Attribut »Typ_Oder« legt fest, welcher Verbindungstyp vorliegt. Außerdem kann mit dem Attribut »Aufteilend« festgelegt werden, ob die Verbindung den Ablauf der Prozesskette aufteilt oder wieder zusammenführt. Im Klassendiagramm in Abbildung 4.8 sind zwei Assoziationen von der Klasse Verbindung zum Prozessschritt dargestellt. Diese beschreiben die zwei unterschiedlichen Seiten (Input und Output) einer Verbindung. Auf einer der beiden Seiten befindet sich immer nur ein Prozessschritt und auf der anderen mindestens zwei. Durch das Attribut »Aufteilend« ist festgelegt, auf welcher Seite sich der einzelne Prozess befindet. Wenn der Wert

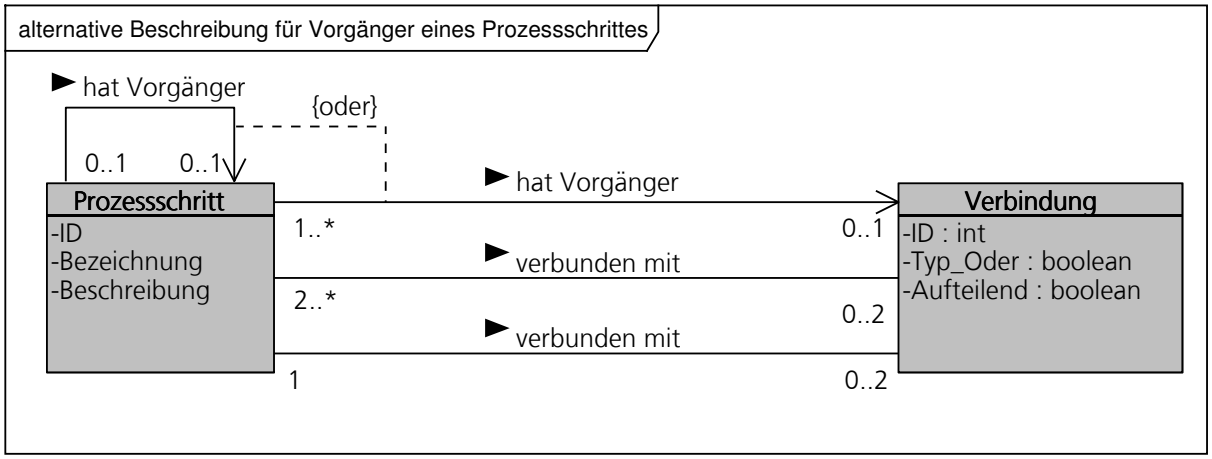


Abbildung 4.9: Alternative Beschreibung für Vorgänger eines Prozessschrittes

»TRUE« ist, dann befindet sich der einzelne Prozess auf der Seite des Inputs und beim Wert »FALSE« auf der Seite des Outputs.

Zur Beschreibung des Ablaufs einer Prozesskette sollen die Prozessschritte und Verbindungen miteinander verknüpft werden. Die Verknüpfung soll über die Zuordnung eines Vorgängers bei jedem Prozessschritt erfolgen. Die Ausnahme ist der erste Prozessschritt einer Prozesskette, da für diesen kein Vorgänger existiert. Ein Vorgänger wird durch die Klasse **»Vorgänger«** repräsentiert. Diese abstrakte Klasse beschreibt die zwei Möglichkeiten für einen Vorgänger eines Prozessschritts:

- Zum einen kann ein einzelner Prozessschritt ein Vorgänger sein.
- Zum anderen kann eine Verbindung ein Vorgänger sein.

Die abstrakte Klasse Vorgänger dient zur besseren Strukturierung, aber von ihr können keine Objekte erzeugt werden. Eine alternative Möglichkeit zur Modellierung der Vorgänger von Prozessschritten ist in Abbildung 4.9 dargestellt.

Weiterhin müssen zu einem Prozessschritt eine oder mehrere **»Importalternativen«** festgelegt werden. Diese beschreiben jeweils eine mögliche Vorgehensweise für den Import von Daten, die für die Durchführung des Prozessschrittes benötigt werden. Es lassen sich damit Alternativen für den Datenaustausch mit vorherigen Prozessschritten beschreiben. Dadurch wird die Anforderung erfüllt, dass Alternativen beim Datenaustausch beschrieben werden können. Weiterhin umfassen Importalternativen jeweils eine oder mehrere Schnittstellen, die zum Import verwendet werden sollen. Durch die Zuordnung von mehreren Schnittstellen können Kopplungen beschrieben werden, das heißt, dass ein Prozessschritt bei einer Importalternative Daten von mehreren CAx-System_Versionen benötigt.

Zum besseren Verständnis der erläuterten Klassen zur Definition von Prozessketten und -schritten ist in Abbildung 4.10 ein Beispiel für eine Prozesskette dargestellt. Die Prozesskette besteht aus vier Prozessschritten, von denen »Prozessschritt 2« und »3« durch eine »Oder-Verbindung« mit »Prozessschritt 4« verknüpft sind. Der Prozessablauf ist mit Hilfe der »durchgezogenen Pfeile« beschrieben. Die »gestrichelten Pfeile« stellen Importalternativen dar. Zum Import von Daten in »Prozessschritt 2« gibt es zwei verschiedene

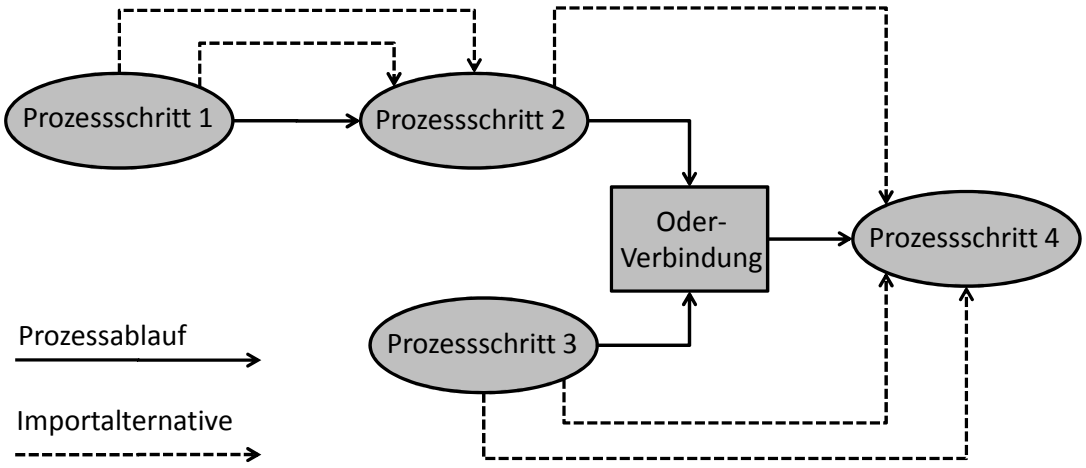


Abbildung 4.10: Beispiel für eine Prozesskette

Möglichkeiten, um Daten von »Prozessschritt 1« zu importieren. Beim »Prozessschritt 4« gibt es drei verschiedene Importalternativen. Es können entweder Daten vom »Prozessschritt 2« oder durch zwei unterschiedliche Möglichkeiten von »Prozessschritt 3« importiert werden.

Mit den bisher beschriebenen Klassen lassen sich Prozessketten und -schritte sowie Alternativen und Kopplungen zum Datenaustausch definieren. Zwei weitere Anforderungen sind die Beschreibung von durchgeführten Prozessketten und der dabei durchgeführten Datenaustauschprozesse. Die dafür benötigten Klassen sind im Klassendiagramm dunkelgrau dargestellt und werden im Folgenden erläutert.

Klassen zur Beschreibung von durchgeführten Prozessketten

Von einer Prozesskette kann mit der Klasse **»Prozesskette_Instanz«** eine beliebige Anzahl von Instanzen erzeugt werden. Eine Instanz entspricht einer durchgeführten Prozesskette. Sie besteht aus einer Menge von **»Prozessschritt_Instanzen«**, welche jeweils einen durchgeführten Prozessschritt beschreiben. Bei beiden Instanzen kann durch das Attribut »Nummer« die Zahl der Instanz im Bezug auf die Prozesskette beziehungsweise den Prozessschritt festgelegt werden. Zu einer Prozessketten_Instanz dürfen nur Prozessschritt_Instanzen zugeordnet werden, deren zugehöriger Prozessschritt zu der Prozesskette der Prozessketten_Instanz gehört. Bei einer durchgeführten Prozesskette müssen aber nicht alle Prozessschritte ausgeführt werden, da es zum Teil alternative Abläufe in einer Prozesskette geben kann. Daher müssen nicht zu allen Prozessschritten einer Prozesskette Instanzen angelegt werden.

Da bei einem durchgeführten Prozessschritt nur eine der definierten Importalternativen verwendet werden konnte, wird der Instanz eines Prozessschrittes die ausgewählte Importalternative zugeordnet (siehe Abbildung 4.11). Zum einen kann dadurch nachvollzogen werden, welche Schnittstellen für den Import verwendet wurden. Zum anderen wirkt sich die ausgewählte Importalternative auf die zugehörigen **»Austauschvorgang_Instanzen«** aus. Der Grund dafür ist, dass durch die Importalternative feststeht,

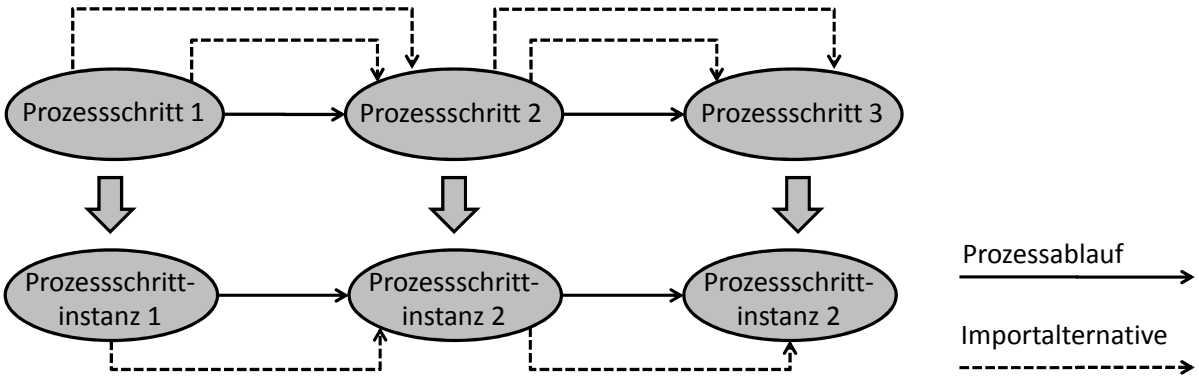


Abbildung 4.11: Beispiel für eine Prozessketteninstanz

welche Schnittstellen zum Import einer Prozessschritt_Instanz verwendet wurden. Die Schnittstellen umfassen wiederum einen oder mehrere Austauschvorgänge. Zu diesen Austauschvorgängen sollen Instanzen beschrieben werden können, um die genauen Datenaustauschprozesse zwischen CAx-System_Versionen nachvollziehen zu können.

Die Instanz eines Austauschvorgangs hat ein Attribut »Nummer«, mit dem die Zahl der Instanz in Bezug zum Austauschvorgang festgelegt werden kann. Außerdem kann durch das Attribut »Datum« festgelegt werden, wann der konkrete Austauschvorgang durchgeführt wurde. Welche Daten bei einem Austauschvorgang anfallen, wurde in Abschnitt 3.3 analysiert und zusammenfassend in Abbildung 3.10 dargestellt. Bei jedem durchgeführten Austauschvorgang wird mindestens eine Datei von der CAx-System_Version, welche den Sender darstellt, erzeugt beziehungsweise exportiert. Diese Dateien, die zwischen den zwei CAx-System_Versionen der zugehörigen Schnittstelle ausgetauscht werden sollen, werden durch die Klasse **»Austausch-Datei«** repräsentiert. Sie ist eine Spezialisierung der Klasse **»Datei«**, welche eine beliebige Datei repräsentiert. Der Dateiname kann durch das gleichnamige Attribut festgelegt werden. Weiterhin können bei einem Austauschvorgang zusätzliche Dateien beim Import und Export entstehen, wie zum Beispiel Log- oder Konfigurationsdateien. Diese Dateien werden durch die Klassen **»Importinformations-Datei«** und **»Exportinformations-Datei«** repräsentiert.

Durch die Zuordnung von erzeugten Dateien eines durchgeführten Austauschvorgangs beziehungsweise der gesamten Prozesskette werden diese mit vielen Informationen in Beziehung gesetzt. Dadurch kann nachvollzogen werden, wie die Dateien entstanden sind. Alle Daten, die in der Prozess- und Systemverwaltung mit den Dateien in Verbindung stehen, dienen zur näheren Beschreibung der Dateien und können daher auch als Metadaten bezeichnet werden.

Des Weiteren können zu einer Instanz eines Austauschvorgangs die durchgeführten Einstellungen in den beiden zugehörigen CAx-System_Versionen für den Export beziehungsweise Import abgelegt werden. Die Export- und Importeinstellungen eines durchgeführten Austauschvorgangs werden durch die Klasse **»Einstellungen_Instanz«** repräsentiert. Sie umfasst verschiedene **»Einstellungswerte«**, welche die eingestellten Werte für eine vorhandene Einstellungsmöglichkeit darstellen. Der Anwender muss in Abhängigkeit der zugehörigen CAx-System_Versionen Werte für die in der Systemverwaltung festgeleg-

ten Einstellungsmöglichkeiten eingeben. Die Ermittlung der möglichen Einstellungen, die bei einer Instanz eines Austauschvorgangs für den Import und Export getroffen werden können, kann wie folgt durchgeführt werden. Der Austauschvorgang_Instanz ist ein Austauschvorgang zugeordnet, welcher eine Beziehung zu dem verwendeten Dateiformat hat. Außerdem können vom Austauschvorgang über die zugeordnete Schnittstelle die zugehörigen CAx-System_Versionen bestimmt werden. Zu beiden CAx-System_Versionen können so die Einstellungsmöglichkeiten ermittelt werden. Für alle möglichen Einstellungen der beiden CAx-Systeme sollen die festgelegten Einstellungswerte eines durchgeführten Austauschvorgangs angegeben werden können. Darauf wird im folgenden Abschnitt bei der Erläuterung des Anwendungsfalls »durchgeführten Austauschvorgang beschreiben« im Detail eingegangen.

Das Problem bei den Einstellungswerten ist, dass sie nicht immer automatisch erfasst werden können. Die Voraussetzung für eine Automatisierung ist, dass ein CAx-System eine Möglichkeit bietet, die durchgeführten Einstellungen in eine Datei zu exportieren. Bei der Analyse des CAD-Systems Pro/E in Abschnitt 3.3 wurde festgestellt, dass einige Einstellungen zum Teil in den erstellten Export- und Importinformations-Dateien abgelegt werden. Diese könnten durch einen Parser automatisiert ausgelesen werden. Damit wäre aber nur ein Teil der Einstellungswerte nachvollziehbar abgelegt. Die Möglichkeit mit der automatisierten Erfassung bietet sich daher nur an, wenn alle relevanten Eigenschaften aus einer Datei, die vom CAx-System erstellt wird, ausgelesen werden können. Der Nachteil bei dieser Variante ist außerdem, dass der Parser normalerweise nicht von den Anwendern des CAD-Systems geschrieben werden kann und bei Änderung der Exportbeziehungsweise Importinformations-Dateien angepasst werden muss. Eine automatisierte Lösung ist daher mit einem hohen Implementierungs- und Pflegeaufwand verbunden. Wenn eine automatisierte Lösung nicht möglich oder zu aufwändig ist, muss der Anwender die Einstellungswerte manuell erfassen. Dafür gibt es zwei unterschiedliche Möglichkeiten:

- Der Anwender kann die Einstellungen auf einem Blatt Papier oder in einer Text-Datei notieren und dann in der Prozessverwaltung beim Anlegen der Austauschvorgang_Instanz eingeben. Dies ist aber für den Anwender im Gegensatz zu einer automatisierten Lösung aufwändig und fehleranfällig.

- Daher ist eine weitere Idee zur Lösung des Problems, dass der Anwender einen oder mehrere **»Screenshots«** mit den festgelegten Einstellungen bei der Durchführung eines Datenaustauschs erstellt. Auf Basis der Screenshots können die Werte für die Einstellungen eingegeben werden. Eine andere Möglichkeit ist, die Werte nicht abzulegen, sondern nur die Screenshots zu speichern. Die Nachvollziehbarkeit der Einstellungen ist trotzdem gewährleistet, da über die Screenshots die Einstellungswerte jederzeit nachgeschlagen werden können. Beim Anlegen einer Austauschvorgang_Instanz müssten die erstellten Screenshots dafür zu den Einstellungs_Instanzen zugeordnet werden.

Damit wurde das für die Prozessverwaltung entwickelte Datenmodell erläutert. Im folgenden Abschnitt werden die funktionalen Anforderungen, die in Abschnitt 4.2.1 spezifiziert wurden, durch Anwendungsfälle näher beschrieben.

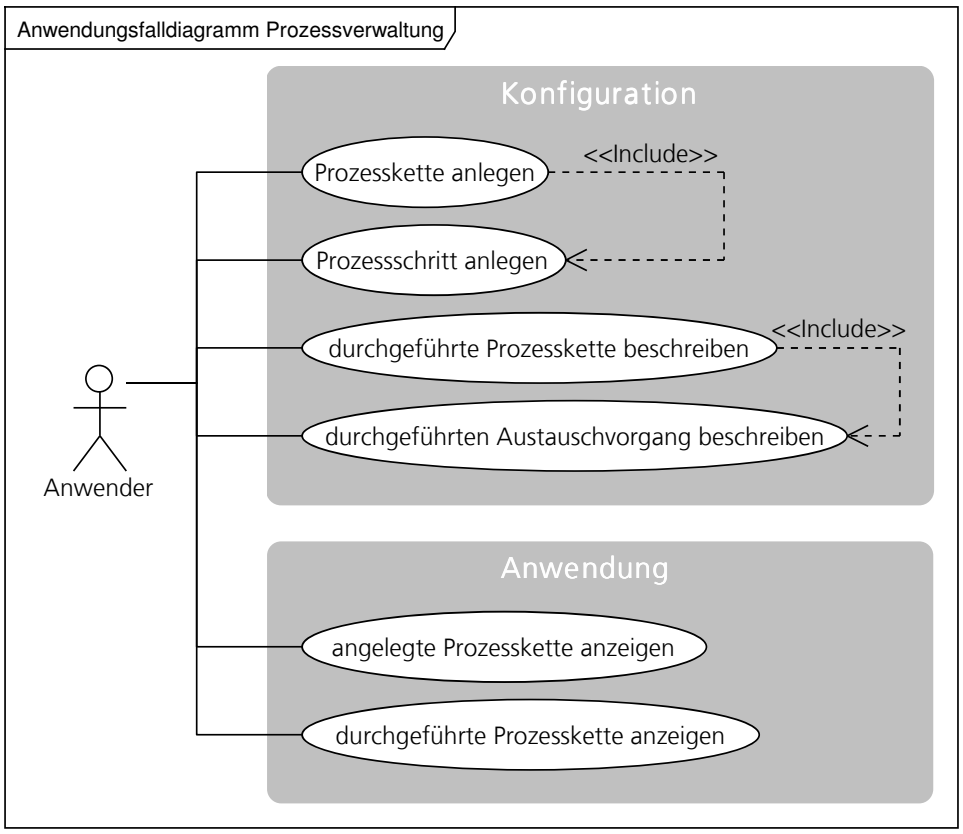


Abbildung 4.12: Anwendungsfalldiagramm »Prozessverwaltung«

4.2.3 Anwendungsfälle

In Abbildung 4.12 ist ein Anwendungsfalldiagramm dargestellt, welches die AWF und ihre Beziehungen zueinander beschreibt. Die AWF repräsentieren mögliche Interaktionen mit der zu entwickelnden Prozessverwaltung. Zur Verbesserung der Übersicht wurden die AWF wie bei der Systemverwaltung in zwei Gruppen unterteilt:

- Zum einen AWF, welche sich auf das Anlegen von Daten beziehen (Konfiguration)
- Zum anderen AWF, welche Funktionen auf Basis der Daten realisieren (Anwendung)

Die AWF aus der Gruppe »Konfiguration« repräsentieren die funktionale Anforderung »Daten für die Prozessverwaltung anlegen können« sowie »Durchgeführte Prozesskette beschreiben« aus Abschnitt 4.1.1.

Zur weiteren Untersetzung wurde für jeden AWF ein Aktivitätsdiagramm erstellt. Im Folgenden sollen nur vier AWF näher erläutert werden. Im Anhang B.2 befinden sich die Aktivitätsdiagramme von allen weiteren AWF. Außerdem befindet sich in der Tabelle A.2 im Anhang A eine Beschreibung zu allen AWF der Prozessverwaltung.

AWF »Prozesskette anlegen«

Beim AWF »Prozesskette anlegen« soll eine Prozesskette definiert werden. Der Ablauf des AWF ist in einem Aktivitätsdiagramm in Abbildung 4.13 dargestellt. Zunächst muss ei-

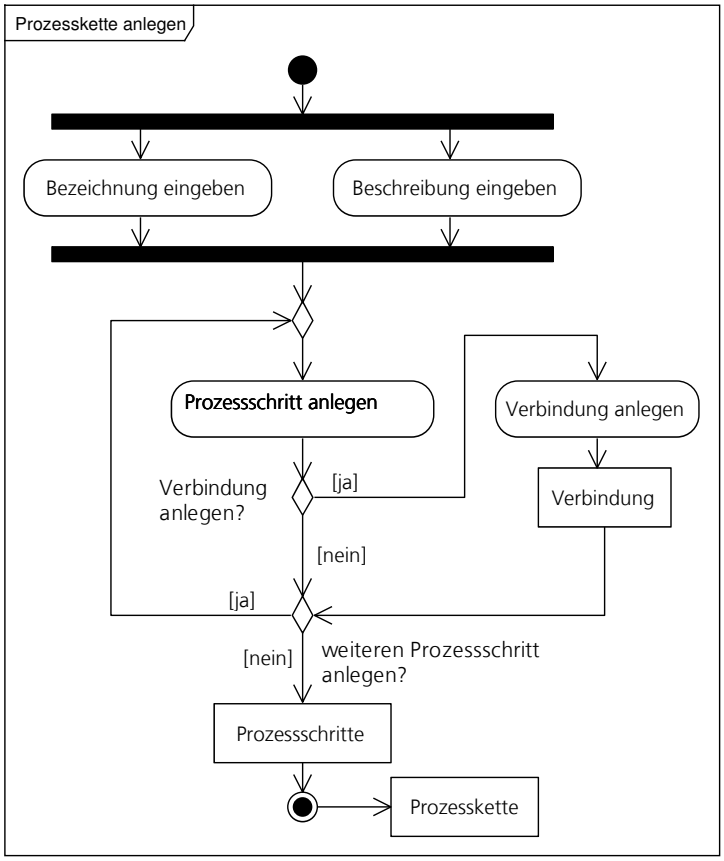


Abbildung 4.13: Aktivitätsdiagramm zum Anwendungsfall »Prozesskette anlegen«

ne Bezeichnung der Prozesskette sowie eine Beschreibung eingegeben werden, welche den Zweck der Prozesskette ausführlich erläutert. Anschließend werden mehrere Prozessschritte mit dem AWF »Prozessschritt anlegen« erzeugt, welcher in den AWF »Prozesskette anlegen« eingebunden ist (siehe Anwendungsfalldiagramm in Abbildung 4.12). Im Anschluss daran können beliebig viele Verbindungen angelegt werden.

AWF »Prozessschritt anlegen«

Der AWF »Prozessschritt anlegen« ist in einem Aktivitätsdiagramm in Abbildung 4.14 dargestellt. Als erstes muss eine CAx-System_Version ausgewählt werden, welche zur Durchführung des Prozessschritts verwendet werden soll. Wenn der Prozessschritt der erste Prozessschritt einer Prozesskette ist, kann kein Vorgänger definiert werden, so dass die Zuordnung übersprungen werden kann. Ansonsten wird als Vorgänger entweder ein anderer Prozessschritt oder eine Verbindung bestimmt. Nachdem die Vorgänger gegebenenfalls festgelegt wurden, müssen bei allen Prozessschritten noch sogenannte Importalternativen definiert werden, welche verschiedene Möglichkeiten zum Import von Daten für den Prozessschritt darstellen. Da eine Importalternative eine oder mehrere Schnittstellen umfassen kann, werden diese zunächst ausgewählt. Anschließend wird die Importalternative angelegt und die zugehörige Nummer eingegeben oder automatisch erzeugt. Für jeden Prozessschritt können beliebig viele Importalternativen definiert werden. Die Be-

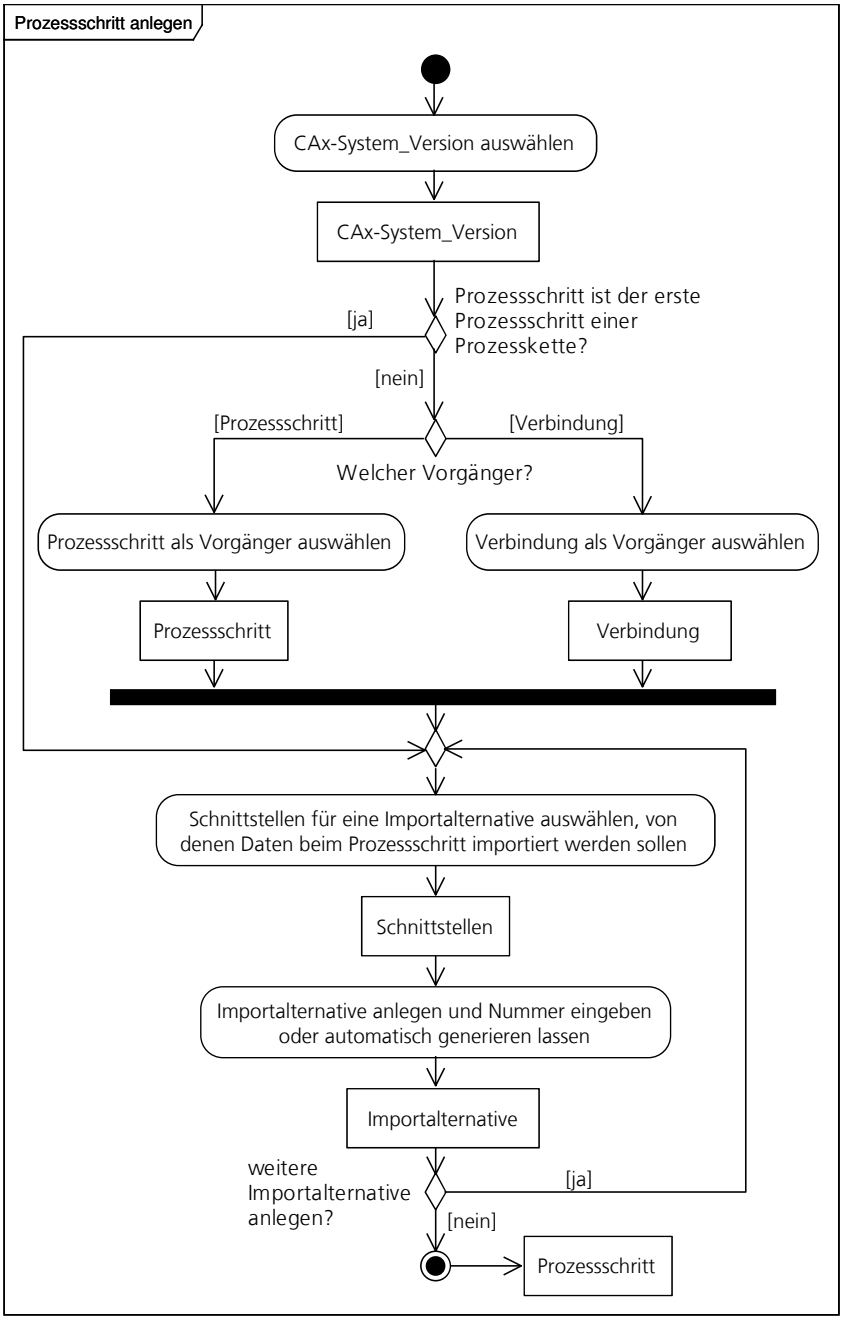


Abbildung 4.14: Aktivitätsdiagramm zum Anwendungsfall »Prozessschritt anlegen«

schreibung des AWF »Prozesskette anlegen« ist unabhängig von einer Implementierung. Das Anlegen kann zum Beispiel rein textbasiert über Eingabefelder umgesetzt werden. Besser wäre jedoch eine Umsetzung durch einen »grafischen Editor«.

AWF »durchgeführte Prozesskette beschreiben«

Beim AWF »durchgeführte Prozesskette beschreiben« sollen Daten über eine durchgeführte Prozesskette abgelegt werden, um sie dauerhaft nachvollziehbar zu machen. Der Ablauf

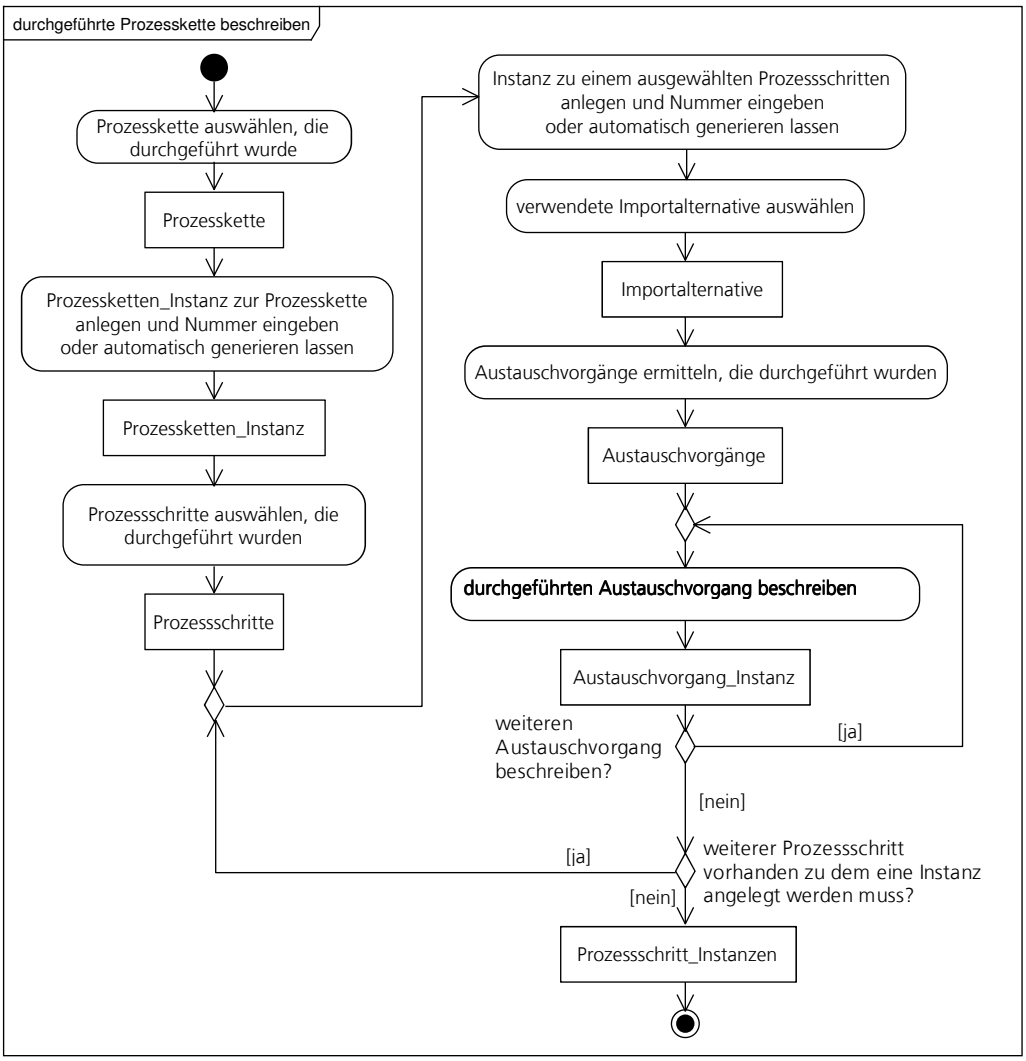


Abbildung 4.15: Aktivitätsdiagramm zum Anwendungsfall »durchgeführte Prozesskette beschreiben«

des AWF ist in einem Aktivitätsdiagramm in Abbildung 4.15 dargestellt. Als erstes muss eine Prozesskette ausgewählt werden, die durchgeführt wurde. Für diese Prozesskette wird eine Instanz angelegt und es muss die Nummer der Instanz eingegeben oder automatisch generiert werden. Anschließend muss der Anwender die Prozessschritte auswählen, die er durchgeführt hat. Dies ist wichtig, da bei Prozessketten alternative Abläufe möglich sind, von denen bei einer durchgeführten Prozesskette aber jeweils nur eine der Alternativen durchgeführt werden konnte. Das zu realisierende Software-System könnte vom Anwender für alle Alternativen abfragen, welche Möglichkeit jeweils ausgewählt wurde, um so die durchgeführten Prozessschritte zu ermitteln.

Zu jedem durchgeführten Prozessschritt wird danach eine Instanz angelegt, zu der eine Nummer eingegeben oder automatisch generiert werden muss. Für einen Prozessschritt können mehrere Importalternativen definiert worden sein, von denen bei einem durchgeführten Prozessschritt aber nur eine verwendet werden konnte. Daher muss für den durchgeführten Prozessschritt die verwendete Importalternative ausgewählt werden. Mit

Hilfe der festgelegten Importalternative lassen sich die zugehörigen Schnittstellen und für diese jeweils die Austauschvorgänge ermitteln, welche durchgeführt wurden. Jeder Austauschvorgang soll vom Anwender im Detail beschrieben werden. Dies ist mit dem AWF »durchgeführten Austauschvorgang beschreiben« möglich, der in den AWF »durchgeführte Prozesskette beschreiben« eingebunden ist (siehe Anwendungsfalldiagramm in Abbildung 4.12).

AWF »durchgeführten Austauschvorgang beschreiben«

Der AWF »durchgeführten Austauschvorgang beschreiben« ist in einem Aktivitätsdiagramm in Abbildung 4.16 dargestellt. Für jeden Austauschvorgang wird eine Instanz angelegt, für die eine Nummer eingegeben oder automatisch generiert sowie ein Datum festgelegt werden muss. Zur Instanz werden die beim Austauschvorgang entstandenen Dateien zugeordnet. Dies sind auf jeden Fall eine oder mehrere Dateien, die ausgetauscht wurden (Austausch-Dateien). Außerdem können Exportinformations- und Importinformations-Dateien entstanden sein, welche auch zugeordnet werden sollen. Neben den Dateien sind auch die Einstellungen beim Export und Import wichtig, die bei einem durchgeführten Austauschvorgang festgelegt wurden. Für diese wird jeweils eine Einstellungs_Instanz angelegt. Zur Ermittlung der festgelegten Einstellungswerte gibt es drei verschiedene Möglichkeiten:

- erstellte Screenshots zur Einstellungs_Instanz zuordnen
- Einstellungswerte manuell eingeben
- Einstellungswerte automatisch ermitteln

Bei der Variante mit der manuellen Eingabe der Einstellungswerte soll der Anwender die Einstellungswerte eingeben. Für die Einstellungen können gegebenenfalls Auswahlmöglichkeiten definiert sein, die bei der manuellen Eingabe berücksichtigt werden sollen. Für den Anwender sollen Auswahlfelder mit den festgelegten Auswahlmöglichkeiten für die Einstellungen vorgegeben werden, wenn Auswahlmöglichkeiten definiert wurden. Für alle weiteren Einstellungen soll ein Eingabefeld zur Eingabe eines Einstellungswertes bereitgestellt werden.

Für die anderen beiden Varianten müssen entweder Screenshots erstellt worden sein, die zur Einstellungs_Instanz zugeordnet werden können, oder es muss ein Parser vorhanden sein, welcher die Einstellungswerte automatisch ermitteln kann.

Die Ermittlung der Einstellungswerte wird zunächst für eine der beiden Einstellungs_Instanzen durchgeführt. Anschließend kann für die zweite Einstellungs_Instanz wieder eine der drei Möglichkeiten ausgewählt werden. Der Grund dafür ist, dass sich die Möglichkeiten beim Export und Import der zwei beteiligten CAx-System_Versionen unterscheiden können. Zum Beispiel kann es sein, dass bei den Exporteinstellungen einer CAx-System_Version die Einstellungswerte in eine Exportinformations-Datei geschrieben und die Einstellungswerte automatisch ermittelt werden können. Bei den Importeinstellungen der anderen CAx-System_Version sind aber auf Grund der Vielzahl an Einstellungen nur Screenshots sinnvoll.

Eine Kombination der drei Möglichkeiten ist im Aktivitätsdiagramm nicht modelliert, da sie sich nicht so leicht darstellen lässt. Sie sollte aber auch unterstützt werden. So könnten

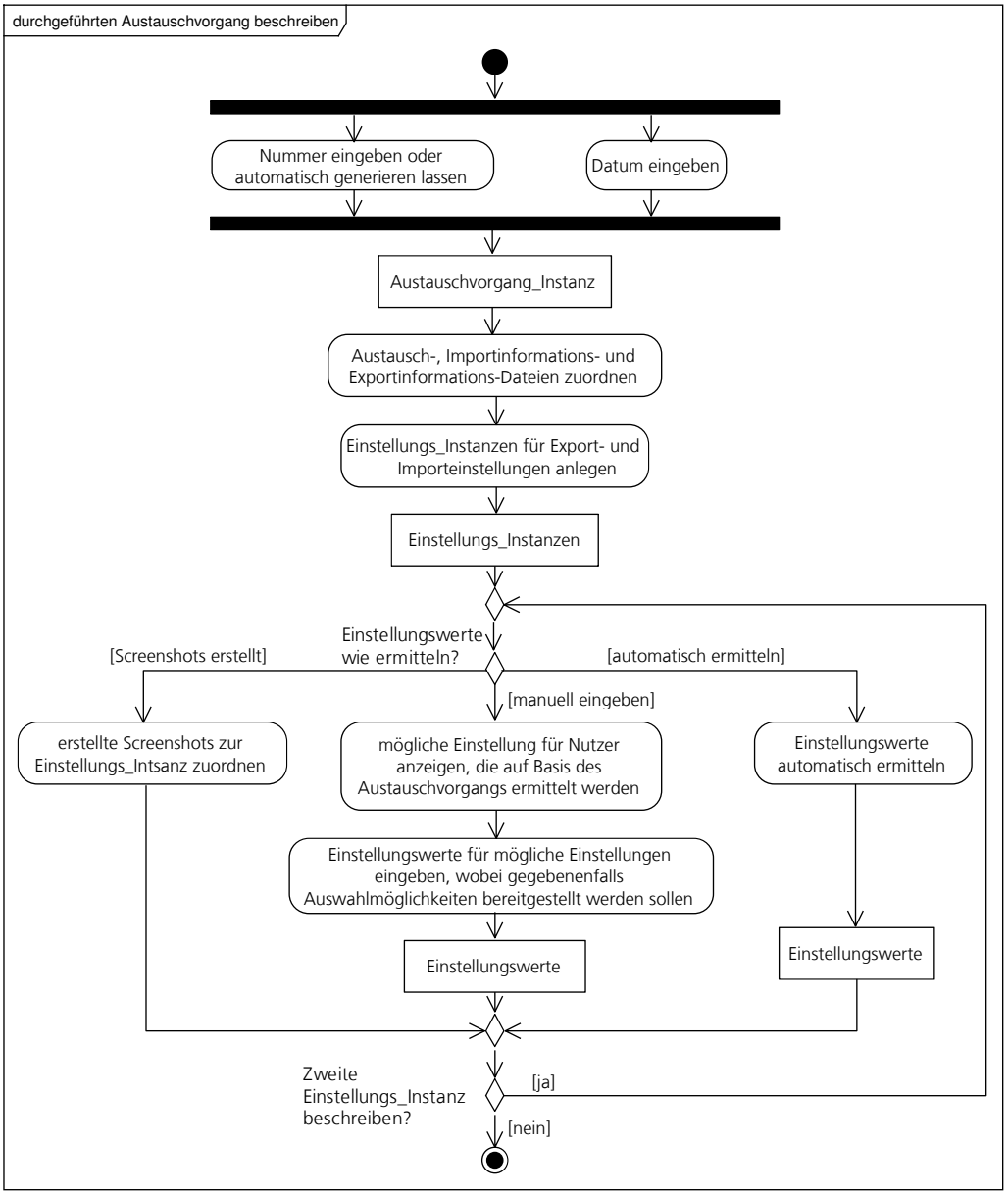


Abbildung 4.16: Aktivitätsdiagramm zum Anwendungsfall »durchgeführten Austauschvorgang beschreiben«

zum Beispiel einige Einstellungswerte vom Anwender manuell eingegeben und zusätzlich Screenshots erstellt werden.

4.3 Zusammenfassung

In diesem Kapitel wurde ein Konzept für ein Software-System zur Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung erläutert. Das Konzept setzt sich aus den beiden Teilkonzepten System- und Prozessverwaltung zusammen. Mit der

Systemverwaltung soll der Überblick über die Systemlandschaft in einem Unternehmen unterstützt werden, indem die vorhandenen CAx-Systeme und deren Schnittstellen verwaltet werden. Die Anwender eines Unternehmens können dadurch zum Beispiel Informationen über CAx-Systeme, deren Export- und Importmöglichkeiten und Einstellungen, die beim Export oder Import möglich sind, nachschlagen. Außerdem können die Schnittstellen zwischen CAx-Systemen beschrieben werden, so dass ein Anwender sich über die Möglichkeiten zum Austausch von Daten zwischen zwei CAx-Systemen informieren kann. Dazu zählt zum Beispiel welche Dateiformate verwendet werden können, welche Unterschiede die Dateiformate bieten und was allgemein beachtet werden muss.

Mit der Prozessverwaltung sollen durchgeführte Datenaustauschprozesse verwaltet werden können, um somit deren Nachvollziehbarkeit zu ermöglichen. Dafür können Prozessketten inklusive Alternativen und Kopplungen definiert werden. Mit Hilfe der festgelegten Prozessketten können die Anwender in einem Unternehmen zum einen Informationen über den Datenaustausch zwischen CAx-Systemen nachschlagen. Dazu zählt zum Beispiel die Beschreibung von Alternativen, so dass die Anwender bei der Wahl der geeigneten Vorgehensweise zum Datenaustausch unterstützt werden. Zum anderen dienen die festgelegten Prozessketten als Grundlage für die Beschreibung von durchgeführten Datenaustauschprozessen, welche als Instanzen der definierten Prozessketten verwaltet werden. Nachdem alle Daten über einen durchgeführten Datenaustauschprozess in der Prozessverwaltung eingetragen wurden, kann dieser über die Prozessverwaltung jederzeit nachvollzogen werden. Zum Beispiel können die durchgeführten Einstellungen sowie die beim Datenaustausch verwendeten Dateien nachgeschlagen werden.

Für beide Teilkonzepte wurden ein Datenmodell und Anwendungsfälle erläutert. Zwischen den Datenmodellen der beiden Teilkonzepte gibt es Verbindungen, so dass sich durch die Kombination beider Datenmodelle ein Gesamtdatenmodell ergibt, welches in Abbildung 4.17 dargestellt ist. Es umfasst alle Klassen der beiden Teilkonzepte. Die Legende und die Verbindungen wurden bereits in Abschnitt 4.2.2 bei der Beschreibung des Datenmodells der Prozessverwaltung erläutert. Das Gesamtdatenmodell stellt eines der wesentlichen Ergebnisse dieser Arbeit dar, weil damit alle Daten und ihre Beziehungen, die zu Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung notwendig sind, beschrieben werden.

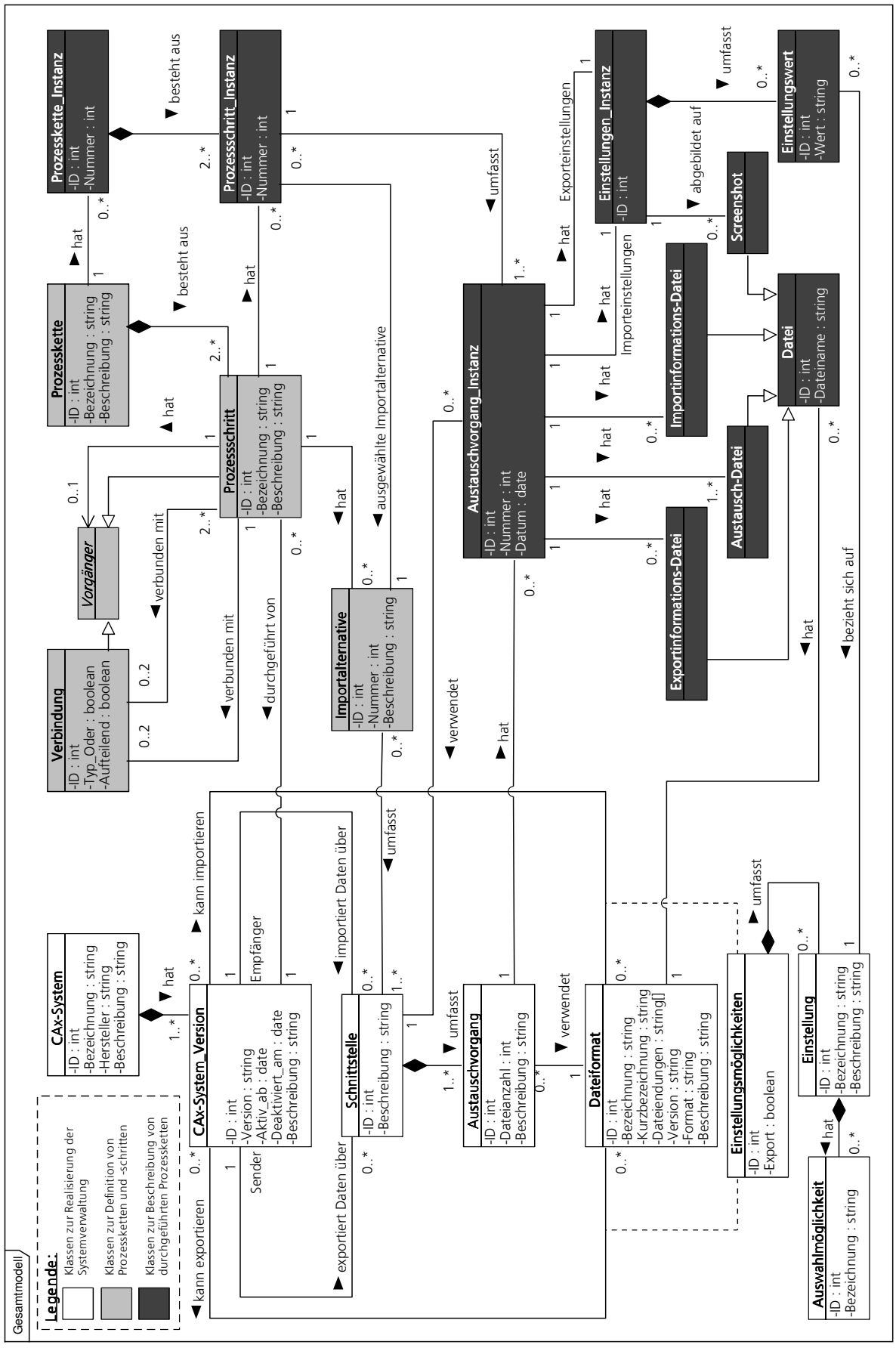


Abbildung 4.17: Klassendiagramm »Gesamtmodell«

5 Implementierung

In diesem Kapitel wird eine prototypische Implementierung des im vorangehenden Kapitel beschriebenen Konzeptes zur Unterstützung des dateibasierten Datenaustauschs erläutert. Zur Umsetzung wurde ein bestehender Prototyp verwendet, welcher um im Konzept beschriebenen Funktionen zur System- und Prozessverwaltung erweitert wurde. Der Grund dafür ist, dass das entwickelte Konzept in ein Forschungsprojekt eingebunden ist, für das ein einheitlicher Prototyp entwickelt werden soll. Die Architektur und die verwendeten Technologien des Prototypen werden in Abschnitt 5.1 beschrieben. Anschließend wird die Erweiterung der Datenhaltung in Abschnitt 5.2 sowie die Erweiterung der Präsentation und Anwendungslogik des Prototypen in Abschnitt 5.3 erläutert, durch welche das Konzept umgesetzt wurde.

5.1 Prototyp

In diesem Abschnitt soll der bestehende Prototyp beschrieben werden, der für die Umsetzung des Konzeptes verwendet wurde. Im folgenden Abschnitt wird zunächst die Architektur des Prototypen erläutert. In Abschnitt 5.1.2 wird ein »Framework« beschrieben, das bei der Entwicklung des Prototypen eingesetzt wird.

5.1.1 Architektur

Zur Realisierung des Prototypen wurde eine webbasierte »Client-Server-Architektur« gewählt, welche in Abbildung 5.1 dargestellt ist. Die Architektur lässt sich in die drei Schichten »Präsentation«, »Anwendungslogik« und »Datenhaltung« unterteilen. Die Präsentationsschicht wird vom Client realisiert. Er stellt die Benutzeroberfläche für den Anwender bereit und nimmt Interaktionen entgegen. Die Benutzeroberfläche wird durch »Hypertext Markup Language (HTML)-«, »Cascading Style Sheets (CSS)-« und »JavaScript-Dateien« realisiert. Als Client kann ein beliebiger Webbrowser verwendet werden, der JavaScript unterstützt.

Auf der Serverseite kommen der Webserver »Apache Tomcat« und eine relationale Datenbank zum Einsatz. Auf dem Webserver läuft die Webanwendung, welche in der objektorientierten Programmiersprache Java geschrieben ist und die Anwendungslogik realisiert. Die Kommunikation zwischen dem Webbrowser und -server findet über das »Hypertext Transfer Protocol (HTTP)« statt. Die Datenhaltungsschicht wird durch eine relationale Datenbank realisiert. Die Datenbank wurde mit dem objektrelationalen »Open-Source« »Datenbankmanagement-System (DBMS)« »PostgreSQL« in der Version »8.4« erstellt und wird von diesem verwaltet. Die Datenbank speichert alle persistenten Daten der Webanwendung. Die Kommunikation zwischen der Webanwendung und der Datenbank findet über die »Java Database Connectivity (JDBC)«-Schnittstelle statt.

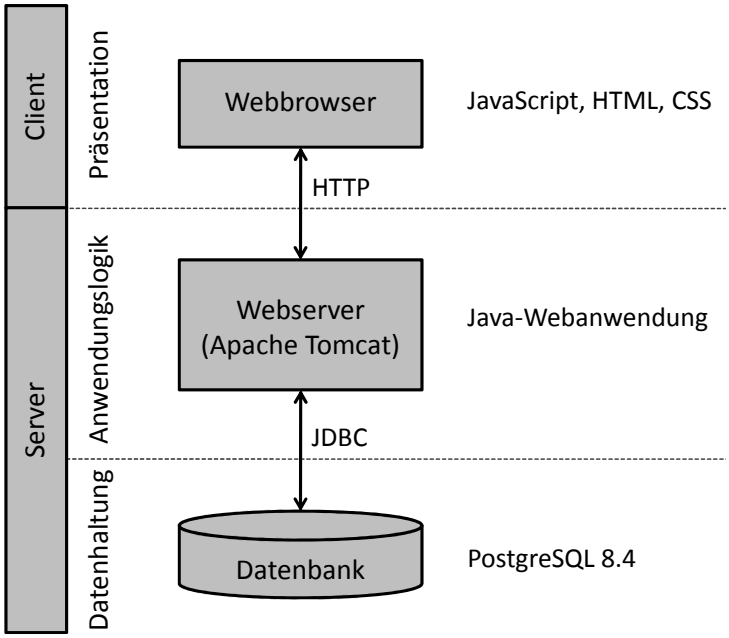


Abbildung 5.1: Architektur des Prototypen

Zur Umsetzung des Konzeptes mussten Erweiterungen an den drei Schichten der Architektur durchgeführt werden. Bevor die Änderungen beschrieben werden, wird im folgenden Abschnitt das bei der Entwicklung des Prototypen eingesetzte Framework »Google Web Toolkit« erläutert.

5.1.2 Google Web Toolkit

Das Google Web Toolkit (GWT)[1] ist ein Framework, mit dem sich sogenannte »Ajax-Webanwendungen« realisieren lassen [Friedrichsen u. Henning, 2006; Schäfer u. Taboada, 2010]. Die Abkürzung Ajax steht für »Asynchronous JavaScript and XML«. Der Begriff geht auf einen Artikel von Jesse James Garret zurück, der 2005 veröffentlicht wurde[2]. Der Begriff Ajax fasst verschiedene Technologien zusammen, mit denen sich Webanwendungen entwickeln lassen, die sich annähernd wie Desktop-Anwendungen verhalten [Paulson, 2005].

Bei den klassischen Webanwendungen wird die grafische Benutzeroberfläche auf dem Server erzeugt. Jede Interaktion bewirkt, dass der Webbrowser eine Anfrage an den Webserver stellt, welcher eine komplett neue Webseite generiert und an den Webbrowser schickt [Friedrichsen u. Henning, 2006, S. 52]. Bei diesem Ansatz müssen viele Daten zwischen Webserver und -browser übertragen werden. Auch Daten, die sich nicht geändert haben, werden erneut übertragen. Außerdem kann der Nutzer in der Zeit, in der die Seite neu geladen wird, nicht mit ihr interagieren. Zum Teil muss der Nutzer mehrere Sekunden warten und sich zusätzlich auf der aktualisierten Seite wieder neu orientieren.

[1] `http://code.google.com/webtoolkit/` (zuletzt überprüft am 10.04.2011)
[2] `http://www.adaptivepath.com/ideas/e000385` (zuletzt überprüft am 10.04.2011)

Bei der Realisierung mit Ajax gibt es die beschriebenen Probleme nicht. Der Grund ist, dass Daten zwischen einem Webbrowser und -server asynchron übertragen werden können. Für den Nutzer bleibt die Webseite dabei weiterhin ansprechbar. Im Webbrowser läuft dafür eine sogenannte JavaScript-basierte »Ajax engine (AE)«. Eine Interaktion des Nutzers wird von der AE verarbeitet. Teilweise kann die AE ohne den Webserver die notwendigen Änderungen an der Webseite durchführen, wie zum Beispiel eine Validierung von eingegebenen Werten in einem Formular. Wenn die AE Daten vom Webserver benötigt, fordert sie nur Teile der Webseite oder sogar nur reine Daten an. Die Übertragung der Daten vom Webserver ist neben XML mittlerweile auch mit anderen Formaten, wie zum Beispiel »JavaScript Object Notation (JSON)[1]«, möglich. Die AE baut den geänderten Teil in die Webseite ein, ohne dass diese komplett neu geladen werden muss.

Gegenüber den klassischen Webanwendungen eignet sich Ajax vor allem für Webanwendungen, die viele Benutzerinteraktionen erfordern. Außerdem ergeben sich Vorteile bei dynamischen Informationen, wie zum Beispiel Aktienkurse, da einzelne Teile einer Webseite mit Ajax aktualisiert werden können.

Die Umsetzung von Ajax-Webanwendungen ist aber auf Grund der JavaScript-Programmierung aufwändig. Zu den JavaScript-Dateien müssen zusätzlich HTML- und CSS-Dateien erstellt und aufeinander abgestimmt werden. Zur Unterstützung der Erstellung sind daher eine Vielzahl von Frameworks entstanden. Ein Framework ist das GWT. Das Besondere an dem GWT ist, dass damit die vollständige Entwicklung von Ajax-Webanwendungen in Java möglich ist. Die für den Client benötigten HTML- und JavaScript-Dateien können aus dem Java-Code erzeugt werden. Das GWT stellt dafür Java-Bibliotheken bereit, die mit dem »GWT-Compiler« übersetzt werden können. Für die Gestaltung der webbasierten Benutzeroberfläche gibt es speziell vorgefertigte Komponenten, welche als »Widgets« bezeichnet werden. Beispiele für Widgets sind Button, Checkbox, Textbox, Tabelle, Scrollleiste und Menüleiste.

Für die wichtigsten Browser wird vom GWT-Compiler eine angepasste JavaScript-Datei erzeugt. Dadurch wird sichergestellt, dass jeder Webbrowser nur den notwendigen JavaScript-Code laden muss. Außerdem wird der JavaScript-Code so gut wie möglich komprimiert. Mit dem GWT-Compiler können auch verschiedene Landessprachen einer Webanwendung unterstützt werden. Es wird dann für jede Kombination aus Landessprache und Browser eine angepasste JavaScript-Datei erstellt. Die Vorteile des GWT sind zusammengefasst:

- Vollständige Entwicklung von Ajax-Webanwendungen in Java
  - HTML- und JavaScript-Kenntnisse werden nicht benötigt
  - Vorteile der Objektorientierung von Java können genutzt werden
  - Entwicklungsumgebungen mit vielen Funktionen stehen zur Verfügung
- Erzeugung von optimiertem JavaScript-Code durch den GWT-Compiler
  - Anpassung für verschiedene Browser
  - Anpassung für verschiedene Landessprachen
  - Komprimierung des Quellcodes

[1] `http://www.json.org/` (zuletzt überprüft am 10.04.2011)

5.2 Erweiterung der Datenhaltung

Bei der Realisierung des Konzeptes müssen für die System- und Prozessverwaltung Daten gespeichert und verwaltet werden können. Hierfür können Dateien (zum Beispiel XML-Dateien) oder Datenbanken verwendet werden. Für die Umsetzung bieten sich Datenbanken an, weil sie unter anderem folgende Vorteile gegenüber Dateien bieten (siehe [Saake et al., 2008, S. 7 ff.]):

- Gleichzeitiger Zugriff auf die Daten von verschiedenen Benutzern beziehungsweise Anwendungen ohne gegenseitige Beeinflussung (Mehrbenutzerfähigkeit)
- Gewährleistung von korrekten Datenbankinhalten und der korrekten Änderung von Daten durch Konsistenzüberwachung
- Unterstützung von redundanzarmer Speicherung
- Kontrolle des Zugriffs auf die Daten durch Benutzerrollen und Rechte
- Gewährleistung von Datensicherheit und Unterstützung der Wiederherstellung von Daten nach Systemfehlern

Zur Umsetzung des Konzeptes wurde die bereits vorhandene Datenbank des Prototypen verwendet. Für die Erweiterung wurde ein Datenbankentwurf durchgeführt, der an eine allgemeine Vorgehensweise angelehnt ist, welche in [Saake et al., 2008, S. 128 ff.] beschrieben ist. Der »konzeptionelle Entwurf« wurde bereits im Rahmen der Konzeptentwicklung, die im vorangehenden Kapitel 4 beschrieben wurde, durchgeführt. Das Klassendiagramm, welches in Abbildung 4.17 dargestellt ist, entspricht einem konzeptionellen Schema. Auf der Grundlage des konzeptionellen Schemas wurden zunächst ein »logischer Entwurf« und anschließend eine »Datendefinition« durchgeführt. Diese Schritte werden in den beiden folgenden Abschnitten erläutert.

5.2.1 Logischer Entwurf

Der bestehende Prototyp verwendet eine relationale Datenbank. Daher wurde das konzeptionelle Schema im logischen Entwurf in ein relationales Schema überführt. Die Überführung des konzeptionellen Schemas in Form eines Klassendiagramms kann weitgehend automatisch durchgeführt werden. Eine Klasse wird dabei auf eine Tabelle im relationalen Schema abgebildet. Es gibt aber einige Konstrukte im Klassendiagramm, die sich nicht direkt abbilden lassen. Dazu zählen unter anderem:

- Spezialisierungen und Generalisierungen
- »N:M«-Beziehungen
- Mehrwertige Attribute

Das erstellte relationale Schema ist in Abbildung 5.2 dargestellt. Es beschreibt Tabellen und Beziehungen zwischen diesen. Für jede Tabelle ist die Bezeichnung im grau hinterlegten Kopf dargestellt. Außerdem sind die Attribute beziehungsweise Spalten bei jeder Tabelle aufgelistet. Für die Attribute sind außerdem die Primär- (mit »PK«) und Fremdschlüssel (mit »FKx«) gekennzeichnet. Die Fremdschlüsselbeziehungen sind durch Pfeile dargestellt. Ein Pfeil zeigt dabei von einer Tabelle mit einem Fremdschlüssel auf die Tabelle mit dem zugehörigen Primärschlüssel.

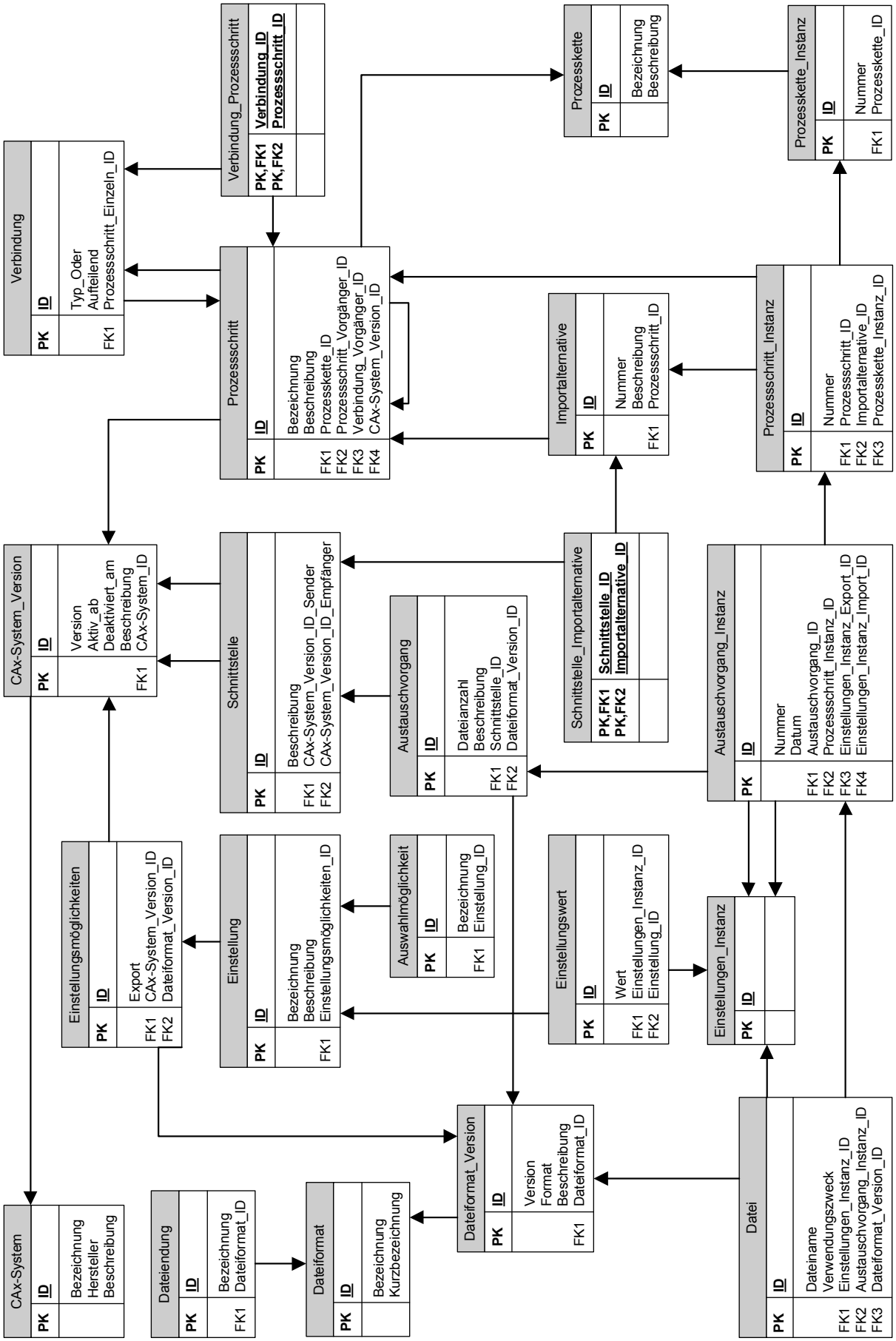


Abbildung 5.2: Relationales Schema der geplanten Datenbank

Abbildung 5.3: Auflösung der Assoziationsklasse »Einstellungsmöglichkeiten«

Einige Klassen konnten auf eine Tabelle im relationalen Schema abgebildet werden. Alle weiteren komplexen Abbildungen sowie durchgeführten Optimierungen sollen im Folgenden erläutert werden. Die Klasse »Dateiformat« beinhaltet das mehrwertige Attribut »Dateiendungen«. Zur Abbildung auf das relationale Schema wurde eine zusätzliche Tabelle »Dateiendung« erstellt. Außerdem wurden zur Normalisierung das Attribut »Version« und »Format« in eine weitere Tabelle (»Dateiformat_Version«) ausgelagert.

Die Klassen »Austausch-«, »Importinformations-« und »Exportinformations-Datei« sowie die Klasse »Screenshot« sind Spezialisierungen der Klasse »Datei«. Spezialisierungen können, wie bereits erwähnt, nicht direkt in ein relationales Schema abgebildet werden. Eine mögliche Abbildung ist die Tabelle »Datei« mit den Attributen aller vier Klassen und einem weiteren Attribut »Verwendungszweck«, mit dem der Typ der Datei festgelegt werden kann. Durch die Abbildung der Klassen auf eine Tabelle werden auch die Assoziationen zu den Klassen »Austauschvorgang_Instanz« und »Einstellungen_Instanz« durch diese Tabelle abgebildet. Da nach dem konzeptionellen Schema eine Datei nur zu einer der beiden Klassen zugeordnet werden kann, muss bei der Implementierung in einer Datenbank darauf geachtet werden, dass nur ein Verweis für jeden Eintrag in der Tabelle »Datei« möglich ist.

Die Klasse »Vorgänger« ist eine abstrakte Klasse und dient zur besseren Strukturierung des Klassendiagramms. Da von der Klasse »Vorgänger« keine Objekte erzeugt werden können, muss bei der Abbildung ins relationale Schema keine Tabelle für diese Klasse erstellt werden. Die zwei unterschiedlichen Varianten für die Vorgänger eines Prozessschritts müssen aber abgebildet werden. Die Überführung in das relationale Schema orientiert sich an der alternativen Darstellung für die Vorgänger, die in Abbildung 4.9 dargestellt ist. In der Tabelle »Prozessschritt« gibt es die zwei Fremdschlüssel »Prozessschritt_Vorgänger_ID« und »Verbindung_Vorgänger_ID«, um den Vorgänger festlegen zu können. Bei der Implementierung in einer Datenbank muss darauf geachtet werden, dass immer nur ein Vorgänger festgelegt werden kann.

Weiterhin gibt es im Klassendiagramm mehrere N:M-Beziehungen, die nicht direkt ins relationale Schema abgebildet werden konnten. Die N:M-Beziehung zwischen den beiden Klassen »CAx-System_Version« und »Dateiformat« fällt beim Auflösen der Assoziationsklasse weg. Dies ist in Abbildung 5.3 dargestellt. Die anderen beiden N:M-Beziehungen zwischen den Klassen »Schnittstelle« und »Importalternative« sowie zwischen »Verbin-

dung« und »Prozessschritt« wurden jeweils durch eine zusätzliche Tabelle (»Schnittstelle_Importalternative« und »Verbindung_Prozessschritt«) abgebildet.

5.2.2 Datendefinition

Auf Basis des relationalen Schemas wurden die Tabellen in der PostgreSQL-Datenbank implementiert, die beim vorhandenen Prototypen bereits vorgegeben ist. Zur Implementierung wurde die Datendefinitionssprache von PostgreSQL verwendet, welche an den SQL-Standard angelehnt ist. Damit wurde ein für PostgreSQL angepasstes Schema in Form eines SQL-Skripts erstellt. Die Bezeichnung der Tabellen und Spalten im Skript wurden in Anlehnung an den Prototypen in Englisch definiert. Das SQL-Skript ist im Anhang C dargestellt.

Es wurden alle Tabellen, wie im relationalen Schema vorgegeben, implementiert bis auf die Tabelle »Datei«. Der Grund dafür ist, dass im bestehenden Prototyp bereits eine Dateiverwaltung implementiert ist. In diesem Zusammenhang gibt es die Tabelle »files«, welche unter anderem Spalten für die »id« und den »Dateinamen« einer Datei enthält. Um die fehlenden Daten, die im Konzept in Bezug zu Dateien definiert wurden, speichern zu können, wurde noch eine zusätzliche Tabelle »file_file_format_version« erstellt, mit der die Einträge in der Tabelle »files« um zusätzliche Informationen erweitert werden können. Dafür wird die »id« der Tabelle »files« als Primär- und Fremdschlüssel verwendet. In der Tabelle »file_file_format_version« können das Attribut »Verwendungszweck« und die Zuordnungen zu Einträgen in den Tabellen »Einstellungen_Instanz«, »Austauschvorgang_Instanz« und »Dateiformat_Version« abgespeichert werden. Im Folgenden wird das erstellte SQL-Skript genauer beschrieben.

Verwendete Datentypen

Bei der Definition eines SQL-Skripts zum Anlegen von Tabellen muss für jede Spalte ein Datentyp festgelegt werden. Die gewählten Datentypen im erstellten SQL-Skript lehnen sich an die im konzeptionellen Schema festgelegten Datentypen für die Attribute an (siehe Abbildung 4.17). Die im SQL-Skript verwendeten Datentypen von PostgreSQL 8.4 sind in Tabelle 5.1 beschrieben. Eine ausführliche Beschreibung der Datentypen befindet sich in der Dokumentation von PostgreSQL 8.4[1].

Der Datentyp »SERIAL« soll genauer beschrieben werden. Er bietet sich bei PostgreSQL als Datentyp für IDs an. Durch die Zuweisung des Datentyps SERIAL zu einer Spalte wird für diese automatisch eine »Sequenz« erzeugt. Eine Sequenz ist ein Datenbankobjekt und kann zur Generierung von ganzen Zahlen für Primärschlüssel verwendet werden. Der Wert einer Sequenz wird nach der Verwendung für einen Eintrag in der Datenbank automatisch erhöht. Durch die Verwendung von SERIAL können einige Befehle eingespart werden, wodurch sich ein übersichtliches SQL-Skript erstellen lässt. Intern wird im DBMS aber an Stelle von SERIAL der Datentyp »INTEGER« verwendet.

[1] `http://www.postgresql.org/docs/8.4/static/datatype.html` (zuletzt überprüft am 10.04.2011)

Tabelle 5.1: Verwendete Datentypen von PostgreSQL 8.4

Datentyp	Beschreibung
SERIAL	Spezieller Datentyp für selbstzählende Zahlen
INTEGER	Ganze Zahlen von -2.147.483.648 bis 2.147.483.648
BOOLEAN	Wahrheitswerte: wahr (TRUE) oder falsch (FALSE)
VARCHAR(n)	Variable Zeichenketten mit maximal n Zeichen
TEXT	Variable Zeichenketten ohne Obergrenze
DATE	Datum bestehend aus Jahr, Monat und Tag

Mit dem folgenden Code wird ein einfaches Beispiel für die Verwendung von SERIAL gezeigt, womit eine Tabelle »beispiel« mit der Spalte »id« als Primärschlüssel angelegt werden kann:

```
CREATE TABLE beispiel
(
  id SERIAL PRIMARY KEY
);
```

Listing 5.1: Beispiel zur Verwendung des Datentyps »SERIAL«

Äquivalent dazu kann der folgende Code verwendet werden, bei dem an Stelle von SERIAL der Datentyp INTEGER für die Spalte »id« verwendet wird und die Sequenz zusätzlich definiert sowie der Spalte »id« zugewiesen werden muss:

```
CREATE SEQUENCE beispiel_id_seq;
CREATE TABLE beispiel
(
  id INTEGER PRIMARY KEY DEFAULT nextval('beispiel_id_seq')
);
```

Listing 5.2: Äquivalenter Code ohne Verwendung des Datentyps »SERIAL«

CHECK-Constraints

Neben den Datentypen wurden im SQL-Skript einige »CHECK-Constraints« definiert, mit denen sich bestimmte Regeln festlegen lassen. Beim Einfügen eines Eintrags in eine Tabelle werden die Regeln überprüft. Nur wenn die Regeln erfüllt sind, wird der Eintrag in der Tabelle gespeichert. Ein CHECK wurde bei der Tabelle »file_format_versions« definiert, um drei verschiedene Typen für die Spalte »format« vorzugeben. Falls weitere Formate benötigt werden, müssen sie an dieser Stelle festgelegt werden. Bei der Tabelle »file_file_format_version« wurden die möglichen Typen auch mit Hilfe eines CHECK-Constraints festgelegt. Weiterhin wurde CHECK noch bei der Tabelle »exchanges« verwendet, um zu überprüfen, ob ein einzufügender Wert in die Spalte »numberoffiles« größer Null ist. Dies ist wichtig, da es keine negative Dateianzahl geben kann.

Wie im vorangehenden Abschnitt beschrieben wurde, soll bei den beiden Tabellen »Datei« und »Prozessschritt« von zwei Fremdschlüsseln nur einer festgelegt werden können. Dies wurde bei den Tabellen »file_file_format_version« und »process_steps« auch mit Hilfe eines CHECK-Constraints realisiert.

Das implementierte Datenbankschema ist in Abbildung 5.4 dargestellt. Es zeigt die Tabellen, die Spalten und die festgelegten Datentypen. Außerdem sind die Fremdschlüsselbeziehungen durch Pfeile dargestellt. Sie zeigen ausgehend von der Tabelle, die einen Fremdschlüssel enthält, auf die zugehörige Tabelle mit dem Primärschlüssel.

5.3 Erweiterung der Präsentation und Anwendungslogik

Zur Umsetzung der im Konzept beschriebenen funktionalen Anforderungen mussten die Präsentation und Anwendungslogik des Prototypen erweitert werden. Dies wurde durch die Implementierung von Java-Klassen realisiert. Dabei wurde das in Abschnitt 5.1.2 beschriebene Framework Google Web Toolkit verwendet. Bei der Implementierung der Java-Klassen wurden die beim Prototypen vorhandenen Entwurfsmuster eingehalten. Zur Trennung der Präsentation von der Anwendungslogik wird das Entwurfsmuster »Model-View-Presenter (MVP)« [Alles et al., 2006] verwendet. Bevor die Erweiterung der Präsentation und Anwendungslogik in Abschnitt 5.3.2 an einem Beispiel beschrieben wird, wird im folgenden Abschnitt zunächst das MVP-Muster erläutert.

5.3.1 Entwurfsmuster »Model-View-Presenter«

Das MVP-Muster ist aus dem Entwurfsmuster »Model-View-Controller« hervorgegangen. Im Gegensatz dazu erfolgt aber eine klarere Trennung der Darstellung, Präsentationslogik und Daten, die dargestellt werden sollen. Die einzelnen Elemente des MVP-Musters sind in Abbildung 5.5 dargestellt und werden im Folgenden erläutert:

- **Model:** Das Modell stellt alle Daten bereit, die in der View dargestellt werden sollen. Es kann dafür auch Anwendungslogik umfassen, um die Daten zu erzeugen oder aus einer Datenbank auszulesen. Die Steuerung des Modells erfolgt vom Presenter. Das Model selbst kennt weder die View noch den Presenter.
- **View:** Die View enthält keine Logik und ist nur für die Darstellung (Benutzeroberfläche) und Ein- und Ausgaben (Benutzerinteraktionen) zuständig. Sie erhält weder Zugriff auf die Funktionalität des Presenters noch auf das Model. Die gesamte Steuerung der View erfolgt vom Presenter.
- **Presenter:** Der Presenter realisiert die Präsentationslogik. Er ist das Bindeglied zwischen Model und View und steuert die logischen Abläufe zwischen den beiden Elementen. Der Presenter definiert ein »Interface« für die zugehörige View. Dadurch ist die View austauschbar. Benutzerinteraktionen werden von der View entgegengenommen, an den Presenter weitergeleitet und vom ihm verarbeitet. Das Model kann mit dem Presenter aktualisiert und ausgelesen werden.

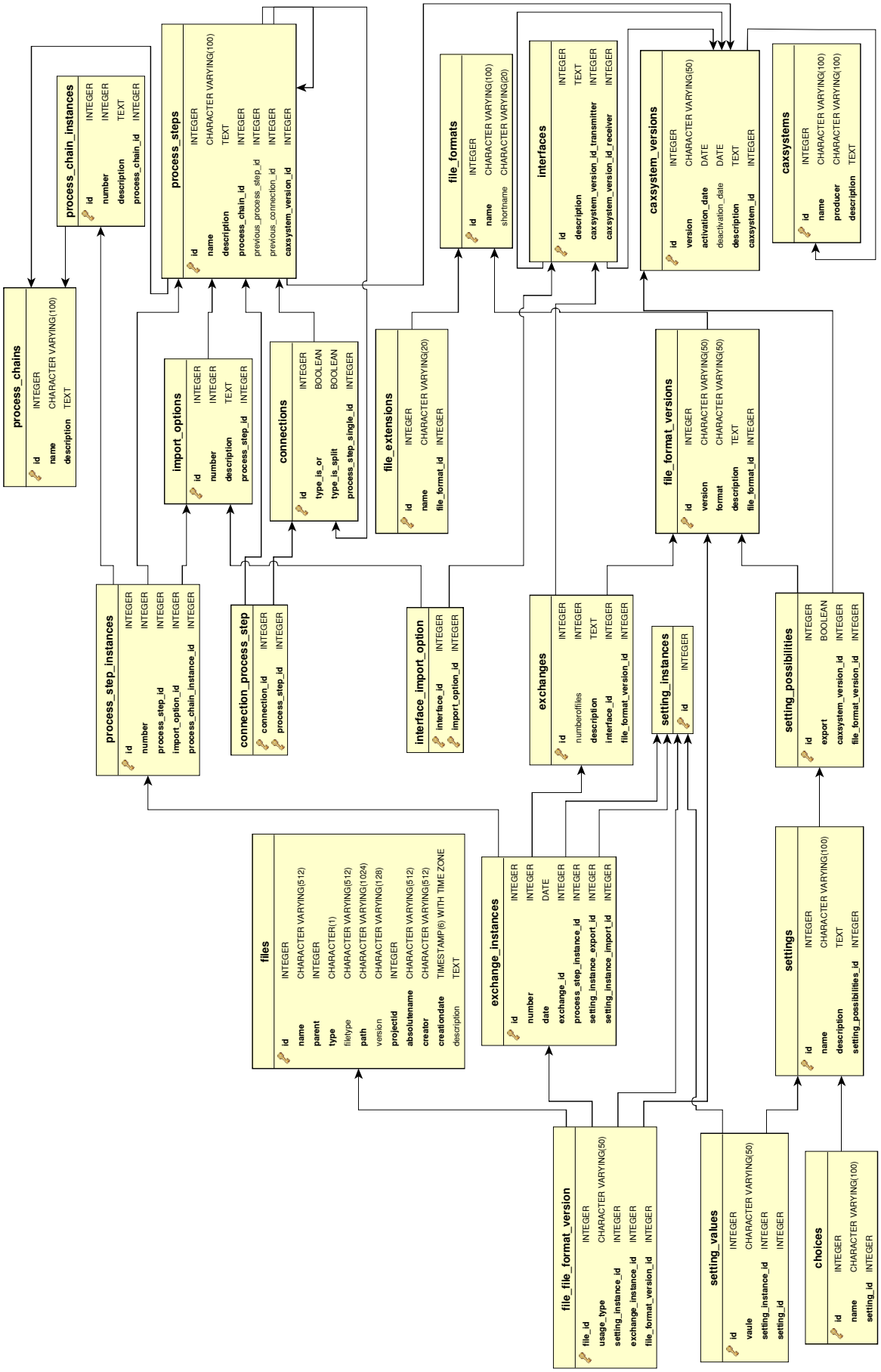


Abbildung 5.4: Implementiertes Datenbankschema

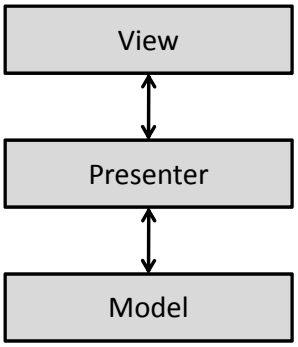


Abbildung 5.5: Entwurfsmuster »Model-View-Presenter«

5.3.2 Realisierung der Erweiterung

Die Erweiterung der Präsentation und Anwendungslogik wird in diesem Abschnitt anhand eines Beispiels erläutert. Als Beispiel wurde die Funktion der Systemverwaltung zum Anzeigen der gespeicherten CAx-Systeme gewählt. Im Rahmen der Umsetzung wurden dafür die Klassen, welche im Klassendiagramm in Abbildung 5.6 dargestellt sind, implementiert.

Die Klassen sind an das MVP-Muster angelehnt. Die Klasse »SystemsOverviewView« entspricht der View. In ihr ist der Aufbau der Benutzeroberfläche mit Hilfe von Widgets definiert (siehe Abschnitt 5.1.2). Auf der Benutzeroberfläche werden die CAx-Systeme und ihrer Eigenschaften in einer Tabelle angezeigt. Die Klasse »SystemsOverviewView« implementiert das Interface »Display«, welches vom Presenter definiert wird.

Der Presenter wird durch die Klasse »SystemsOverviewPresenter« realisiert. Durch den »rpcService« können asynchrone Anfragen an den Server gestellt werden. Über den »event-Bus« können Events, wie zum Beispiel das Anklicken eines Menüeintrags, ausgelöst werden.

Aus den Klassen »SystemsOverviewView« und »SystemsOverviewPresenter« werden vom GWT-Compiler die für die Darstellung der Webanwendung im Client benötigten HTML- und JavaScript-Dateien erzeugt.

Die Anwendungslogik, die auf dem Server ausgeführt wird, entspricht dem Model. Dieses wird durch verschiedene Klassen realisiert. Über den »rpcService« kann der Presenter die Klasse »GetAllCAXSystemsHandler« aufrufen und die in der Datenbank vorhandenen CAx-Systeme anfordern. Die Klasse »GetAllCAXSystemsHandler« verwendet zum Auslesen der CAx-Systeme aus der Datenbank die Klasse »CAXSystemDAOJDBC«. Die Klasse »CAXSystemDAOJDBC« stellt außerdem weitere Methoden bereit, um Daten bezüglich der CAx-Systeme in der Datenbank verändern oder auslesen zu können. Die Klasse »CAXSystem« entspricht der Implementierung der Klasse »CAx-System« aus dem Konzept (siehe Abbildung 4.17).

Der Aufruf der Klassen beim Anzeigen der CAx-Systeme ist im Sequenzdiagramm in Abbildung 5.7 dargestellt. Nachdem ein Anwender im Menü den entsprechenden Eintrag ausgewählt hat, um die CAx-Systeme anzeigen zu lassen, fordert der »SystemsOverview-Presenter« die CAx-Systeme vom Model an. Dafür wird über den »rpcService« der »Get-

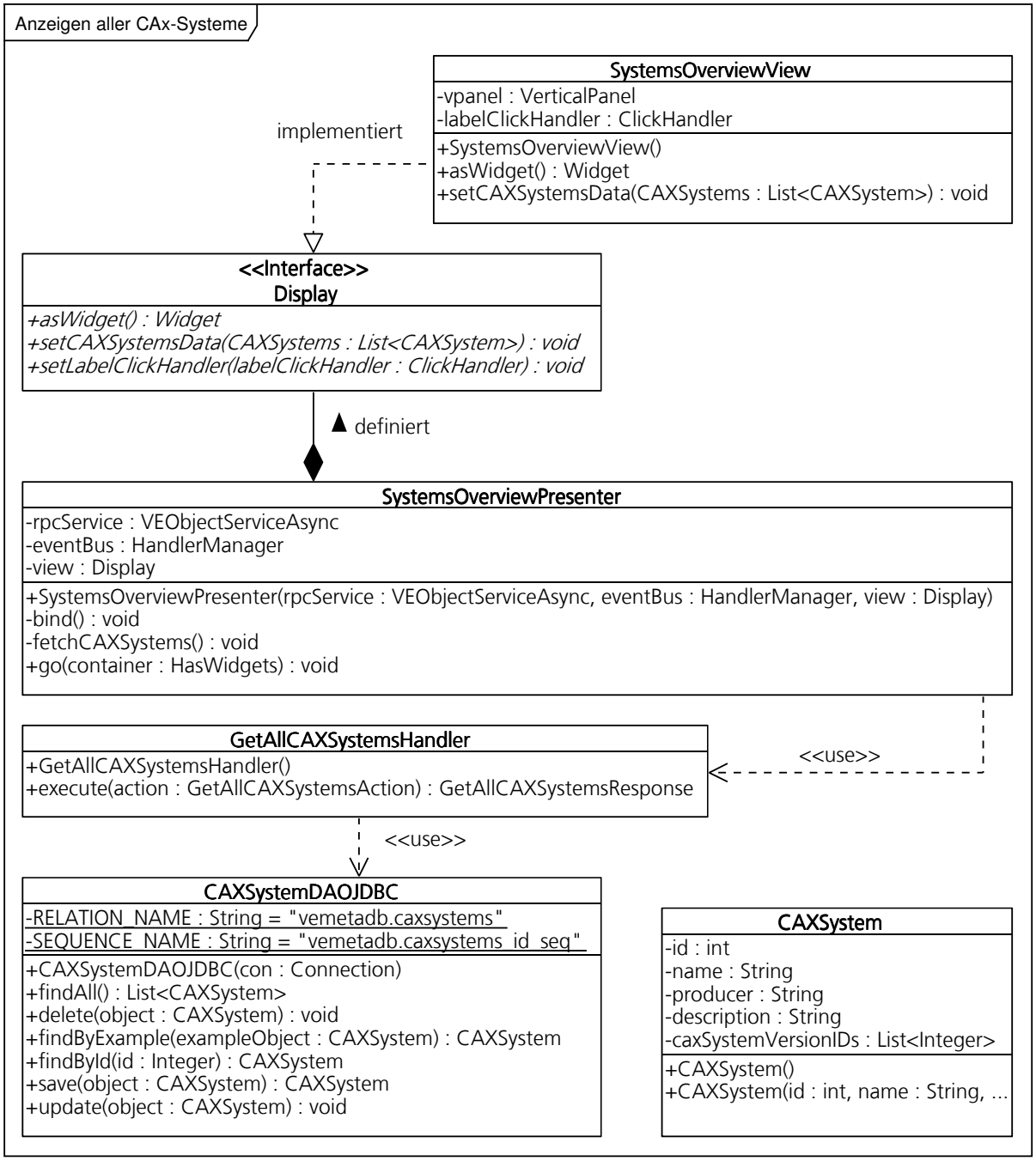


Abbildung 5.6: Klassendiagramm »Anzeigen aller CAx-Systeme«

AllCAXSystemsHandler« aufgerufen. Dieser wiederum liest mit Hilfe der Klasse »CAX-SystemDAOJDBC« und der Methode »findAll()« alle in der Datenbank gespeicherten CAx-Systeme aus. Anschließend wird eine Liste mit Objekten der Klasse »CAXSystem« an den Handler und von da an den Presenter zurückgegeben. Der Presenter veranlasst daraufhin, dass die »SystemsOverviewView« die CAx-Systeme darstellt.

Das Ergebnis, welches auf der Benutzeroberfläche angezeigt wird, ist in Abbildung 5.8 dargestellt. Im oberen Bereich befindet sich das Menü des Prototypen, welches grau hin-

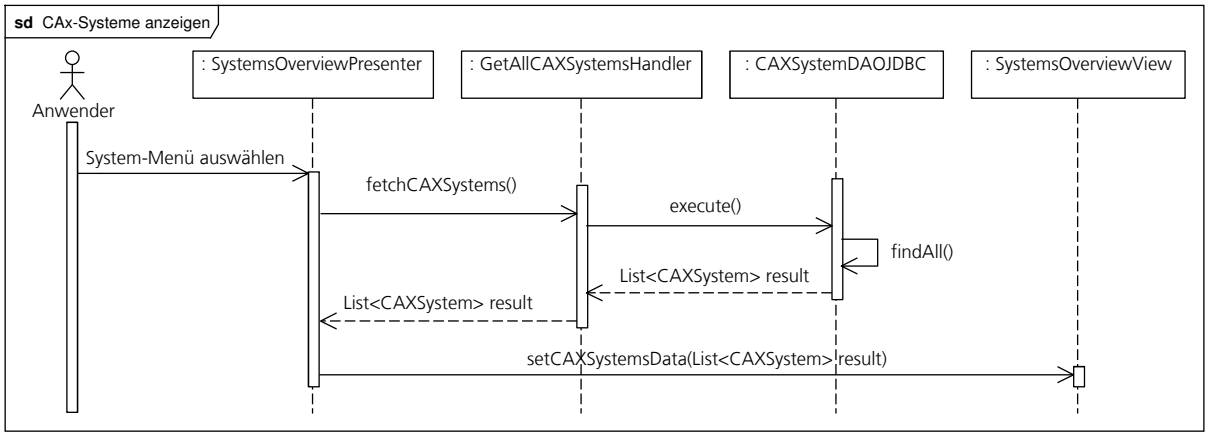


Abbildung 5.7: Sequenzdiagramm »CAx-Systeme anzeigen«

terlegt ist. Die erste Zeile stellt die Hauptmenüeinträge dar. In Abhängigkeit eines gewählten Menüeintrags werden in der zweiten Zeile weitere Untermenüeinträge angezeigt. Im unteren rechten Bereich befindet sich die Tabelle, welche die gespeicherten CAx-Systeme und ihren Eigenschaften (Bezeichnung, Hersteller, Anzahl der Versionen, Beschreibung) darstellt. Die im Prototyp angelegten Daten basieren auf dem Entwurfsprozess, der in Abschnitt 3.1.3 beschrieben ist.

5.4 Zusammenfassung

In diesem Kapitel wurde anhand einer prototypischen Implementierung eine Möglichkeit zur Umsetzung des in Kapitel 4 beschriebenen Konzeptes erläutert. Zur Umsetzung wurde ein bestehender Prototyp verwendet, welcher um im Konzept beschriebenen Funktionen zur System- und Prozessverwaltung erweitert wurde.

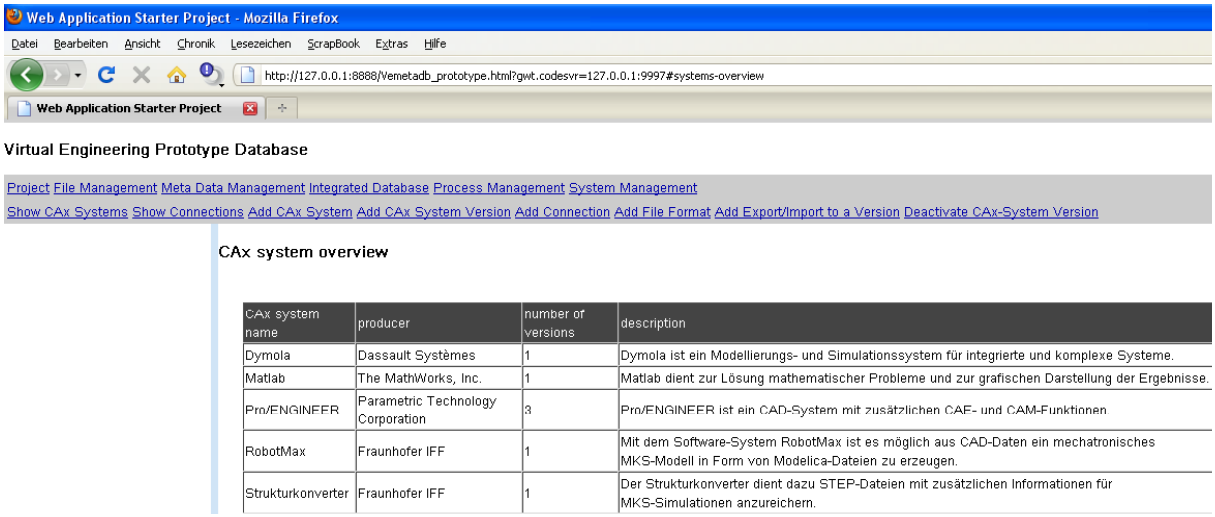


Abbildung 5.8: Anzeigen aller CAx-Systeme im Prototypen

Die Architektur des Prototypen ist eine webbasierte Client-Server-Architektur, bestehend aus Webbrowser, Webserver und einer Datenbank. Die Datenbank wird vom DBMS PostgreSQL in der Version 8.4 verwaltet. Zur Entwicklung des Prototypen wird das Framework Google Web Toolkit (GWT) eingesetzt. Damit lassen sich Ajax-Webanwendungen erstellen, die sich annähernd wie Desktop-Anwendungen verhalten. Die Grundlage dafür ist, dass Daten zwischen einem Webserver und -browser asynchron übertragen werden können. Im Gegensatz zu anderen Frameworks, bei denen HTML- und JavaScript-Code implementiert werden müssen, erfolgt die Entwicklung mit dem GWT vollständig in Java. Aus dem Java-Code können mit Hilfe des GWT-Compilers die für den Webbrowser benötigten HTML- und JavaScript-Dateien erzeugt werden.

Zur Umsetzung des Konzeptes mussten Erweiterungen der drei Schichten »Präsentation«, »Anwendungslogik« und »Datenhaltung« realisiert werden. Für die Erweiterung wurde ein Datenbankentwurf durchgeführt. Dafür wurde zunächst aus dem konzeptionellen Gesamtschema in Form eines Klassendiagramms, welches in Abbildung 4.17 dargestellt ist, ein relationales Schema erstellt, weil beim bestehenden Prototyp eine relationale Datenbank verwendet wird. Die dazu notwendigen Abbildungen vom Klassendiagramm in das relationale Schema wurden beschrieben. Auf Basis des relationalen Schemas wurde ein für PostgreSQL angepasstes Schema in Form eines SQL-Skripts erstellt, welches in Anhang C dargestellt ist. Die dabei verwendeten Datentypen und speziellen Anpassungen wurden beschrieben.

Zur Umsetzung der im Konzept beschriebenen funktionalen Anforderungen mussten die Präsentation und Anwendungslogik des Prototypen erweitert werden. Die Erweiterung wurde anhand eines Beispiels erläutert. Als Beispiel wurde die Funktion der Systemverwaltung zum Anzeigen der gespeicherten CAx-Systeme gewählt. Die im Rahmen der Umsetzung implementierten Klassen wurden beschrieben und der Aufruf der Klassen beim Anzeigen der CAx-Systeme wurde anhand eines Sequenzdiagramms erläutert.

6 Zusammenfassung und Ausblick

In diesem abschließenden Kapitel werden die Ergebnisse der vorangehenden Kapitel zusammengefasst und es wird ein Ausblick für weiterführende Arbeiten gegeben.

Beim dateibasierten Datenaustausch in der Produktentwicklung ergeben sich zwei wesentliche Probleme. Zum einen entstehen komplexe Systemlandschaften, die aus einer Vielzahl an CAx-Systemen und Schnittstellen zwischen diesen bestehen und zum anderen sind durchgeführte Datenaustauschprozesse im Nachhinein schwer nachvollziehbar. Das Ziel dieser Diplomarbeit war daher die Entwicklung eines Konzeptes zur Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung, wobei der Schwerpunkt auf der Entwicklung eines Datenmodells lag, welches die zur Unterstützung notwendigen Daten und ihre Beziehungen untereinander beschreibt.

Anhand von drei Analysen wurden zunächst die Probleme, die sich im Zusammenhang mit dem dateibasierten Datenaustausch ergeben, genauer untersucht. Als erstes wurde der dateibasierte Datenaustausch zwischen verschiedenen CAx-Systemen bei einem Entwicklungsprozess von Industrierobotern untersucht. Als Ergebnis dieser Analyse wurden zwei wesentliche Probleme festgestellt. Zum einen treten auf Grund von verschiedenen Dateiformaten und CAx-Systemen Alternativen und Kopplungen beim Datenaustausch auf und zum anderen können die durchgeführten Datenaustauschprozesse nicht nachvollzogen werden.

Durch zwei vertiefende Analysen wurden diese beiden Probleme detailliert untersucht. Zum einen wurde ein Vergleich von Dateiformaten zum Austausch von CAD-Daten durchgeführt, um Unterschiede von Dateiformaten zu ermitteln. Dabei hat sich herausgestellt, dass im Laufe der Zeit eine Vielzahl an neutralen Dateiformaten entwickelt wurde und zur Zeit neue entwickelt werden. Für die Mitarbeiter in einem Unternehmen ist es schwierig zu wissen, welche Daten mit welchem Dateiformat übertragen werden können. Zum anderen wurde die Nachvollziehbarkeit des Datenaustauschs anhand des CAD-Systems »Pro/ENGINEER Wildfire« analysiert, um zu ermitteln, welche Daten im Zusammenhang mit der Nachvollziehbarkeit von Bedeutung sind. Als Ergebnis wurde ein allgemeiner Ablauf für Datenaustauschprozesse und die dabei anfallenden Daten abgeleitet.

Im Anschluss an die Ergebnisse der durchgeführten Analysen wurde das entwickelte Konzept beschrieben, welches sich aus den beiden Teilkonzepten System- und Prozessverwaltung zusammensetzt. Die Systemverwaltung konzentriert sich auf die Unterstützung der komplexen Systemlandschaften und die Prozessverwaltung auf die Nachvollziehbarkeit von durchgeführten Datenaustauschprozessen. Ausgehend von den Ergebnissen der Analysen wurden für beide Teilkonzepte jeweils Anforderungen spezifiziert, die an ein Datenmodell und an die Funktionen eines Software-Systems gestellt werden. Anschließend wurden die beiden entwickelten Datenmodelle sowie die Anwendungsfälle zur Beschreibung der Funktionen des im Konzept geplanten Software-Systems erläutert.

Zwischen den Datenmodellen der beiden Teilkonzepte gibt es Verbindungen, so dass sich durch die Kombination beider Datenmodelle ein Gesamtdatenmodell ergibt, welches in

Abbildung 4.17 dargestellt ist. Dieses Gesamtdatenmodell stellt ein wesentliches Ergebnis dieser Arbeit dar, weil damit alle Daten und ihre Beziehungen, die zur Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung notwendig sind, beschrieben werden.

Mit dem entwickelten Konzept kann ein Software-System realisiert werden, dass für die Mitarbeiter eines Unternehmens folgende Unterstützung ermöglicht. Sie können mit Hilfe der Systemverwaltung Informationen über die aktuelle und zurückliegende Versionen der Systemlandschaft abrufen. Dies umfasst Informationen über CAx-Systeme und deren Schnittstellen, die zum Beispiel folgende Fragestellungen beantworten können:

- Welche CAx-Systeme gibt es?
- Wie sah die Systemlandschaft zu einem bestimmten Zeitpunkt aus?
- Welche Export- und Importmöglichkeiten bietet ein CAx-System?
- Welche Einstellungen sind beim Export oder Import möglich und was bewirken sie?
- Welche Schnittstellen gibt es, um Daten von CAx-System A nach B auszutauschen?
- Welche Vor- und Nachteile bieten jeweils die Schnittstellen?

Des Weiteren ist es für die Mitarbeiter durch die Prozessverwaltung möglich, Informationen über Datenaustauschprozesse nachzuschlagen sowie Daten über durchgeführte Datenaustauschprozesse zu speichern und diese damit nachvollziehbar zu verwalten. Beispielsweise können folgende Fragestellungen mit Hilfe der Prozessverwaltung beantwortet werden:

- Wie sollen Datenaustauschprozesse durchgeführt werden?
- Welche alternativen Möglichkeiten gibt es dabei?
- Welche Kopplungen müssen berücksichtigt werden?
- Wie wurde ein konkreter Datenaustauschprozess durchgeführt?
- Welche Einstellungen wurden dabei getroffen?

Eine prototypische Umsetzung des entwickelten Konzeptes wurde auf Basis eines bestehenden Prototypen durchgeführt. Zur Umsetzung mussten Erweiterungen der drei Schichten »Präsentation«, »Anwendungslogik« und »Datenhaltung« des Prototypen realisiert werden. Für die Erweiterung der Datenhaltung wurde ein Datenbankentwurf durchgeführt, bei dem ausgehend vom Gesamtdatenmodell des Konzeptes ein relationales Schema und anschließend ein für PostgreSQL spezifisches SQL-Skript erstellt wurde. Zur Umsetzung der im Konzept beschriebenen funktionalen Anforderungen mussten die Präsentation und Anwendungslogik des Prototypen erweitert werden. Die Erweiterung wurde anhand eines Beispiels beschrieben, für das die zur Umsetzung implementierten Klassen erläutert wurden.

Mit der Entwicklung und prototypischen Umsetzung des Konzeptes wurde eine Lösung zur Unterstützung des dateibasierten Datenaustauschs in der Produktentwicklung erarbeitet und damit die Zielstellung dieser Arbeit erreicht. Ausgehend von den beschriebenen Ergebnissen sind aber weiterführende Untersuchungen zur Evaluierung, zur Entwicklung einer kommerziellen Umsetzung und zur Erweiterung des Konzeptes möglich, die im Folgenden erläutert werden.

Um das entwickelte Konzept zu evaluieren, kann der Prototyp in einer realen Entwicklungsumgebung getestet und von Anwendern bewertet werden. Dadurch wird überprüft,

ob die angestrebten Verbesserungen erreicht werden oder ob eine Erweiterung beziehungsweise Anpassung des Prototypen oder des Konzeptes erforderlich ist.

Da das Konzept bisher nur prototypisch umgesetzt wurde, wäre als weiterführende Arbeit die Entwicklung einer kommerziellen Lösung sinnvoll. Hierfür könnte die Umsetzung auf Basis von »Produktdatenmanagement (PDM)-Systemen« untersucht werden, weil sich diese in der Produktentwicklung, zur Verwaltung von Produktdaten und den damit in Verbindung stehenden Prozessen, bereits etabliert haben. PDM-Systeme bieten in der Regel umfangreiche Möglichkeiten zur Anpassung des zu Grunde liegenden Datenmodells. Dadurch kann davon ausgegangen werden, dass sich das Datenmodell des entwickelten Konzeptes in einem PDM-System abbilden lässt. Somit könnte die Verwaltung von Dateien und zugehörigen Metadaten sowie weitere Funktionen, die ein PDM-System bietet, genutzt werden. Eine Möglichkeit für eine weiterführende Arbeit wäre daher die Untersuchung der Umsetzung des Konzeptes anhand von einem oder mehreren PDM-Systemen. In diesem Zusammenhang könnte auch überprüft werden, inwiefern sich Workflowmanagement-Komponenten von PDM-Systemen zur Unterstützung der Datenaustauschprozesse verwenden lassen.

Auf der Basis von zusätzlichen Analysen kann eine Erweiterung des Konzeptes durchgeführt werden. Zum Beispiel können die Datenaustauschprozesse zwischen CAx-Systemen in verschiedenen Unternehmen und Branchen detailliert untersucht werden, um festzustellen, ob es weitere spezifische Probleme gibt. In diesem Zusammenhang könnten auch Interviews mit Anwendern des Anwendungsgebietes durchgeführt werden, um eventuell weitere Anforderungen an die Unterstützung des dateibasierten Datenaustauschs zu ermitteln.

Quellenverzeichnis

Literatur

[Abramovici 2002] Abramovici, Michael: Virtuelle Produktentwicklung - Ein wichtiger Meilenstein auf dem Weg zum Digital Business. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* (2002), Nr. 6, S. 312–315

[Alles et al. 2006] Alles, Micah ; Crosby, David ; Erickson, Carl ; Harleton, Brian ; Marsiglia, Michael ; Pattison, Greg ; Stienstra, Curt: Presenter First: Organizing Complex GUI Applications for Test-Driven Development. In: IEEE (Hrsg.): *Proceedings of AGILE 2006 Conference, 23.-28. Juli, Minneapolis, USA*. 2006. – ISBN 0–7695–2562–8

[Anderl u. Grabowski 2007] Anderl, Reiner ; Grabowski, Hans: Elektronische Datenverarbeitung. In: Grote, Karl-Heinrich (Hrsg.) ; Feldhusen, Jörg (Hrsg.): *Dubbel: Taschenbuch für den Maschinenbau*. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2007. – ISBN 978–3–540–49714–1

[Anderl u. Trippner 2000] Anderl, Reiner (Hrsg.) ; Trippner, Dietmar (Hrsg.): *STEP - STandard for the Exchange of Product Model Data: Eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP)*. Stuttgart : Teubner, 2000. – ISBN 3–519–06377–8

[Bernard 2005] Bernard, A.: Virtual engineering: methods and tools. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* (2005), Nr. 5, S. 413–421

[Bhattacharya et al. 2006] Bhattacharya, Partha ; Welakwe, Nick-Ange S. ; Makanaboyina, Ranga ; Chimalakonda, Adithya: Integration of CATIA with Modelica. In: Kral, Christian (Hrsg.): *Modelica 2006: Proceedings of the 5th International Modelica Conference; 4. und 5. September, Wien, Österreich*, 2006, S. 671–675

[Brecher et al. 2010] Brecher, Christian ; Herfs, Werner ; Fayzullin, Kamil ; Özdemir, Denis ; Feng, Juejing ; Hamadou, Mehdi ; Müller, Andreas: Integration und Durchgängigkeit von Information im Produktionsmittellebenszyklus. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* (2010), Nr. 4, S. 271–276

[Bullinger et al. 2009] Bullinger, Hans-Jörg (Hrsg.) ; Spath, Dieter (Hrsg.) ; Warnecke, Hans-Jürgen (Hrsg.) ; Westkämper, Engelbert (Hrsg.): *Handbuch Unternehmensorganisation: Strategien, Planung, Umsetzung*. 3., neu bearb. Aufl. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2009. – ISBN 978–3–540–72136–9

[Cyon Research 2006] Cyon Research: *The Business Case for a Common Data Distribution Platform: A Look at UGS' JT: White Paper*. 2006

[Deisinger 2010] Deisinger, Joachim: JT optimiert Zusammenarbeit und Transparenz. In: *interface* (2010), Nr. 03, S. 12–13

[Dotzauer u. Martin 2010] Dotzauer, Rudolf ; Martin, Hans-Peter: *JT in the Application - Report from the ProSTEP iViP / VDA JT Activities (Vortrag), ProSTEP iViP Symposium 28.04.2010.* 2010

[Dumke 2003] Dumke, Reiner: *Software Engineering.* 4., überarb. und erw. Aufl. Wiesbaden : Vieweg, 2003. – ISBN 3–528–35355–4

[Dyla 2002] Dyla, Andreas: *Modell einer durchgängig rechnerbasierten Produktentwicklung.* München, Technische Universität München, Diss., 2002

[Eigner 2009] Eigner, Martin: IT-Lösungen für den Produktentwicklungsprozess. In: *[Bullinger et al., 2009].* 2009, S. 247–260

[Eigner et al. 2007] Eigner, Martin ; Bitzer, Michael ; Burr, Holger: Produktlebenszyklus mit Unterbrechungen: Abstimmungsbedarf zwischen Konstruktion und Produktion. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* (2007), Nr. 9, S. 582–586

[Eigner u. Stelzer 2009] Eigner, Martin ; Stelzer, Ralph: *Product Lifecycle Management: Ein Leitfaden für Product Development und Life Cycle Management.* 2., neu bearbeitete Auflage. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2009. – ISBN 978–3–540–44373–5

[Elmqvist u. Mattsson 1997] Elmqvist, Hilding ; Mattsson, Sven E.: An Introduction to the Physical Modeling Language Modelica. In: Hahn, W. (Hrsg.) ; Lehmann, A. (Hrsg.): *Proceedings of the 9th European Simulation Symposium ESS'97, 19.-23. Oktober, Passau.* 1997, S. 110–114

[Elmqvist et al. 1999] Elmqvist, Hilding ; Mattsson, Sven E. ; Otter, Martin: Modelica - A Language for Physical System Modeling. In: IEEE (Hrsg.): *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design, 22.-27. August, Hawaii, USA.* Piscataway, NJ : IEEE Service Center, 1999. – ISBN 0–7803–5500–8, S. 630–639

[Engelson et al. 2003] Engelson, Vadim ; Bunus, Peter ; Popescu, Lucian ; Fritzson, Peter: Mechanical CAD with Multibody Dynamic Analysis Based on Modelica Simulation. In: Erik Dahlquist (Hrsg.): *Proceedings of the 44th Scandinavian Conference on Simulation and Modeling (SIMS2003), 18.-19. September, Västerås, Schweden*, 2003

[Faisst u. Stürken 1997] Faisst, Wolfgang ; Stürken, Momme: *Daten-, Funktions- und Prozeß-Standards für Virtuelle Unternehmen - strategische Überlegungen: Arbeitspapier der Reihe "Informations- und Kommunikationssysteme als Gestaltungselement virtueller Unternehmen", Nr. 12/1997.* 1997

[Feeney 2002] Feeney, Allison B.: The STEP Modular Architecture. In: *Journal of Computing and Information Science in Engineering* (2002), Nr. 2, S. 132–135

[Fowler 1995] Fowler, Julian: *STEP for data management, exchange and sharing.* Twickenham, UK : Technology Appraisals, 1995. – ISBN 1–871802–36–9

[Franke et al. 2009] Franke, Jörg ; Goth, Christian ; Fischer, Christian ; Pfeffer, Michael: Effiziente rechnergestützte Produktentwicklung für räumliche elektronische Baugruppen. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* (2009), Nr. 11, S. 925–930

[Friedrichsen u. Henning 2006] Friedrichsen, Uwe ; Henning, Dirk: Das "Google Web Toolkit". In: *OBJEKTspektrum* (2006), Nr. 06, S. 52–56

[Grabowski u. Anderl 1990] Grabowski, Hans ; Anderl, Reiner: *Produktdatenaustausch und CAD-Normteile*. Ehningen bei Böblingen : expert-Verlag, 1990. – ISBN 3–8169–0606–0

[Grabowski et al. 1993] Grabowski, Hans ; Anderl, Reiner ; Polly, Adam: *Integriertes Produktmodell*. 1. Aufl. Berlin : Beuth Verlag GmbH, 1993. – ISBN 3–410–12920–0

[Gümbel 2006] Gümbel, Christoph: Wie digitale Prototypen durch Virtual Reality lebendig gemacht werden. In: Schenk, Michael (Hrsg.): *Virtual Reality - Mensch und Maschine im interaktiven Dialog: Gastvortragsreihe 2006* Bd. 3. Magdeburg : Fraunhofer Institut für Fabrikbetrieb und -automatisierung IFF, 2006. – ISBN 3–8167–7256–0, S. 11–63

[Han et al. 2002] Han, S. ; Choi, Y. ; Yoo, S. ; Park, N.: Collaborative engineering design based on an intelligent STEP database. In: *Concurrent Engineering* (2002), Nr. 3, S. 239–249. – ISSN 1063–293X

[Holmdahl 2007] Holmdahl, Lars: *Integrierte Produktentwicklung*. Bd. 11: *Complexity Aspects of Product Development: Otto-von-Guericke-Universität, Fakultät für Maschinenbau, Dissertation*. Magdeburg : Otto-von-Guericke-Universität Magdeburg, 2007. – ISBN 978–3–9807688–9–4

[Isermann 2008] Isermann, Rolf: *Mechatronische Systeme: Grundlagen*. 2., vollst. neu bearb. Aufl. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2008. – ISBN 978–3–540–32336–5

[Juhász 2008] Juhász, Tamás: *Advanced Solutions in Object–Oriented Mechatronic Simulation*. Budapest, Budapest University of Technology and Economics, Diss., 2008

[Juhász u. Schmucker 2008] Juhász, Tamás ; Schmucker, Ulrich: Automatic Model Conversion to Modelica for Dymola-based Mechatronic Simulation. In: Bachmann, Bernhard (Hrsg.): *Modelica 2008: Proceedings of the 6th International Modelica Conference; 3. und 4. März, Bielefeld*, 2008, S. 719–726

[Kern u. Bøhn 1997] Kern, Vincius M. ; Bøhn, Jan H.: STEP Databases for Product Data Exchange. In: *Proceedings of the 1st International Congress of Industrial Engineering, 4.-7. September, São Carlos, Brasilien*. 1997, S. 1337–1341

[Kimura 2002] Kimura, Ikuo: *Fortschritte in der Maschinenkonstruktion*. Bd. 7: *Product Development with Mathematical Modeling, Rapid Prototyping, and Virtual Prototyping: Otto-von-Guericke-Universität, Fakultät für Maschinenbau, Dissertation*. Aachen : Shaker, 2002. – ISBN 3–8322–0896–8

[Kramer u. Xu 2009] Kramer, Thomas ; Xu, Xun: STEP in a Nutshell. In: Xu, Xun (Hrsg.) ; Nee, Andrew Y. C. (Hrsg.): *Advanced Design and Manufacturing Based on STEP*. London : Springer-Verlag London, 2009. – ISBN 978–1–84882–738–7, S. 1–22

[Krause et al. 2002] Krause, Frank-Lothar (Hrsg.) ; Tang, Trac (Hrsg.) ; Ahle, Ulrich (Hrsg.): *Leitprojekt integrierte Virtuelle Produktentstehung: Abschlussbericht*. Stuttgart : Fraunhofer IRB Verlag, 2002. – ISBN 3–8167–6165–8

[Krehmer et al. 2010] Krehmer, Hartmut ; Eckstein, Raiko ; Lauer, Wolfgang ; Roelofsen, Julia ; Stöber, Christina ; Troll, Alexander ; Zapf, Jochen ; Weber, Nadine ; Meerkamm, Harald ; Henrich, Andreas ; Lindemann, Udo ; Rieg, Frank ; Wartzack, Sandro: Das FORFLOW-Prozessmodell zur Unterstützung der multidisziplinären Produktentwicklung. In: *Konstruktion* (2010), Nr. 10, S. 59–68

[Lindemann 2007] Lindemann, Udo: *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden*. 2., bearbeitete Auflage. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2007. – ISBN 978–3–540–37435–0

[Lubell et al. 2004] Lubell, J. ; Peak, R. S. ; Srinivasan, V. ; Waterbury, S. C.: STEP, XML, and UML: Complementary Technologies. In: *DETC 2004: ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference 28. September - 02. Oktober, Salt Lake City, USA* (2004)

[Lüder et al. 2010] Lüder, A. ; Hundt, L. ; Keibel, A.: Description of manufacturing processes using AutomationML. In: *15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) 13.-16. September, Bilbao, Spanien* (2010)

[Meerkamm et al. 2009] Meerkamm, Harald (Hrsg.) ; Henrich, Andreas (Hrsg.) ; Jablonski, Stefan (Hrsg.) ; Krcmar, Helmut (Hrsg.) ; Lindemann, Udo (Hrsg.) ; Rieg, Frank (Hrsg.): *Flexible Prozessunterstützung in der Produktentwicklung: Prozesse - Daten - Navigation: Bayerischer Forschungsverbund FORFLOW für Prozess- und Workflowunterstützung zur Planung und Steuerung der Abläufe in der Produktentwicklung; Abschlussbericht 01.10.2006 - 30.09.2009*. Aachen : Shaker, 2009. – ISBN 978–3–8322–8640–8

[Nagel et al. 1980] Nagel, Roger N. ; Braithwaite, Walt W. ; Kennicott, Philip R.: *Initial Graphics Exchange Specification (IGES), Version 1.0, N.B.S. Report NBSIR 80-1978 (R)*. Januar 1980

[Oestereich 2009] Oestereich, Bernd: *Analyse und Design mit UML 2.3: Objektorientierte Softwareentwicklung*. 9., aktualisierte und erw. Aufl. München : Oldenbourg, 2009. – ISBN 978–3–486–58855–2

[Otter u. Elmqvist 2000] Otter, Martin ; Elmqvist, Hilding: Modelica - Language, Libraries, Tools, and Conferences. In: *Simulation News Europe* Dezember (2000), S. 3–8

[Otter u. Schweiger 2004] Otter, Martin ; Schweiger, Christian: Modellierung mechatronischer Systeme mit Modelica. In: Verein Deutscher Ingenieure (Hrsg.):

Mechatronischer Systementwurf: Methoden, Werkzeuge, Erfahrungen, Anwendungen; Tagung Darmstadt, 22. und 23. Juni 2004 (VDI-Berichte 1842). Düsseldorf : VDI-Verlag, 2004. – ISBN 3–18–091842–X, S. 39–50

[Pahl et al. 2007] Pahl, Gerhard ; Beitz, Wolfgang ; Feldhusen, Jörg ; Grote, Karl-Heinrich: *Pahl/Beitz Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung.* 7. Auflage. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2007. – ISBN 978–3–540–34060–7

[Paulson 2005] Paulson, Linda D.: Building Rich Web Applications with Ajax. In: *Computer* (2005), Nr. 10, S. 14–17. – ISSN 0018–9162

[Pratt 2001] Pratt, Michael J.: Introduction to ISO 10303 - the STEP Standard for Product Data Exchange. In: *Journal of Computing and Information Science in Engineering* (2001), Nr. 1, S. 102–103

[Ranger 2007] Ranger, Tony: *A comparison of the benefits of JT Open and STEP (Vortrag), UGS Connection - PLM World.* 2007

[Reichert u. Dadam 2000] Reichert, Manfred ; Dadam, Peter: Geschäftsprozessmodellierung und Workflow-Management - Konzepte, Systeme und deren Anwendung. In: *Industrie Management* (2000), Nr. 3, S. 23–27

[Reim et al. 2007] Reim, Reinhard ; Cordon, Peter ; Hund, Andreas ; Stjepandic, Josip: CAD-Konvertierung: Motivation, Probleme und Lösungen. In: *CAD-CAM Report* (2007), Nr. 9, S. 2–7

[Richert et al. 2004] Richert, F. ; Rückert, J. ; Schlosser, A. ; Abel, D.: Modellbildung und Simulation eines Dieselmotors unter Dymola und Matlab. In: Verein Deutscher Ingenieure (Hrsg.): *Steuerung und Regelung von Fahrzeugen und Motoren - AUTOREG 2004: Tagung Wiesloch, 2. und 3. März 2004 (VDI-Berichte 1828).* Düsseldorf : VDI-Verlag, 2004. – ISBN 3–18–091828–4, S. 331–341

[Richter 2009] Richter, Michael: Prozessmanagement in der Produktentwicklung. In: Bullinger, Hans-Jörg (Hrsg.) ; Spath, Dieter (Hrsg.) ; Warnecke, Hans-Jürgen (Hrsg.) ; Westkämper, Engelbert (Hrsg.): *Handbuch Unternehmensorganisation: Strategien, Planung, Umsetzung.* Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2009. – ISBN 978–3–540–72136–9, S. 431–448

[Roddeck 2006] Roddeck, Werner: *Einführung in die Mechatronik.* 3., überarb. und erg. Aufl. Wiesbaden, 2006. – ISBN 3–8351–0071–8

[Saake et al. 2008] Saake, Gunter ; Sattler, Kai-Uwe ; Heuer, Andreas: *Datenbanken: Konzepte und Sprachen.* 3., aktualisierte und erw. Aufl. Heidelberg : mitp, 2008. – ISBN 978–3–8266–1664–8

[Schäfer u. Taboada 2010] Schäfer, Steffen ; Taboada, Papick G.: Das Google Web Toolkit. In: *JavaSPEKTRUM* (2010), Nr. 03, S. 28–31

[Schenk u. Schmucker 2009] Schenk, Michael ; Schmucker, Ulrich: Durchgängiges Virtual Engineering für Maschinen und Anlagen. In: *Industrie Management* (2009), Nr. 1, S. 53–56

[Schenk u. Straßburger 2005] Schenk, Michael ; Strassburger, Steffen: Virtual Engineering für die Automobilindustrie. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* (2005), Nr. 9, S. 507–509

[Schlund et al. 2009] Schlund, Sebastian ; Riekhof, Florian ; Winzer, Petra: Probleme bei der Entwicklung mechatronischer Systeme - Ergebnisse einer Industriebefragung. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* (2009), Nr. 1-2, S. 54–59

[Schuh u. Baessler 2009] Schuh, Günter ; Baessler, Elke: Lebenszyklusorientierte Produktentwicklung. In: *[Bullinger et al., 2009].* 2009, S. 236–246

[Schuhmann 2001] Schuhmann, Sören: *Methoden zur Optimierung des Austausches von Geometrie- und Parametrikdaten.* Magdeburg, Otto-von-Guericke-Universität Magdeburg, Diss., 2001

[Sendler 2009] Sendler, Ulrich: Entscheidend ist die Qualität der Daten - JT, führendes Format für die Verbreitung und Visualisierung von CAD-Daten, wird zum ISO-Standard. In: *interface* (2009), Nr. 02, S. 10–11

[Shabana 2005] Shabana, Ahmed A.: *Dynamics of multibody systems.* 3. Aufl. New York, NY : Cambridge University Press, 2005. – ISBN 978–0–521–85011–7

[Spur u. Krause 1997] Spur, Günter ; Krause, Frank-Lothar: *Das virtuelle Produkt: Management der CAD-Technik.* München : Hanser, 1997. – ISBN 3–446–19176–3

[Stark et al. 2011] Stark, R. ; Hayka, H. ; Israel, J. H. ; Kim, M. ; Müller, P. ; Völlinger, U.: Virtuelle Produktentstehung in der Automobilindustrie. In: *Informatik-Spektrum* (2011), Nr. 1, S. 20–28

[Stekolschik 2007] Stekolschik, Alexander: *Ein Beitrag zum ganzheitlichen Qualitätsmanagement von CAD-Modellen in der Produktentstehung.* Bochum, Ruhr-Universität Bochum, Diss., 2007

[Swienczek et al. 1998] Swienczek, B. ; Arnold, F. ; Kilb, T. ; Janocha, A. T. ; Sartiono, R.: Online-Kopplung von CAx-Systemen für die virtuelle Produktentwicklung - ein Vergleich mit dem dateibasierten Datenaustausch. In: Verein Deutscher Ingenieure (Hrsg.): *Prozessketten für die virtuelle Produktentwicklung in verteilter Umgebung: Tagung München, 20. und 21. Oktober 1998 (VDI-Berichte 1435).* Düsseldorf : VDI-Verlag, 1998. – ISBN 3–18–091435–1, S. 219–238

[Tenbusch 2002] Tenbusch, Alexander: *Fortschritte in der Maschinenkonstruktion.* Bd. 5: *Rechnergestützte geometriedaten-bestimmte Prozeßketten zur Produktentwicklung: Otto-von-Guericke-Universität, Fakultät für Maschinenbau, Dissertation.* Aachen : Shaker, 2002. – ISBN 3–8322–0801–1

[Vajna et al. 2009] Vajna, Sándor ; Weber, Christian ; Bley, Helmut ; Zeman, Klaus ; Hehenberger, Peter: *CAx für Ingenieure: Eine praxisbezogene Einführung.* 2., völlig neu bearb. Aufl. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2009. – ISBN 978–3–540–36038–4

[Vettermann 2010] Vettermann, Steven: *It's all about Interoperability - ProSTEP iViP Standardization Activities: JT & STEP (Vortrag), JT Open International Conference, 12.-14. September.* 2010

[Vornholt et al. 2010] Vornholt, Stephan ; Geist, Ingolf ; Li, Yuexiao: Categorisation of Data Management Solutions for Heterogeneous Data in Collaborative Virtual Engineering. In: Saake, Gunter (Hrsg.) ; Köppen, Veit (Hrsg.): *IWDE 2010: proceedings of the 1st International Workshop on Digital Engineering; Otto-von-Guericke-Universität Magdeburg*. Magdeburg : Universität Magdeburg, 2010, S. 9–16

[Warschat 2009] Warschat, Joachim: Virtual Engineering. In: *[Bullinger et al., 2009]*. 2009, S. 530–544

[Westkämper u. Niemann 2009] Westkämper, Engelbert ; Niemann, Jörg: Digitale Produktion - Herausforderung und Nutzen. In: *[Bullinger et al., 2009]*. 2009, S. 515–530

[Zorriassatine et al. 2003] Zorriassatine, F. ; Wykes, C. ; Parkin, R. ; Gindy, N.: A survey of virtual prototyping techniques for mechanical product development. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* (2003), Nr. 4, S. 513–530

Richtlinien und Normen

[DIN 6789-1 1990] DIN 6789-1: *Dokumentationssystematik - Teil 1: Aufbau Technischer Produktdokumentationen*. Berlin : Beuth Verlag GmbH, September 1990

[DIN ISO 15226 1999] DIN ISO 15226: *Technische Produktdokumentation - Lebenszyklusmodell und Zuordnung von Dokumenten*. Berlin : Beuth Verlag GmbH, Oktober 1999

[ISO 10303-1 1994] ISO 10303-1: *Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles*. 1994

[ISO 10303-11 2004] ISO 10303-11: *Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual*. 2004

[ISO 10303-203 2005] ISO 10303-203: *Industrial automation systems and integration - Product data representation and exchange - Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies (modular version)*. 2005

[ISO 10303-209 2001] ISO 10303-209: *Industrial automation systems and integration - Product data representation and exchange - Part 209: Application protocol: Composite and metallic structural analysis and related design*. 2001

[ISO 10303-21 2002] ISO 10303-21: *Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure*. 2002

[ISO 10303-214 2010] ISO 10303-214: *Industrial automation systems and integration - Product data representation and exchange - Part 214: Application protocol: Core data for automotive mechanical design processes*. 2010

[ISO 10303-22 1998] ISO 10303-22: *Industrial automation systems and integration - Product data representation and exchange - Part 22: Implementation methods: Standard data access interface.* 1998

[ISO 10303-25 2005] ISO 10303-25: *Industrial automation systems and integration - Product data representation and exchange - Part 25: Implementation methods: EXPRESS to XMI binding.* 2005

[ISO 10303-28 2007] ISO 10303-28: *Industrial automation systems and integration - Product data representation and exchange - Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas.* 2007

[ISO 14772-1 1997] ISO 14772-1: *Information technology - Computer graphics and image processing - The Virtual Reality Modeling Language - Part 1: Functional specification and UTF-8 encoding.* 1997

[ISO 19775-1 2004] ISO 19775-1: *Information technology - Computer graphics and image processing - Extensible 3D (X3D) - Part 1: Architecture and base components.* 2004

[ISO/PAS 14306 2009] ISO/PAS 14306: *JT File Format Specification harvesting for 3D Visualization of industrial data.* 2009

[VDI 2206 2004] VDI 2206: *Entwicklungsmethodik für mechatronische Systeme.* Berlin : Beuth Verlag GmbH, Juni 2004

[VDI 2209 2009] VDI 2209: *3-D-Produktmodellierung - Technische und organisatorische Voraussetzungen - Verfahren, Werkzeuge und Anwendungen - Wirtschaftlicher Einsatz in der Praxis.* Berlin : Beuth Verlag GmbH, März 2009

[VDI 2211 2003] VDI 2211: *Blatt 2: Informationsverarbeitung in der Produktentwicklung - Berechnungen in der Konstruktion.* Berlin : Beuth Verlag GmbH, März 2003

[VDI 2219 1993] VDI 2219: *Informationsverarbeitung in der Produktentwicklung - Einführung und Wirtschaftlichkeit von EDM/PDM-Systemen.* Berlin : Beuth Verlag GmbH, Mai 1993

[VDI 2221 1993] VDI 2221: *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte.* Berlin : Beuth Verlag GmbH, Mai 1993

[VDI 4449 2008] VDI 4449: *Blatt 1: Digitale Fabrik - Grundlagen.* Berlin : Beuth Verlag GmbH, Februar 2008

Anhang

A Anwendungsfallbeschreibungen

In diesem Anhang befinden sich Anwendungsfallbeschreibungen zu den Anwendungsfällen, die bei der Erläuterung des Konzeptes in Abbildung 4.3 (auf Seite 57) und Abbildung 4.12 (auf Seite 71) dargestellt sind.

Tabelle A.1: Anwendungsfallbeschreibungen Teilkonzept »Systemverwaltung«

Anwendungsfall	Beschreibung
CAx-System anlegen	Ein Anwender möchte ein CAx-System in der Systemverwaltung anlegen. Dafür müssen die unveränderlichen Eigenschaften (Bezeichnung des CAx-Systems, Hersteller, Beschreibung) eingegeben sowie eine erste Version des CAx-Systems anlegt und beschrieben werden (mit dem eingebundenen AWF »CAx-System_Version anlegen«).
CAx-System_Version anlegen	Ein Anwender möchte eine Version eines CAx-Systems anlegen. Eine CAx-System_Version wird entweder als erste Version im Zusammenhang mit dem Anlegen eines CAx-Systems angelegt oder wenn ein CAx-System aktualisiert wurde und damit in einer neuen Version eingesetzt wird. Der Anwender muss die Version, eine Beschreibung und das Datum eingeben, ab dem die CAx-System_Version gültig ist.
Dateiformat anlegen	Ein Anwender möchte ein Dateiformat anlegen. Zum Anlegen sind einige Eigenschaften des Dateiformates erforderlich (Bezeichnung, Version, Dateiendungen, Format, Beschreibung) und es kann gegebenenfalls eine Kurzbezeichnung eingegeben werden.
Export- oder Importmöglichkeit einer CAx-System_Version definieren	Ein Anwender möchte zu einer CAx-System_Version eine Export- oder Importmöglichkeit festlegen. Nach der Auswahl der CAx-System_Version kann entweder ein bereits angelegtes Dateiformat gewählt oder ein neues angelegt werden (mit dem eingebundenen AWF »Dateiformat anlegen«). Der Anwender muss festlegen, ob es sich um eine Export- oder Importmöglichkeit handelt. Falls beim Export oder Import Einstellungen möglich sind, sollen diese definiert werden. Für Einstellungen, bei denen aus verschiedenen Möglichkeiten eine ausgewählt werden kann, sollen die entsprechenden Auswahlmöglichkeiten festgelegt werden.

Schnittstelle definieren	Ein Anwender möchte eine Schnittstelle zwischen zwei CAx-System_Versionen definieren. Dafür müssen die beiden CAx-System_Versionen bereits angelegt sein. Eine Version wird als Sender und die andere als Empfänger bestimmt. Zur Unterstützung sollen dem Anwender anschließend die Dateiformate angezeigt werden, die zum Austausch zwischen den ausgewählten CAx-System_Versionen verwendet werden können, das heißt die Dateiformate, welche der Sender exportieren und der Empfänger gleichzeitig importieren kann. Der Anwender wählt die Dateiformate aus, die im Zusammenhang mit der Schnittstelle ausgetauscht werden sollen. Für jedes Dateiformat muss ein Austauschvorgang anlegt werden, für den eine Beschreibung und gegebenenfalls die Dateianzahl eingegeben werden muss. Zum Abschluss muss die Schnittstelle noch beschrieben werden.
CAx-System_Version deaktivieren	Ein Anwender möchte eine CAx-System_Version deaktivieren, weil sie nicht mehr verwendet werden kann. Dafür muss die entsprechende CAx-System_Version ausgewählt und anschließend ein Datum, ab dem die Version nicht mehr aktiv ist, eingegeben werden.
Systemlandschaft anzeigen	Ein Anwender möchte sich die Systemlandschaft zu einem gewünschten Zeitpunkt anzeigen lassen. Dafür kann entweder ein beliebiger Zeitpunkt eingegeben oder die aktuelle Systemlandschaft ausgewählt werden.
Export- und Importmöglichkeit einer CAx-System_Version anzeigen	Ein Anwender möchte sich zu einer ausgewählten CAx-System_Version alle Export- und Importmöglichkeiten anzeigen lassen.
Schnittstellen anzeigen	Ein Anwender möchte sich über Schnittstellen informieren. Entweder kann er sich alle Schnittstellen zu einer ausgewählten CAx-System_Version oder die Schnittstellen, welche zwischen zwei ausgewählten CAx-System_Versionen möglich sind, anzeigen lassen.

Tabelle A.2: Anwendungsfallbeschreibungen Teilkonzept »Prozessverwaltung«

Anwendungsfall	Beschreibung
Prozesskette anlegen	Ein Anwender möchte eine Prozesskette anlegen, welche einen gewünschten Ablauf beschreibt. Dafür muss er zunächst eine Bezeichnung und eine Beschreibung der Prozesskette eingeben. Anschließend kann die Prozesskette durch mehrere Prozessschritte und gegebenenfalls Verbindungen definiert werden.
Prozessschritt anlegen	Dieser AWF ist in den AWF »Prozesskette anlegen« eingebunden. Ein Anwender kann damit einen Prozessschritt anlegen und beschreiben. Es muss eine CAx-System_Version ausgewählt werden, welche zur Durchführung des Prozessschritts verwendet werden soll. Außerdem muss bei einem Prozessschritt, welcher nicht der erste Prozessschritt der Prozesskette ist, ein Vorgänger definiert werden. Dies ist entweder ein Prozessschritt oder eine Verbindung. Anschließend müssen noch eine oder mehrere Importalternativen angelegt werden.
durchgeführte Prozesskette beschreiben	Ein Anwender möchte eine durchgeführte Prozesskette beschreiben. Dafür muss jeweils eine Instanz der Prozesskette und der durchgeführten Prozessschritte angelegt werden. Anschließend werden die Prozessschritte einzeln beschrieben, wobei alle durchgeführten Austauschvorgänge beschrieben werden müssen (mit dem eingebundenen AWF »durchgeführten Austauschvorgang beschreiben«).
durchgeführten Austauschvorgang beschreiben	Dieser AWF ist in den AWF »durchgeführte Prozesskette beschreiben« eingebunden. Zur Beschreibung eines Austauschvorgangs müssen die dabei durchgeführten Einstellungen beim Export und Import abgelegt werden. Außerdem wird die konkrete Instanz des Austauschvorgangs mit den erzeugten Dateien in Verbindung gesetzt.
angelegte Prozesskette anzeigen	Ein Anwender möchte sich über eine angelegte Prozesskette informieren und alle definierten Eigenschaften der Prozesskette anzeigen lassen.
durchgeführte Prozesskette anzeigen	Ein Anwender möchte alle abgelegten Informationen über eine durchgeführte Prozesskette nachschlagen.

B Weitere Aktivitätsdiagramme des Konzeptes

Die in diesem Anhang dargestellten Aktivitätsdiagramme wurden zur Untersetzung von Anwendungsfällen des Konzeptes erstellt. Sie wurden in Kapitel 4 nicht beschrieben, weil sie zum einen nicht komplex sind und zum anderen aus Platzgründen. Dieser Anhang wurde in zwei Abschnitte unterteilt. Der Abschnitt B.1 umfasst die Aktivitätsdiagramme zum Teilkonzept »Systemverwaltung«, welches in Abschnitt 4.1 beschrieben wird. Der Abschnitt B.2 umfasst die Aktivitätsdiagramme zum Teilkonzept »Prozessverwaltung«, welches in Abschnitt 4.2 beschrieben wird.

B.1 Aktivitätsdiagramme zum Teilkonzept »Systemverwaltung«

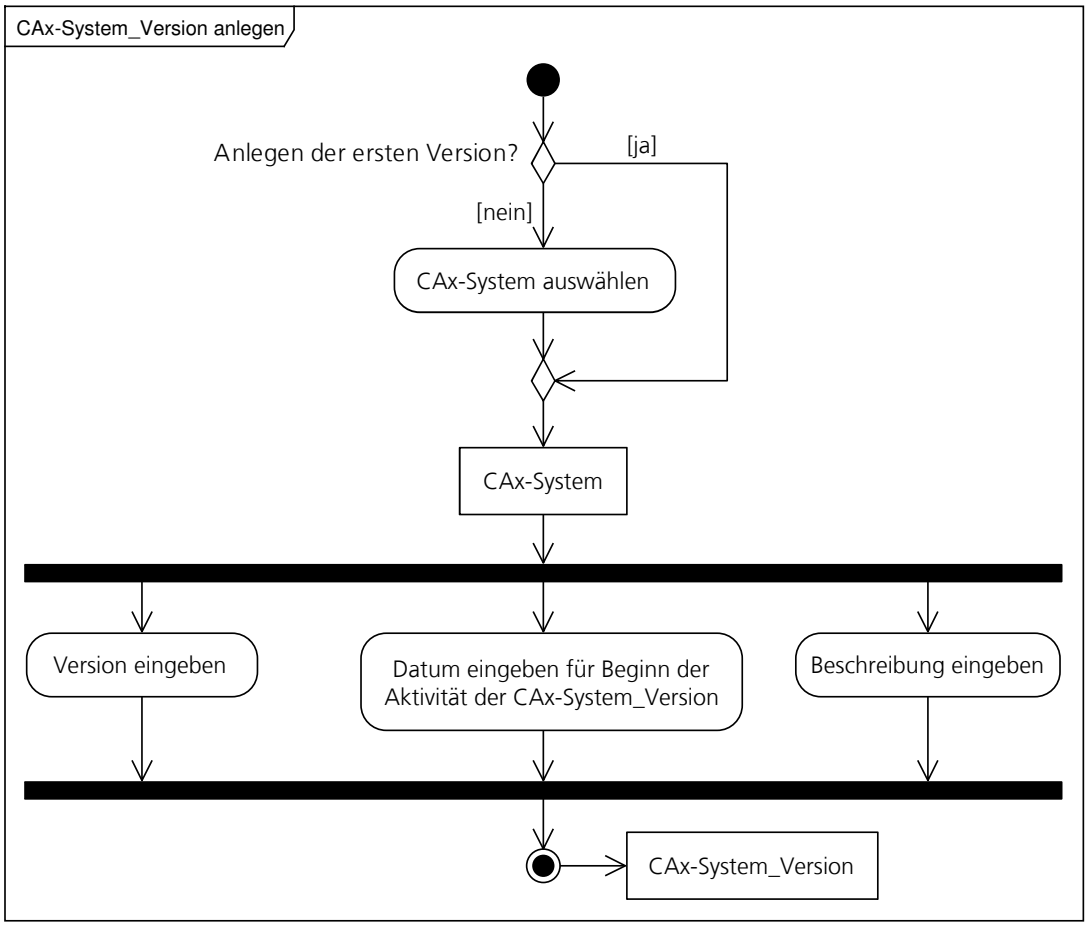


Abbildung B.1: Aktivitätsdiagramm zum Anwendungsfall »CAx-System_Version anlegen«

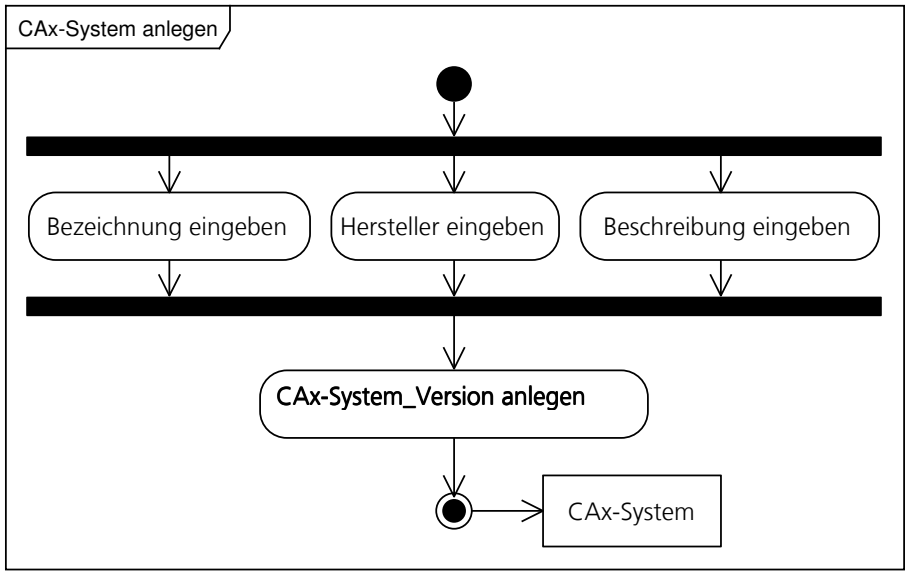


Abbildung B.2: Aktivitätsdiagramm zum Anwendungsfall »CAx-System anlegen«

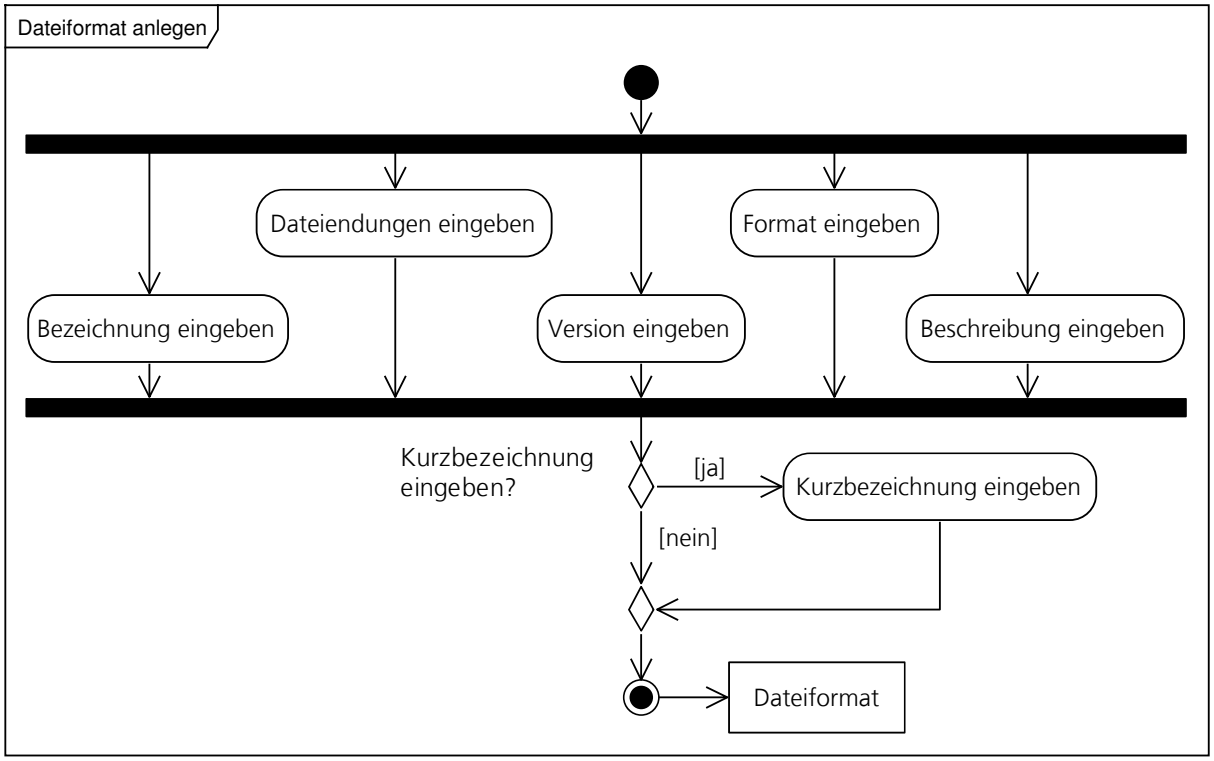


Abbildung B.3: Aktivitätsdiagramm zum Anwendungsfall »Dateiformat anlegen«

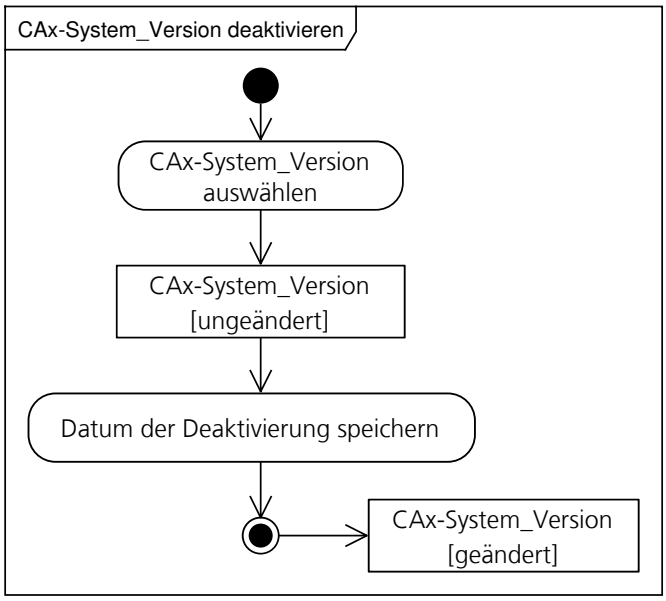


Abbildung B.4: Aktivitätsdiagramm zum Anwendungsfall »CAx-System_Version deaktivieren«

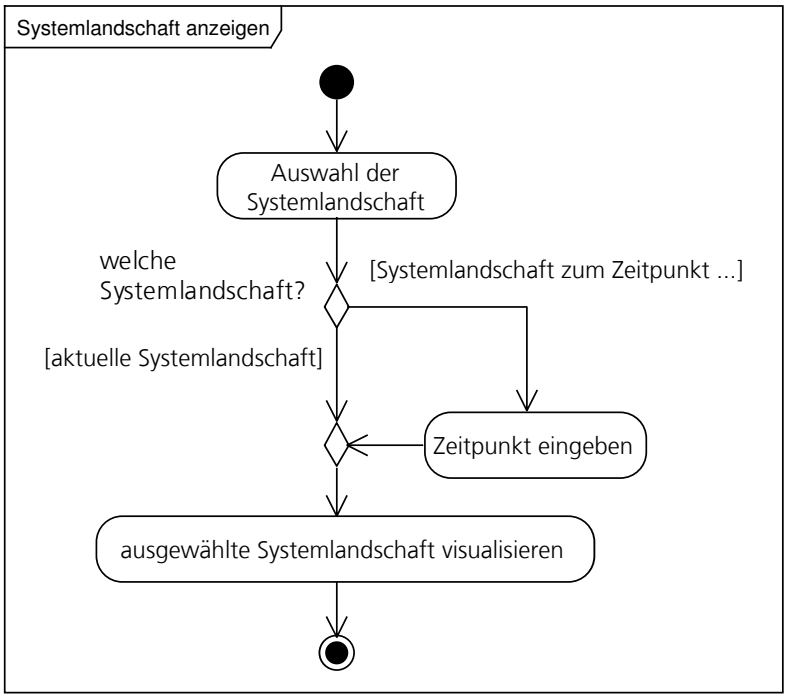


Abbildung B.5: Aktivitätsdiagramm zum Anwendungsfall »Systemlandschaft anzeigen«

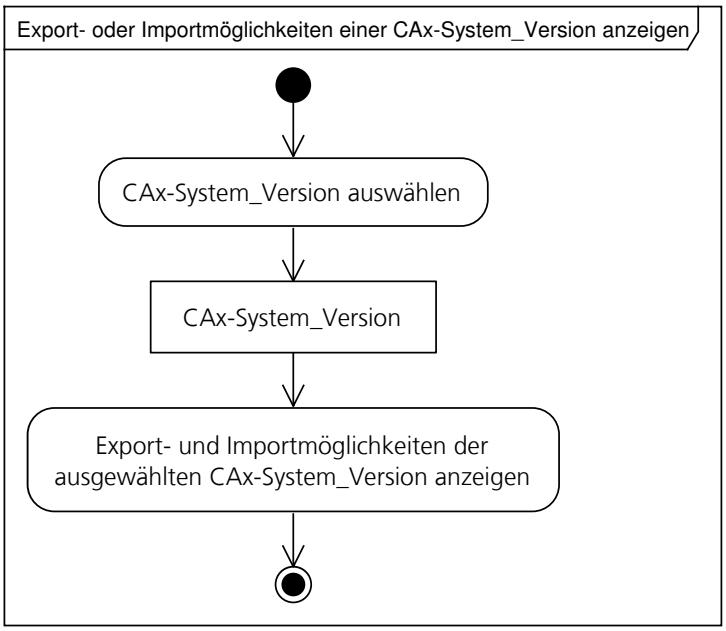


Abbildung B.6: Aktivitätsdiagramm zum Anwendungsfall »Export- oder Importmöglichkeiten einer CAx-System_Version anzeigen«

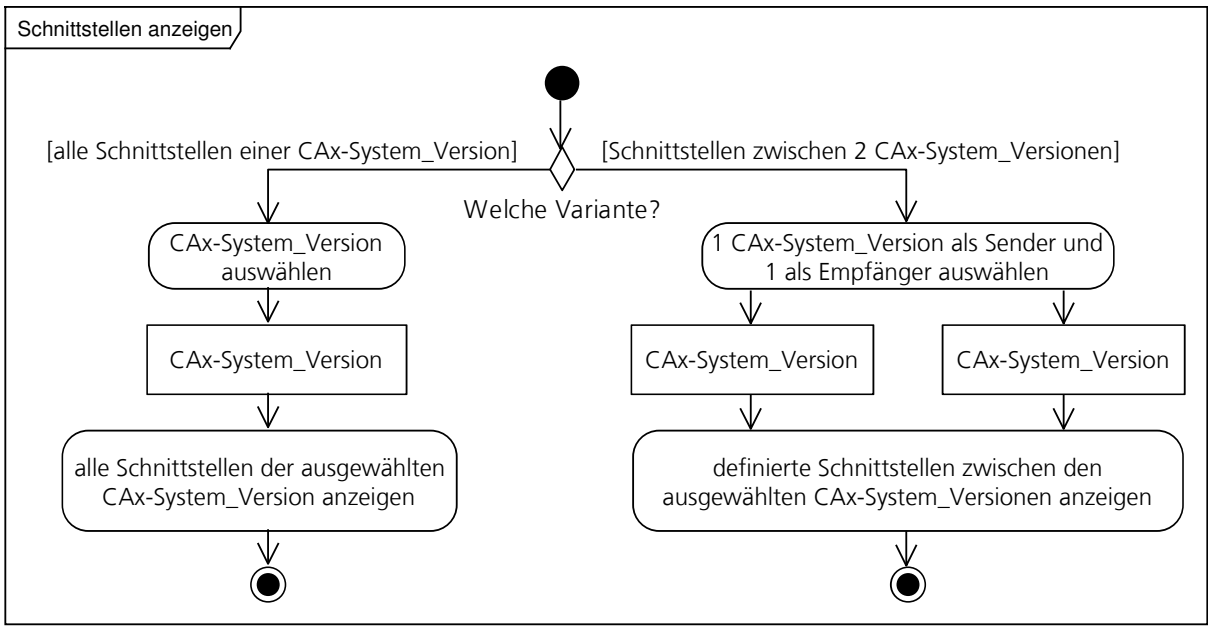


Abbildung B.7: Aktivitätsdiagramm zum Anwendungsfall »Schnittstellen anzeigen«

B.2 Aktivitätsdiagramme zum Teilkonzept »Prozessverwaltung«

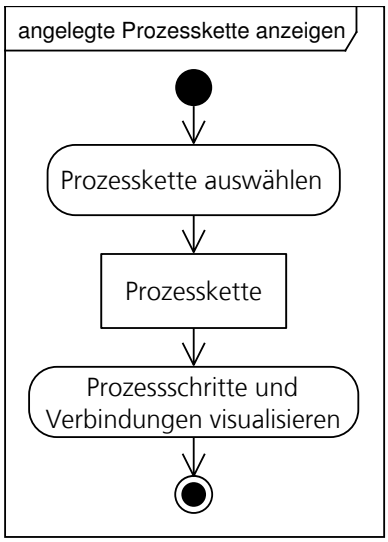


Abbildung B.8: Aktivitätsdiagramm zum Anwendungsfall »angelegte Prozesskette anzeigen«

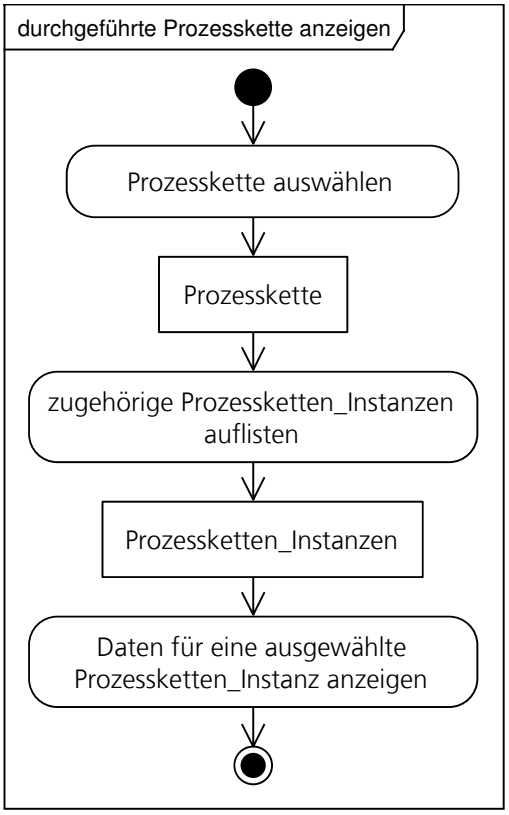


Abbildung B.9: Aktivitätsdiagramm zum Anwendungsfall »durchgeführte Prozesskette anzeigen«

C SQL-Skript

In diesem Anhang ist das SQL-Skript dargestellt, welches zur Implementierung einer Datenbank bei der prototypischen Umsetzung des Konzeptes erstellt wurde. Das Skript dient zum Anlegen von Tabellen im DBMS »PostgreSQL 8.4«.

```
-----------------------------------------
--   Tabelle caxsystems
-----------------------------------------

CREATE TABLE vemetadb.caxsystems
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    producer VARCHAR(100) NOT NULL,
    description TEXT NOT NULL
);

-----------------------------------------
--   Tabelle caxsystem_versions
-----------------------------------------

CREATE TABLE vemetadb.caxsystem_versions
(
    id SERIAL PRIMARY KEY,
    version VARCHAR(50) NOT NULL,
    active BOOLEAN NOT NULL,
    activation_date DATE NOT NULL,
    deactivation_date DATE,
    description TEXT NOT NULL,
    caxsystem_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.caxsystem_versions ADD CONSTRAINT caxsystem_versions_fk1 FOREIGN KEY
     (caxsystem_id) REFERENCES vemetadb.caxsystems(id) ON UPDATE No action ON DELETE No
    action;

-----------------------------------------
--   Tabelle file_formats
-----------------------------------------

CREATE TABLE vemetadb.file_formats
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    shortname VARCHAR(20)
);

-----------------------------------------
--   Tabelle file_extensions
-----------------------------------------

CREATE TABLE vemetadb.file_extensions
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(20) NOT NULL,
    file_format_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.file_extensions ADD CONSTRAINT file_extensions_fk1 FOREIGN KEY (
    file_format_id) REFERENCES vemetadb.file_formats(id) ON UPDATE No action ON DELETE No
     action;

-----------------------------------------
--   Tabelle file_format_versions
-----------------------------------------
```

```
CREATE TABLE vemetadb.file_format_versions
(
    id SERIAL PRIMARY KEY,
    version VARCHAR(50) NOT NULL,
    format VARCHAR(50) NOT NULL CHECK(format in ('ASCII', 'XML', 'Binär')),
    description TEXT NOT NULL,
    file_format_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.file_format_versions ADD CONSTRAINT file_format_versions_fk1 FOREIGN
     KEY (file_format_id) REFERENCES vemetadb.file_formats(id) ON UPDATE No action ON
    DELETE No action;

-----------------------------------------
--   Tabelle setting_possibilities
-----------------------------------------

CREATE TABLE vemetadb.setting_possibilities
(
    id SERIAL PRIMARY KEY,
    export BOOLEAN NOT NULL,
    caxsystem_version_id INTEGER NOT NULL,
    file_format_version_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.setting_possibilities ADD CONSTRAINT setting_possibilities_fk1
    FOREIGN KEY (caxsystem_version_id) REFERENCES vemetadb.caxsystem_versions(id) ON
    UPDATE No action ON DELETE No action;
ALTER TABLE vemetadb.setting_possibilities ADD CONSTRAINT setting_possibilities_fk2
    FOREIGN KEY (file_format_version_id) REFERENCES vemetadb.file_format_versions(id) ON
    UPDATE No action ON DELETE No action;

-----------------------------------------
--   Tabelle settings
-----------------------------------------

CREATE TABLE vemetadb.settings
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    setting_possibilities_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.settings ADD CONSTRAINT settings_fk1 FOREIGN KEY (
    setting_possibilities_id) REFERENCES vemetadb.setting_possibilities(id) ON UPDATE No
    action ON DELETE No action;

-----------------------------------------
--   Tabelle choices
-----------------------------------------

CREATE TABLE vemetadb.choices
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    setting_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.choices ADD CONSTRAINT choices_fk1 FOREIGN KEY (setting_id)
    REFERENCES vemetadb.settings(id) ON UPDATE No action ON DELETE No action;

-----------------------------------------
--   Tabelle interfaces
-----------------------------------------

CREATE TABLE vemetadb.interfaces
(
    id SERIAL PRIMARY KEY,
```

```
    description TEXT NOT NULL,
    caxsystem_version_id_transmitter INTEGER NOT NULL,
    caxsystem_version_id_receiver INTEGER NOT NULL
);

ALTER TABLE vemetadb.interfaces ADD CONSTRAINT interfaces_fk1 FOREIGN KEY (
    caxsystem_version_id_transmitter) REFERENCES vemetadb.caxsystem_versions(id) ON
    UPDATE No action ON DELETE No action;
ALTER TABLE vemetadb.interfaces ADD CONSTRAINT interfaces_fk2 FOREIGN KEY (
    caxsystem_version_id_receiver) REFERENCES vemetadb.caxsystem_versions(id) ON UPDATE
    No action ON DELETE No action;

-----------------------------------------
--   Tabelle exchanges
-----------------------------------------

CREATE TABLE vemetadb.exchanges
(
    id SERIAL PRIMARY KEY,
    numberoffiles INTEGER CHECK (numberoffiles > 0),
    description TEXT NOT NULL,
    interface_id INTEGER NOT NULL,
    file_format_version_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.exchanges ADD CONSTRAINT exchanges_fk1 FOREIGN KEY (interface_id)
    REFERENCES vemetadb.interfaces(id) ON UPDATE No action ON DELETE No action;
ALTER TABLE vemetadb.exchanges ADD CONSTRAINT exchanges_fk2 FOREIGN KEY (
    file_format_version_id) REFERENCES vemetadb.file_format_versions(id) ON UPDATE No
    action ON DELETE No action;

-----------------------------------------
--   Tabelle process_chains
-----------------------------------------

CREATE TABLE vemetadb.process_chains
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT NOT NULL
);

-----------------------------------------
--   Tabelle process_chain_instances
-----------------------------------------

CREATE TABLE vemetadb.process_chain_instances
(
    id SERIAL PRIMARY KEY,
    number INTEGER NOT NULL,
    description TEXT NOT NULL,
    process_chain_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.process_chain_instances ADD CONSTRAINT process_chain_instances_fk1
    FOREIGN KEY (process_chain_id) REFERENCES vemetadb.process_chains(id) ON UPDATE No
    action ON DELETE No action;


-----------------------------------------
--   Tabelle connections
-----------------------------------------

CREATE TABLE vemetadb.connections
(
    id SERIAL PRIMARY KEY,
    type_is_or BOOLEAN NOT NULL,
    type_is_split BOOLEAN NOT NULL,
    process_step_single_id INTEGER NOT NULL
);
```

```
-----------------------------------------
--   Tabelle process_steps
-----------------------------------------

CREATE TABLE vemetadb.process_steps
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    process_chain_id INTEGER NOT NULL,
    previous_process_step_id INTEGER,
    previous_connection_id INTEGER,
    caxsystem_version_id INTEGER NOT NULL,

    -- Ein Prozessschritt kann keinen oder einen Vorgänger haben, aber nicht zwei. Dies
        soll mit CHECK überprüft werden.
    CHECK( (previous_connection_id >0 AND NOT previous_process_step_id >0) OR (NOT
        previous_connection_id  >0 AND previous_process_step_id >0) OR (NOT
        previous_connection_id >0 AND NOT previous_process_step_id >0) )
);

-- Der Fremdschlüssel für Tabelle connections kann erst hier gesetzt werden, da Tabelle
    process_steps vorher definiert sein muss
ALTER TABLE vemetadb.connections ADD CONSTRAINT connections_fk1 FOREIGN KEY (
    process_step_single_id) REFERENCES vemetadb.process_steps(id) ON UPDATE No action ON
    DELETE No action;

ALTER TABLE vemetadb.process_steps ADD CONSTRAINT process_steps_fk1 FOREIGN KEY (
    process_chain_id) REFERENCES vemetadb.process_chains(id) ON UPDATE No action ON
    DELETE No action;
ALTER TABLE vemetadb.process_steps ADD CONSTRAINT process_steps_fk2 FOREIGN KEY (
    previous_process_step_id) REFERENCES vemetadb.process_steps(id) ON UPDATE No action
    ON DELETE No action;
ALTER TABLE vemetadb.process_steps ADD CONSTRAINT process_steps_fk3 FOREIGN KEY (
    previous_connection_id) REFERENCES vemetadb.connections(id) ON UPDATE No action ON
    DELETE No action;
ALTER TABLE vemetadb.process_steps ADD CONSTRAINT process_steps_fk4 FOREIGN KEY (
    caxsystem_version_id) REFERENCES vemetadb.caxsystem_versions(id) ON UPDATE No action
    ON DELETE No action;

-----------------------------------------
--   Tabelle connection_process_step
-----------------------------------------

CREATE TABLE vemetadb.connection_process_step
(
    connection_id INTEGER NOT NULL,
    process_step_id INTEGER NOT NULL,
    PRIMARY KEY(connection_id, process_step_id)
);

ALTER TABLE vemetadb.connection_process_step ADD CONSTRAINT connection_process_step_fk1
    FOREIGN KEY (connection_id) REFERENCES vemetadb.connections(id) ON UPDATE No action
    ON DELETE No action;
ALTER TABLE vemetadb.connection_process_step ADD CONSTRAINT connection_process_step_fk2
    FOREIGN KEY (process_step_id) REFERENCES vemetadb.process_steps(id) ON UPDATE No
    action ON DELETE No action;

-----------------------------------------
--   Tabelle import_options
-----------------------------------------

CREATE TABLE vemetadb.import_options
(
    id SERIAL PRIMARY KEY,
    number INTEGER NOT NULL,
    description TEXT NOT NULL,
    process_step_id INTEGER NOT NULL
);
```

```
ALTER TABLE vemetadb.import_options ADD CONSTRAINT import_options_fk1 FOREIGN KEY (
    process_step_id) REFERENCES vemetadb.process_steps(id) ON UPDATE No action ON DELETE
    No action;

-----------------------------------------
--   Tabelle interface_import_option
-----------------------------------------

CREATE TABLE vemetadb.interface_import_option
(
    interface_id INTEGER NOT NULL,
    import_option_id INTEGER NOT NULL,
    PRIMARY KEY(interface_id,import_option_id)
);

ALTER TABLE vemetadb.interface_import_option ADD CONSTRAINT interface_import_option_fk1
    FOREIGN KEY (interface_id) REFERENCES vemetadb.interfaces(id) ON UPDATE No action ON
    DELETE No action;
ALTER TABLE vemetadb.interface_import_option ADD CONSTRAINT interface_import_option_fk2
    FOREIGN KEY (import_option_id) REFERENCES vemetadb.import_options(id) ON UPDATE No
    action ON DELETE No action;

-----------------------------------------
--   Tabelle process_step_instances
-----------------------------------------

CREATE TABLE vemetadb.process_step_instances
(
    id SERIAL PRIMARY KEY,
    number INTEGER NOT NULL,
    process_step_id INTEGER NOT NULL,
    import_option_id INTEGER NOT NULL,
    process_chain_instance_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.process_step_instances ADD CONSTRAINT process_step_instances_fk1
    FOREIGN KEY (process_step_id) REFERENCES vemetadb.process_steps(id) ON UPDATE No
    action ON DELETE No action;
ALTER TABLE vemetadb.process_step_instances ADD CONSTRAINT process_step_instances_fk2
    FOREIGN KEY (import_option_id) REFERENCES vemetadb.import_options(id) ON UPDATE No
    action ON DELETE No action;
ALTER TABLE vemetadb.process_step_instances ADD CONSTRAINT process_step_instances_fk3
    FOREIGN KEY (process_chain_instance_id) REFERENCES vemetadb.process_chain_instances(
    id) ON UPDATE No action ON DELETE No
action;

-----------------------------------------
--   Tabelle setting_instances
-----------------------------------------

CREATE TABLE vemetadb.setting_instances
(
    id SERIAL PRIMARY KEY
);

-----------------------------------------
--   Tabelle exchange_instances
-----------------------------------------

CREATE TABLE vemetadb.exchange_instances
(
    id SERIAL PRIMARY KEY,
    number INTEGER NOT NULL,
    date DATE NOT NULL,
    exchange_id INTEGER NOT NULL,
    process_step_instance_id INTEGER NOT NULL,
    setting_instance_export_id INTEGER NOT NULL,
    setting_instance_import_id INTEGER NOT NULL
);
```

```
ALTER TABLE vemetadb.exchange_instances ADD CONSTRAINT exchange_instances_fk1 FOREIGN KEY
     (exchange_id) REFERENCES vemetadb.exchanges(id) ON UPDATE No action ON DELETE No
    action;
ALTER TABLE vemetadb.exchange_instances ADD CONSTRAINT exchange_instances_fk2 FOREIGN KEY
     (process_step_instance_id) REFERENCES vemetadb.process_step_instances(id) ON UPDATE
    No action ON DELETE No action;
ALTER TABLE vemetadb.exchange_instances ADD CONSTRAINT exchange_instances_fk3 FOREIGN KEY
     (setting_instance_export_id) REFERENCES vemetadb.setting_instances(id) ON UPDATE No
    action ON DELETE No action;
ALTER TABLE vemetadb.exchange_instances ADD CONSTRAINT exchange_instances_fk4 FOREIGN KEY
     (setting_instance_import_id) REFERENCES vemetadb.setting_instances(id) ON UPDATE No
    action ON DELETE No action;

-----------------------------------------
--   Tabelle setting_values
-----------------------------------------

CREATE TABLE vemetadb.setting_values
(
    id SERIAL PRIMARY KEY,
    vaule VARCHAR(50) NOT NULL,
    setting_instance_id INTEGER NOT NULL,
    setting_id INTEGER NOT NULL
);

ALTER TABLE vemetadb.setting_values ADD CONSTRAINT setting_values_fk1 FOREIGN KEY (
    setting_instance_id) REFERENCES vemetadb.setting_instances(id) ON UPDATE No action ON
     DELETE No action;
ALTER TABLE vemetadb.setting_values ADD CONSTRAINT setting_values_fk2 FOREIGN KEY (
    setting_id) REFERENCES vemetadb.settings(id) ON UPDATE No action ON DELETE No action;


-----------------------------------------
--   Tabelle file_file_format_version
-----------------------------------------

CREATE TABLE vemetadb.file_file_format_version
(
    file_id INTEGER PRIMARY KEY,
    usage_type VARCHAR(50) NOT NULL CHECK(usage_type in ('exchange','export_information',
        'import_information','screenshot')),
    setting_instance_id INTEGER NOT NULL,
    exchange_instance_id INTEGER NOT NULL,
    file_format_version_id INTEGER NOT NULL

    -- Eine Datei kann entweder zu einem Eintrag in der Tabelle Austauschvorgang_Instanz
        oder Einstellungen_Instanz zugeordnet werden, aber nicht zu beiden. Dies soll mit
         CHECK geprüft werden.
    CHECK( (exchange_instance_id >0 AND NOT setting_instance_id >0) OR (NOT
        exchange_instance_id  >0 AND setting_instance_id  >0) )
);

ALTER TABLE vemetadb.file_file_format_version ADD CONSTRAINT file_file_format_version_fk1
     FOREIGN KEY (setting_instance_id) REFERENCES vemetadb.setting_instances(id) ON
    UPDATE No action ON DELETE No action;
ALTER TABLE vemetadb.file_file_format_version ADD CONSTRAINT file_file_format_version_fk2
     FOREIGN KEY (exchange_instance_id) REFERENCES vemetadb.exchange_instances(id) ON
    UPDATE No action ON DELETE No action;
ALTER TABLE vemetadb.file_file_format_version ADD CONSTRAINT file_file_format_version_fk3
     FOREIGN KEY (file_format_version_id) REFERENCES vemetadb.file_format_versions(id) ON
     UPDATE No action ON DELETE No action;
ALTER TABLE vemetadb.file_file_format_version ADD CONSTRAINT file_file_format_version_fk4
     FOREIGN KEY (file_id) REFERENCES vemetadb.files(id) ON UPDATE No action ON DELETE No
     action;
```

Listing 6.1: SQL-Skript

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, den 11. April 2011

Michael Stoye