
EIT FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

# Verwendung von Merkmalen im Engineering von Systemen

**Dissertation**

zur Erlangung des akademischen Grades

Doktoringenieur
(Dr.-Ing.)

von: Dipl.-Ing. Thomas Hadlich
geboren am: 29.06.1967 in: Berlin

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke-Universität Magdeburg

Gutachter: Prof. Dr.-Ing. Christian Diedrich
Prof. Dr.-Ing. Alexander Fay

Promotionskolloquim am 09.04.2015

For something to exist, it has to be observed.

For something to exist, it has to have a position in time and space.

And this explains why nine-tenths of the mass of the universe is unaccounted for.

Nine-tenths of the universe is the knowledge of the position and direction of everything in the other tenth. Every atom has its biography, every star its file, every chemical exchange its equivalent of the inspector with a clipboard. It is unaccounted for because it is doing the accounting for the rest of it, and you cannot see the back of your own head.*

Nine-tenths of the universe, in fact, is the paperwork.

*Terry Prattchett* in *Thief of Time*

* Except in very **small** universes

# Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl Integrierte Automation der Otto-von-Guericke-Universität Magdeburg. Nur durch Unterstützung vieler Menschen wurde ihre Entstehung ermöglicht. Bei diesen Menschen möchte ich mich hier bedanken.

Mein besonderer Dank gilt Prof. Dr.-Ing. Christian Diedrich. Ohne seine Unterstützung wäre diese Arbeit nicht möglich gewesen und durch seine konstruktive Zusammenarbeit und anregenden Impulse hat er diese Arbeit wesentlich gefördert.

Ich danke Prof. Dr.-Ing. Alexander Fay, Helmut-Schmidt-Universität/Universität der Bundeswehr - Hamburg, für die Betreuung dieser Arbeit als Gutachter.

Einen wesentlichen Beitrag zu dieser Arbeit hat auch der IEC Arbeitskreis TC65 WG16 „Digitale Fabrik“ geleistet - insbesondere Herr Udo Döbrich, Siemens AG Karlsruhe. Mit seinem auf großem Erfahrungsschatz basierenden Rat und mit seiner Bereitschaft zur Diskussion hat er mir sehr geholfen.

Sehr herzlich danke ich auch meinen Kolleginnen und Kollegen am Institut für Automatisierungstechnik, vor allem Dr. Mathias Mühlhause und Stephan Höme. Durch ihre Gesprächsbereitschaft und angeregten Diskussionen sowie durch ihre Bereitschaft zum Korrekturlesen haben sie einen großen Teil zum Gelingen dieser Arbeit beigetragen. Es waren auch zahlreiche Studenten mit Studien-, Diplom-, Bachelor- und Master-Arbeiten an meiner Forschungstätigkeit beteiligt. – Vielen Dank!

Mein innigster Dank gilt meiner Frau und Gefährtin Kerstin Lerch. Nur durch ihre Geduld und Unterstützung war es überhaupt möglich dieses Vorhaben anzugehen.

Meinen Eltern, Hannelore und Klaus Hadlich.

# Inhaltsverzeichnis

Kurzfassung ..... xv
Abstract ..... xvii
1 Einführung ..... 1
2 Grundlagen und Begriffe ..... 5
2.1 Zielstellung ..... 5
2.2 Merkmalmodell ..... 5
2.3 Systemmodell ..... 8
2.3.1 Generelles Systemmodell ..... 8
2.3.1.1 System ..... 8
2.3.1.2 Struktur ..... 9
2.3.1.3 Funktion ..... 14
2.3.1.4 Verhalten ..... 16
2.3.1.5 Zusammenfassung ..... 18
2.3.2 Aspekte technischer Systeme ..... 19
2.3.2.1 Muster für Teilnehmer an einer Elementbeziehung ..... 19
2.3.2.2 Grad der Kopplung ..... 19
2.3.2.3 Schnittstelle ..... 21
2.3.2.4 Kardinalität ..... 24
2.3.2.5 Exklusivität ..... 24
2.3.2.6 Signal ..... 25
2.3.3 Verwendung verschiedener Systemmodelle ..... 26
2.3.4 Systeme von Systemen ..... 27
2.3.5 Eigenschaften von Systemen ..... 28
2.3.6 Komplexität ..... 30
2.3.6.1 Komplexität von Funktionen ..... 31
2.4 Zusammenfassung ..... 32
3 Stand der Wissenschaft und Technik ..... 33
3.1 Zielstellung ..... 33
3.2 Systemmodelle in Wissenschaft und Technik ..... 33
3.2.1 Systeme in der Automatisierungstechnik ..... 33
3.2.2 Funktionale Modelle ..... 34
3.2.3 Strukturmodelle ..... 37
3.2.3.1 Komponentenmodell ..... 37
3.2.3.2 Topologie-Modell und Netzwerkmodell ..... 38
3.2.4 Modell- und Elementbeziehungen ..... 39
3.2.5 Zusammenfassung ..... 40
3.3 CAEX Systembeschreibungssprache ..... 41
3.3.1 Einführung ..... 41
3.3.2 Darstellung der Aufbaustruktur ..... 45
3.3.3 Darstellung funktionaler Verbindungen ..... 46
3.3.4 Darstellung von Verhältnis-Beziehungen ..... 47
3.3.5 Darstellung von Verhaltensmodellen ..... 52
3.3.5.1 Modellierung von Verhalten mit CAEX-Elementen ..... 53
3.3.5.2 Modellierung von Verhalten mit PLCopen XML ..... 54
3.3.6 Integration mit Merkmalmodellen ..... 55
3.4 Merkmalmodelle in Wissenschaft und Technik ..... 56

3.4.1 Übersicht ... 56
3.4.2 Identifikation ... 57
3.4.3 ISO/IEC 11179 ... 58
3.4.4 IEC 61360 ... 59
3.4.5 DIN 4000 Serie ... 61
3.4.6 DIN 4002 Serie ... 61
3.4.7 DIN PAS 1040 Reihe ... 61
3.4.8 PROLIST ... 62
3.4.9 eCl@ss ... 64
3.4.10 Auf IEC 61360 basierende internationale Standards ... 65
3.4.10.1 Darstellung von Schnittstellen laut IEC 61987-10 ... 67
3.4.10.2 Angestrebte Verwendung von Properties und LOPs nach IEC 61360 ... 67
3.4.11 ISO 22745 ... 69
3.4.12 Vergleich von IEC 61360-basierten und ISO 22745-basierten Ansätzen ... 72
3.5 Unterstützung des Engineering-Prozess ... 73
3.5.1 Einordnung in den Anlagenlebenszyklus ... 74
3.5.2 Grundsätzliches Vorgehensmodell ... 76
3.5.3 Verfeinerung (Refinement) ... 77
3.5.4 CAEX-Engineering-Prozess ... 77
3.5.5 Entwicklungsprozess mit SysML ... 79
3.6 Zusammenfassung ... 82
4 Integration von Systemmodell und Merkmalmodell ... 85
4.1 Ziel der Integration ... 85
4.2 FAVA: V-Modell für Automatisierungssysteme ... 85
4.2.1 Im FAVA Vorgehensmodell verwendete Modelle der Anlage ... 88
4.2.2 Nutzung von Merkmalen in einem Entwicklungsschritt ... 90
4.2.3 Zusammenfassung ... 92
4.3 Beziehung zwischen Merkmalmodell und Systemmodellen ... 93
4.4 Nutzung des Merkmalmodells für die Funktionsbeschreibung ... 94
4.5 Nutzung des Merkmalmodells für den Strukturentwurf ... 98
4.5.1 Merkmale von Elementbeziehungen ... 98
4.5.2 Binnenzusammenhang ... 99
4.5.3 Außenzusammenhang ... 99
4.5.4 Regeln für Elementbeziehungen ... 100
4.6 Zusammenfassung ... 103
5 CAEX++ ... 105
5.1 Zielstellung ... 105
5.2 Genereller Ansatz ... 105
5.3 CAEXFile ... 106
5.4 <ExternalReference>-Element ... 106
5.5 <InstanceHierarchy>-Element ... 107
5.6 Darstellung funktionaler Modelle mit CAEX++ ... 108
5.7 Klassen-Attribute und Instanz-Attribute ... 109
5.8 Berechnung von <Attribute>-Werten ... 110
5.9 Beziehungen zwischen Elementen ... 112
5.9.1 Äquivalenz zwischen Klassen ... 112
5.9.2 Beziehungen zwischen Systemelementen ... 114
5.9.3 Erweiterung des <InternalLink>-Elements ... 116

5.9.4 Übersicht ..... 118
5.10 Beziehung zwischen CAEX-Systemmodell und Merkmalmodell ..... 119
5.10.1 Beziehung zwischen <SystemUnitClass> und IEC 61360-ItemClass ..... 119
5.10.2 Beziehung zwischen CAEX-Attribut und IEC 61360-Property ..... 121
5.11 Zusammenfassung ..... 122
6 Anwendung beim Entwurf von Systemen ..... 125
6.1 CAEX++ Editor ..... 125
6.2 Einfache Verbindungen ..... 125
6.2.1 Schraubverbindung ..... 125
6.2.2 Signal-Verbindungen ..... 129
6.3 Anwendungsbeispiel „Flexibles Fertigungssystem“ ..... 132
6.3.1 Funktionales Modell ..... 132
6.3.2 Komponentenmodell ..... 134
6.3.3 Topologiemodell, Stellenplan ..... 135
6.4 Zusammenfassung ..... 138
7 Zusammenfassung und Ausblick ..... 139
Literatur ..... 141
Glossar ..... 153
Index ..... 157
Anhang A Beispiele ..... 159
Anhang B CAEX++ Schema ..... 173

# Abbildungsverzeichnis

Abbildung 1 – Konzept - Modell - Realität ..... 1
Abbildung 2 – Struktur dieser Arbeit ..... 4
Abbildung 3 – Beispiel für Merkmalmodell [Epp11] ..... 6
Abbildung 4 – Beziehung zwischen Merkmalmodell und Systemmodell ..... 7
Abbildung 5 – System ..... 9
Abbildung 6 - Modell für Beziehungen zwischen Systemelementen ..... 11
Abbildung 7 – Beziehungstypen (basierend auf [OMG11b]) ..... 11
Abbildung 8 – Kategorien von Beziehungen ..... 12
Abbildung 9 – Die drei wesentlichen Systemaspekte ..... 18
Abbildung 10 – Schnittstellen und Elementbeziehung ..... 21
Abbildung 11 – Schnittstelle mit gemeinsamen Element ..... 23
Abbildung 12 – System „Elektropumpe“ ..... 28
Abbildung 13 – Emergenzrelation [SS10, S. 451f.] ..... 29
Abbildung 14 – Formalisierte Prozessbeschreibung ..... 35
Abbildung 15 – Aktivitätsdiagramm ..... 36
Abbildung 16 – Beispiel für ein Blockdiagramm ..... 37
Abbildung 17 – Modellbeziehungstypen (basierend auf [Müh12]) ..... 40
Abbildung 18 – Beschreibungsmittel für die verschiedenen Systemaspekte ..... 40
Abbildung 19 – Inhalt eines CAEX-Files ..... 44
Abbildung 20 – Darstellung der Aufbaustruktur eines Systems mit CAEX [IEC08, S. 53f.] ..... 45
Abbildung 21 – Beispiel für mehrere Hierarchien in einem CAEX-Dokument ..... 46
Abbildung 22 – Beispiel für <InternalLink>-Element ..... 46
Abbildung 23 – Beispiel für Verbindung mittels <InternalElement> ..... 47
Abbildung 24 – Hierarchie und Vererbung in einer CAEX-Bibliothek ..... 48
Abbildung 25 – Definition von SystemUnitKlasse und Rolle für ein Systemelement ..... 49
Abbildung 26 – Darstellung von Vererbung und Äquivalenz mit CAEX v2.15 [IEC08, S. 56f.] ..... 50
Abbildung 27 – Darstellung von Vererbung und Äquivalenz mit CAEX v3.0 [Dra13] ..... 51
Abbildung 28 – Verknüpfung von Anlagen- und Prozessbeschreibung [JCS+12] ..... 54
Abbildung 29 – Übersicht über Standards zu Merkmalmodellen ..... 56
Abbildung 30 – Beziehung zw. Merkmaldefinition und Merkmalbeschreibung ..... 57
Abbildung 31 – Darstellung des Referenzmechanismus nach [IEC10] ..... 57
Abbildung 32 – Identifier Format [ISO09b, S. 14f.] ..... 58
Abbildung 33 – IEC 61360 Datentypen ..... 60
Abbildung 34 – Kommunikation nach PROLIST nach [Zgo07, S. 14f.] ..... 63
Abbildung 35 – Struktur von eClass [eCl13] ..... 64
Abbildung 36 – Unterscheidung in Basic und Advanced [eCl13] ..... 65
Abbildung 37 – IEC 61987-10 Datentypen ..... 66
Abbildung 38 – Erläuterung von Kardinalität in [IEC09a] ..... 67
Abbildung 39 –Wie Properties und LOPs genutzt werden ..... 68
Abbildung 40 – Nutzung von Merkmalleisten im Engineering-Prozess [LPZ07] ..... 69
Abbildung 41 – High-level planning model [ISO09a, S. 9f.] ..... 70
Abbildung 42 – Beispiel für Referenzierung von Datenwert und Einheit [ISO07] ..... 71
Abbildung 43 – Informationsaustausch zwischen Lieferanten und Nutzer [Ben12, S. 14f.] ..... 72
Abbildung 44 – Beziehung zwischen soziotechnischem und technischem System ..... 73
Abbildung 45 – Engineering–Prozess ..... 74
Abbildung 46 – Anlagenlebenszyklus nach [ISO04b, S. 2f.] ..... 74

Abbildung 47 – Abstraktionsgrade [RST09, S. 6f.] ..... 76
Abbildung 48 – Entwurfsprozess laut [Dra10, S. 53f.] ..... 78
Abbildung 49 – CAEX Entwurfsprozess ..... 79
Abbildung 50 – Analyse [Wei07, S. 25f.] ..... 80
Abbildung 51 – Umsetzung von funktionalen Modellen [Wei07, S. 26f.] ..... 81
Abbildung 52 – Genereller Workflow im Engineering ..... 83
Abbildung 53 – FAVA Vorgehensmodell [FHE+12a] ..... 86
Abbildung 54 – Einordnung von FAVA in den Anlagenlebenszyklus ..... 87
Abbildung 55 – Entwurfsschritte mit Zwischenergebnissen (basierend auf FEH+12) ..... 88
Abbildung 56 – Modelle im Engineering-Prozess ..... 89
Abbildung 57 – Entwurfsschritt ..... 91
Abbildung 58 – Vergleich für zunehmende Entwicklungsunterstützung [VDF13] ..... 92
Abbildung 59 – FAVA Methodik ..... 93
Abbildung 60 – Beziehung zwischen Systemmodellen und Merkmalmodell ..... 94
Abbildung 61 – Funktion als Merkmalträger ..... 95
Abbildung 62 – Klassifizierung von Systemfunktionen ..... 96
Abbildung 63 – Klassen fertigungstechnischer Funktionen nach [Heh11] ..... 97
Abbildung 64 – Einordnung der Elementbeziehung ..... 98
Abbildung 65 – Binnenzusammenhang ..... 99
Abbildung 66 – Außenzusammenhang ..... 100
Abbildung 67 – Beziehungsmerkmal ..... 101
Abbildung 68 – Beispiel Gewindekompatibilität (Darstellung basierend auf [Jan08]) ..... 102
Abbildung 69 – Beispiel Gewindekompatibiliät als Beziehungsmerkmal ..... 103
Abbildung 70 – Erweiterung des <ExternalReference>-Elements ..... 107
Abbildung 71 – Erweiterung von <InstanceHierarchy> für Lebenszyklus-Information ..... 108
Abbildung 72 – Darstellung eines funktionalen Modells ..... 109
Abbildung 73 – Klassen- und Instanz-Attribute ..... 110
Abbildung 74 – Erweiterung des <AttributeType>-Elements ..... 111
Abbildung 75 – RefSemanticClass ..... 112
Abbildung 76 – Äquivalenz zwischen Klassen ..... 113
Abbildung 77 – <ElementRelation>-Element ..... 114
Abbildung 78 – ElementRelationCollection ..... 115
Abbildung 79 – Elementbeziehung <implements> zwischen Komponente und Funktion ..... 116
Abbildung 80 – Erweiterung des <InternalLink>-Elements ..... 117
Abbildung 81 – <LinkClassType>-Bibliothek ..... 118
Abbildung 82 – Verhältnis zwischen Merkmalmodell und CAEX-Modell ..... 119
Abbildung 83 – Referenz auf eine Klassendefinition in einem eCl@ss-Dictionary ..... 120
Abbildung 84 – CAEX++ Methodik ..... 123
Abbildung 85 – Bestimmungsgrößen eines Gewindes nach DIN 13 Teil 19 [WKT07, S. 6f.] ..... 126
Abbildung 86 – Bestimmungsgrößen eines Gewindes nach DIN 13 Teil 19 [WKT07, S. 10f.] ..... 127
Abbildung 87 – Darstellung einer Verbindungsklasse in CAEX++ ..... 128
Abbildung 88 – Darstellung einer Verbindung nach positiver Evaluierung ..... 129
Abbildung 89 – Refinement von Schnittstellen durch Vererbung dargestellt ..... 130
Abbildung 90 – Refinement von Verbindungsklassen durch Vererbung dargestellt ..... 131
Abbildung 91 – Modell für die Produktion eines Spielzeugautos ..... 132
Abbildung 92 – System „Spielzeugauto“ mit 2 Elementen ..... 134
Abbildung 93 – Einfaches Komponentenmodell für die Flexible Fertigungsanlage ..... 135
Abbildung 94 – Zuordnung von Prozessoperator zu technischer Ressource ..... 135

Abbildung 95 – Detailliertes Modell des Transportsystems [Wil11, S. 17f.] ..... 136
Abbildung 96 – Detailliertes Modell des Fertigungssystems in CAEX++ ..... 136
Abbildung 97 – <realize>-Beziehungen zwischen Elementen verschiedener Modelle ..... 137
Abbildung 98 – Verschiedene Elementbeziehungen für das Flexible Fertigungssystem ..... 137

# Tabellenverzeichnis

Tabelle 1 – Übersicht über Beziehungen nach [Müh12, S. 83f.] ..... 14
Tabelle 2 – Vergleich zwischen enger und loser Kopplung [Kay03, S. 133f.] ..... 21
Tabelle 3 –Beziehungstypen für Modellintegration basierend auf [Müh12, S. 83f.] ..... 26
Tabelle 4 – Herleitung von System-Merkmalen in Abhängigkeit von der Systemstruktur ..... 28
Tabelle 5 - Bestandteil einer Merkmaldefinition laut VDI 3682 ..... 35
Tabelle 6 – Beziehungen in verschiedenen Beschreibungsmitteln ..... 39
Tabelle 7 – Vergleich von CAEX und AutomationML ..... 42
Tabelle 8 – Übersicht über Elementbeziehungen in CAEX ..... 47
Tabelle 9 – Unterstützung von Elementbeziehungen in CAEX ..... 52
Tabelle 10 – Liste von Konzepten zur Repräsentation von „Valve“ [ECC14b] ..... 72
Tabelle 11 – Properties zur Repräsentation von „Anzahl von Sperrklappen“ [ECC14b] ..... 72
Tabelle 12 – MathML-Syntax zum Referenzieren von CAEX-Attributen ..... 111
Tabelle 13 – Unterstützung von Elementbeziehungen in CAEX++ ..... 118
Tabelle 14 – Vergleich von CAEX-SystemUnitClass und IEC 61360-ItemClass ..... 120
Tabelle 15 – Zuordnung von IEC 61987-Element und CAEX-Element ..... 121
Tabelle 16 – Vergleich von CAEX-Attribut und IEC 61360-Property ..... 121
Tabelle 17 – Merkmale von Elementen einer Schraubverbindung ..... 126
Tabelle 18 – Regeln für Schraubverbindung ..... 127
Tabelle 19 – Regeln für Schraubverbindung ..... 128
Tabelle 20 – Relevante Merkmale der Signale für unterschiedliche Entwurfsstadien ..... 129
Tabelle 21 – Regeln für Signalverbindungen ..... 130
Tabelle 22 – Merkmale des Produkts „Spielzeugauto“ und seiner Elemente ..... 133
Tabelle A.1 – Sachmerkmalleiste für eine Schraube ..... 159
Tabelle A.2 – eCl@ss Merkmale für eine Schraube ..... 160
Tabelle A.3 – eCl@ss Merkmale für eine Mutter ..... 164

# Kurzfassung

Gegenstand dieser Arbeit ist die Verwendung von Merkmalen für das Engineering von Systemen. Bereits Linnè setzte Merkmale zur Beschreibung und Klassifizierung von Objekten ein, die Industrie nutzt Merkmale mindestens seit den 1950er Jahren, seit den 1970ern existieren Standards um Systemkomponenten mittels Merkmalen zu beschreiben. Die Anwendung dieser Beschreibungen fokussiert auf die Beschaffung von Systemkomponenten, ein Engineering wird im Allgemeinen nicht unterstützt.

Damit ein Engineering von Systemen auf Merkmalen basieren kann, muss der Zusammenhang zwischen den verschiedenen Aspekten des Systems und den Merkmalen dargestellt werden. Dafür ist es notwendig die Zusammenhänge zwischen den Systemaspekten zu klären. Deswegen wird ein grundsätzliches Systemmodell eingeführt, welches die Basis zur Integration mit dem Merkmalmodell bildet.

Verschiedene heute in der Industrie genutzte Systemmodelle, Merkmalmodelle und Engineering-Prozesse werden diskutiert.

Basierend auf den eingeführten grundsätzlichen Modellen und den diskutierten Industrie-Modellen wird ein Konzept zur Integration von Systemmodell und Merkmalmodell eingeführt. Dabei werden verschiedene Aspekte der Integration dargestellt. Es wird ein angestrebter Engineering-Prozess vorgestellt, die Beziehung zwischen Systemmodellen und Merkmalmodell definiert und ein Bezug zwischen Strukturaspekten des Systemmodells und Merkmalen hergestellt.

Das Konzept zur Integration von Systemmodell und Merkmalmodell wird exemplarisch am Beispiel einer CAEX-Erweiterung umgesetzt und an Anwendungsbeispielen demonstriert.

# Abstract

Topic of this document is the application of characteristics for the engineering of systems. Already Linné used characteristics for description and classification of objects, the industry is using characteristics since the 1950s, since the 1970s there are standards for describing system components with characteristics. The current application of these descriptions focusses on procurement of system components, the engineering of systems in general is not supported.

In order to base the engineering of systems on characteristics, the relationship between different aspects of the system and characteristics needs to be clarified. In order to clarify the relationship between system aspects and characteristics, it is necessary to clarify the relationship between the different aspects of the system. That is why this document introduces a fundamental system model, which provides the base for integration with the characteristics model.

Different system models, characteristic models and engineering processes used today in the industry are discussed.

Basing on the introduced fundamental models and the discussed industry models, a concept for integration of system model and characteristics model is introduced. This introduction includes a discussion of different aspects of the integration. An engineering process is proposed, the relationship between system models and characteristics model is defined and a relation between structural aspects of the system model and characteristics is shown.

The concept for integration of system model and characteristics model is demonstrated with a proposed extension of CAEX and is discussed with some use cases.

# 1 Einführung

Menschen nehmen ihre Umwelt bzw. die physikalische Realität unterschiedlich war. In der Kindheit entwickeln Menschen die Fähigkeit zur Konzeptionalisierung. D.h. die physikalische Realität wird nicht nur in einer mentalen Darstellung abgebildet, sondern auch in einer abstrakten Repräsentation zusammengefasst – dem Konzept. Dabei ist es nicht möglich, sicherzustellen, dass verschiedene Menschen das gleiche Konzept verwenden, sondern es ist davon auszugehen, dass sich die Konzepte verschiedener Menschen unterscheiden, z.B. weil die jeweiligen Konzepte mit unterschiedlichen emotionalen Bedeutungen hinterlegt sind [Tal08].

Werden Konzepte externalisiert, so entstehen Modelle. Menschen können sich über Modelle unterhalten, sie in Beziehung zur Realität setzen und sich auf die Bedeutung der Modelle einigen. D.h. es wird möglich die externalisierten Konzepte zwischen den Menschen abzugleichen.

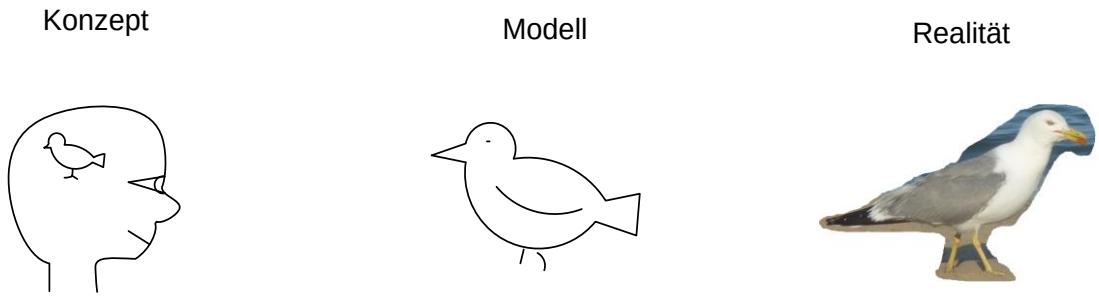


**Abbildung 1 – Konzept - Modell - Realität**

Für verschiedene Anwendungsfälle sowie für verschiedene Sichten auf die Realität gibt es verschiedene Modelle, die sich ergänzen können bzw. die unabhängig voneinander genutzt werden können (auch wenn sie die gleiche Realität beschreiben). Sollen Informationen aus diesen unterschiedlichen Anwendungsfällen integriert werden, so müssen die unterschiedlichen Modelle zueinander in Beziehung gesetzt werden.

Menschen sind gewohnt in Kategorien zu denken. Konzepte werden in verschiedene Schubladen eingeordnet, um generelle Handlungsrichtlinien bei Wahrnehmung von physikalischen Vorbildern parat zu haben. Beispiele für die ersten Klassifizierungen durch Menschen sind wohl „essbar“ bzw. „nicht essbar“ und „gefährliches Tier“ bzw. „nicht gefährliches Tier“. Solche Klassifizierungen werden anhand von Merkmalen vorgenommen, z.B. für die Einordnung in die Kategorie „gefährliches Tier“ sind Merkmale wie „große spitze Zähne“ oder „große scharfe Krallen“ oder überhaupt „groß“ wesentlich. Die Nützlichkeit der Klassifizierung ist stark abhängig vom aktuellen Anwendungsfall. Z.B. wird die Klassifizierung

von Kraftfahrzeugen anhand ihrer Farbe von Ingenieuren im Allgemeinen abgelehnt, während sie durchaus wichtige Anwendungsfälle hat – z.B. beim Zeitvertreib mit Kindern auf der Autobahn („Ich zähle die silbernen Autos, du musst die lila Autos zählen.“).

Eine Klassifizierung von Objekten anhand von Merkmalen wird auch im industriellen Umfeld genutzt. Vorreiter ist dabei das Plant Asset Management, speziell im Bereich Beschaffung. Die dafür aufgebaute und wachsende Wissensbasis soll nun auch im Engineering genutzt werden. Ausgehend vom generellen Prinzip kann man aktuell eine Formalisierung dieser Vorgehensweise und der entsprechenden Modelle beobachten. Für viele Elemente von Automatisierungssystemen werden gegenwärtig Merkmalbeschreibungen[2] erstellt (z.B. IEC 61987-10 [IEC09a], IEC 62683 [IEC11b]) und es wird an Standards für das merkmal-basierte Engineering gearbeitet (z.B. eCl@ss [eCl09], IEC 62832 [IEC14a]). - Merkmalmodelle haben das Potential die DNS des Engineerings zu werden.

DNS bestehen aus zwei Strängen einfacher Nukleotide (Adenin, Thymin, Guanin und Cytosin). Teilabschnitte dieser Stränge stellen die Gene dar, die jeweils die Bauanleitung für Proteine liefern können. Werden DNS interpretiert, so ist es möglich Proteine zu synthetisieren, zu Organen zu kombinieren und so funktionierende Organismen zu generieren [Mit09, S. 90][3]. Genauso wie die DNS sind Merkmalmodelle aus einfachen Bestandteilen (Merkmalen) zusammengesetzt. Sind die Merkmale richtig definiert, so wird es möglich sein, vollständige Systeme so zu beschreiben, dass diese Systeme basierend auf dem Merkmalmodell entwickelt bzw. generiert werden können.

Mitchel beschreibt in [Mit09] auch, dass für die Entwicklung von Organismen basierend auf DNS die richtigen Randbedingungen existieren müssen. – Für das Merkmalmodell werden diese Randbedingungen durch das Systemmodell definiert. Darum wird in dieser Arbeit ausführlicher auf Systemmodelle eingegangen (siehe 2.3).

Bis zu einem Engineering vollständig basierend auf Merkmalen ist es noch ein weiter Weg, auf dem diese Arbeit einige wenige Schritte gehen soll. Ziel ist es, die vorhandenen Beschreibungen für Systeme und Systemelemente so zu nutzen, dass ein Engineering der Systeme basierend auf den Merkmalen der Systemelemente durchführbar werden kann. Das heißt, in dieser Dissertation wird das merkmalsbasierte Engineering für Systeme diskutiert.

Die Arbeit ist wie folgt strukturiert: In Kapitel 1 werden die relevanten Grundlagen und Begriffe geklärt. Diese Klärung ist notwendig, weil grundlegende Begriffe in Forschung und Industrie mit unterschiedlicher Bedeutung verwendet werden. Hierbei erfolgt eine

[2] Bei zusammengesetzten Substantiven, die aus zwei selbständigen Substantiven gebildet werden und bei denen das Bestimmungswort auf ‚-al‘ endet, wird eigentlich ein Fugen-s verwendet. Grammatikalisch richtig wäre also „Merkmalsbeschreibung“, „Merkmalsträger“ und „Merkmalsmodell“ (siehe auch Abbildung 3). – Diese Arbeit folgt aber dem allgemeinen Sprachgebrauch – bei den meisten Veröffentlichungen im Fachgebiet wird auf das Fugen-s verzichtet.

[3] Die Darstellung ist stark vereinfacht und verkürzt.

Diskussion des für diese Arbeit relevanten Systemmodells und das Merkmalmodell wird eingeführt.

In Kapitel 1 wird der aktuelle Stand der Wissenschaft dargestellt. Verschiedene laufende Arbeiten werden vorgestellt und mit den in Kapitel 2 eingeführten Modellen in Bezug gesetzt. Es wird auf Vorgehensweisen für das Engineering eingegangen und CAEX wird eingeführt. Dieser Stand der Wissenschaft bildet die Ausgangsbasis für die weiteren Arbeiten.

Kapitel 4 und 5 stellen die Weiterentwicklungen zur Verwendung des Merkmalmodells in Engineering dar. Im Kapitel 4 wird ausgehend von einem Vorschlag für den Engineering-Prozess ein Konzept für die Integration von Systemmodell und Merkmalmodell vorgestellt. Im Kapitel 5 wird dargestellt, wie die Integration von Systemmodell und Merkmalmodell auf das CAEX-Datenmodell abgebildet werden kann und wie Merkmalbeschreibungen in die Systembeschreibung integriert werden können. In Kapitel 6 erfolgt eine Darstellung von Anwendungsfällen.

Abbildung 2 stellt die Themengebiete dieser Arbeit dar: Systemmodell und Merkmalmodell müssen erst konzeptionell erfasst und integriert werden; anschließend erfolgt die Abbildung auf ein Datenmodell; anhand von Anwendungsfällen wird die Nutzung des integrierten Modells dargestellt.

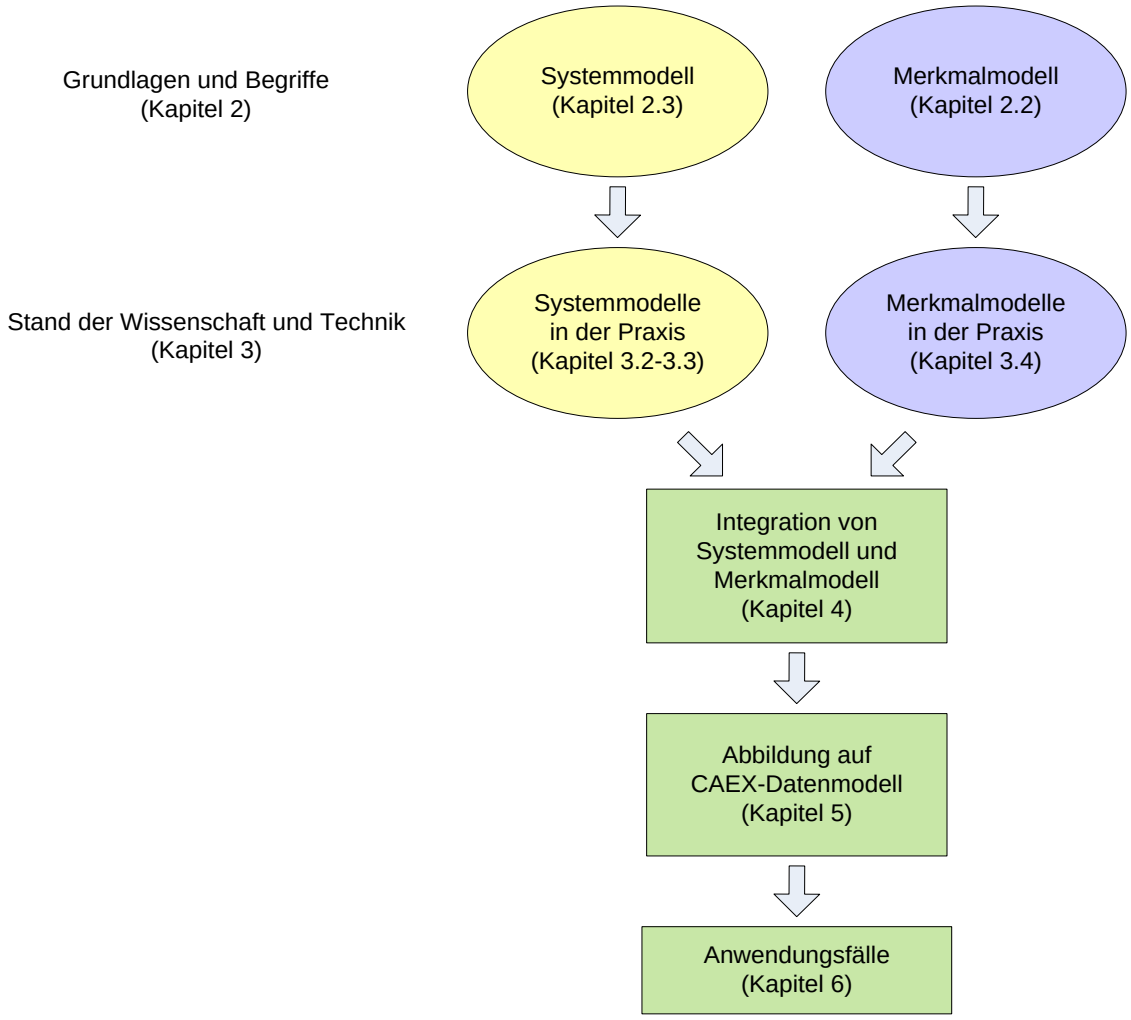


**Abbildung 2 – Struktur dieser Arbeit**

# 2 Grundlagen und Begriffe

## 2.1 Zielstellung

Wie in der Einführung dargestellt, sind Modelle ein wesentliches Mittel zur Kommunikation von Konzepten. In dieser Arbeit soll das System-Engineering basierend auf Merkmalen beschrieben werden. Deswegen ist es notwendig ein gemeinsames Verständnis der grundlegenden Modelle – Merkmalmodell und Systemmodell - herzustellen. In diesem Kapitel erfolgt die Einführung des Merkmalmodells und des Systemmodells, die dieser Arbeit zugrunde liegen. Die Einführung des Systemmodells erfolgt in zwei Stufen: als erstes wird ein grundsätzliches Systemmodell eingeführt, welches prinzipiell für alle Arten von Systemen gilt, erst im zweiten Schritt wird auf technische Systeme eingegangen.

## 2.2 Merkmalmodell

„Ein Merkmal (auch Charakteristikum) ist allgemein eine erkennbare Eigenschaft, die eine Person, eine Sache oder einen abstrakten Zusammenhang von anderen unterscheidet.“ [Wik14]

Die älteste allgemein bekannte Verwendung von Merkmalen ist die Klassifizierung von Pflanzen nach Linné [Fou94]. Linné betrachtete das Merkmal(lat. *charactere*) als Basis für die Definition der Gattung und unterschied drei Arten von Merkmale: künstliche, wesentliche und natürliche [Lin83]. Zur Klassifizierung von Gattungen setzte Linné nur wesentliche Merkmale ein, die sich für ihn aus „der sorgfältigen Beschreibung der Entwicklung der Blüte und Frucht der ersten Art [ergaben]. Alle anderen Arten der Gattung werden mit der ersten verglichen, wobei alle ungleichförmigen Merkmale ausgeschlossen werden. Nach dieser Arbeit erhält man das wesentliche Merkmal.“ ([Lin83] zitiert nach [Fou94]). Das heißt, bereits Linné unterschied zwischen Merkmalen, die für eine Klassifizierung genutzt werden konnten, weil sie allgemein erkennbar zu einer Unterscheidung führten, und Merkmalen, die nicht beachtet werden mussten.

Nach Krauser wurde in den Jahren 1950-1955 ein „*IBM part number system*“ eingeführt, bei dem es um die Klassifizierung von Teilen und Beschreibung mittels ausgewählter Merkmale ging [Kra86]. Dokumente über das System lassen sich leider nicht finden, allerdings werden „*part numbers*“ noch immer und nicht nur bei IBM genutzt um Bauteile von Ausrüstung zu identifizieren.

Seit den 1970-er Jahren gibt es die DIN 4000 (z.B. [DIN75]) in deren vielen Teilen (derzeit ca. 120 Teile) die verschiedensten konstruktiven Elemente des Maschinenbaus beschrieben werden. Krauser beschrieb 1984 eine auf Merkmalen basierende Methodik, die Basis für ein

Informationssystem bilden sollte, welches z.B. bei der Konstruktion die Wiederverwendung vorhandener Teile unterstützen sollte [Kra86].

Auch international wird bereits seit einiger Zeit an gemeinsamen Merkmalmodellen gearbeitet. IEC 61360 wurde im Jahr 1990 in der ersten Version veröffentlicht und liegt inzwischen in der dritten Revision vor [IEC09b].

Im Jahr 2000 wurde der eCl@ss e.V. gegründet, der Merkmalbeschreibungen standardisiert [eCl14]. Für den Bereich der Prozessleittechnik wurde im April 2003 die Projektgruppe ‚Merkmalleisten' (PROLIST) gegründet, die im Oktober 2003 die NAMUR-Empfehlung NE 100, Version 1.0 veröffentlichte [Nam03].

Alle diese Anwendungen von Merkmalen richteten sich darauf, die Eigenschaften eines Betrachtungsgegenstands zu benennen und möglichst einheitlich darzustellen. Dabei ging es vor allem um die Darstellung des Betrachtungsgegenstandes für sich und nicht um Zusammenhänge mit anderen Betrachtungsgegenständen. Ziel dieser Bemühungen war es vor allem die Beschaffung von Geräten der Prozessleittechnik zu unterstützen [Nam05]. Inzwischen gibt es Bestrebungen, diese auf dem Merkmalmodell basierenden Daten zu nutzen, um Eigenschaften von technischen Systemen und Systemelementen zu beschreiben [Epp11].

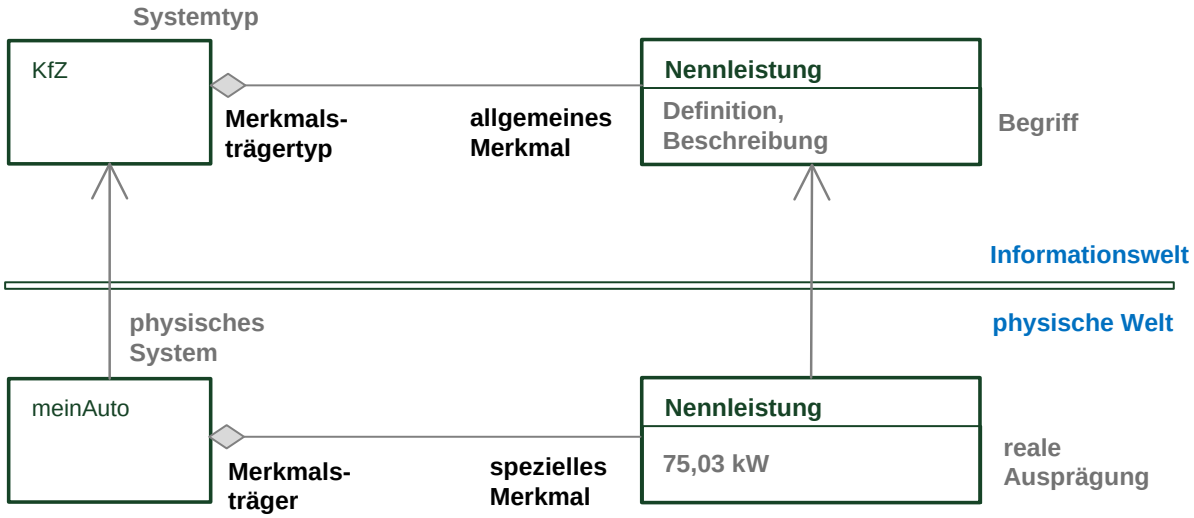


**Abbildung 3 – Beispiel für Merkmalmodell [Epp11]**

Ein Merkmal wird in dieser Arbeit als eine den Merkmalträger klassifizierende Eigenschaft angesehen. D.h. es kommt auf die im Modell verwendete Sicht an, ob das Merkmal relevant ist (also zu einer Klassifizierung führt) oder nicht relevant ist (nicht zu einer Klassifizierung führt). Z.B. kann die Farbe eines KfZs wichtig sein und zu einer Zuordnung eines KfZs zu der Klasse der „RotenAutos" führen, bei strukturellen Betrachtungen ist die Karosserie-Farbe als Merkmal eher nicht relevant und wird nicht betrachtet.

In dieser Arbeit wird folgende grundlegende Definition von Merkmal verwendet:

> *Ein Merkmal ist eine allgemein erkennbare Eigenschaft eines Betrachtungsgegenstandes, die zu einer Klassifizierung des Betrachtungsgegenstandes genutzt werden kann.*

Die Bedeutung eines Merkmals wird durch die Zuordnung zum Merkmalträger definiert. Das Merkmal übernimmt dabei dem Merkmalträger gegenüber eine Rolle, beispielsweise kann das Merkmal „Namen“ den Namen des Herstellers darstellen oder den Namen eines Zwischenverkäufers. D.h. die Semantik eines Merkmals wird nicht nur durch die Definition des Merkmals bestimmt, sondern auch durch die Zuordnung zum Merkmalträger.

Merkmalbeschreibungen sind im Allgemeinen rein phänomenologische Beschreibungen. Die Wirklichkeit in Form von Gegenständen oder Produkten wird anhand ihrer einzelnen Merkmale beschrieben. Beziehungen zwischen den Merkmalen oder Beziehungen zwischen den Merkmalträgern werden nicht betrachtet. Prinzipiell können die Merkmale zur Klassifikation der Merkmalträger genutzt werden, dabei wird jeweils nur ein Teil aller möglichen Merkmale berücksichtigt.

Abbildung 4 stellt die Beziehung zwischen Merkmalen und Systemen dar (das Systemmodell wird in 2.3 diskutiert). Merkmale können als eine spezielle Art von Objekteigenschaften aufgefasst werden. Der Unterschied zu üblichen ro ist, dass Merkmale zur Klassifizierung von Objekten verwendet werden können. Merkmale charakterisieren jeweils den entsprechenden Merkmalträger. System und Systemelemente werden als Merkmalträger betrachtet, d.h. auch sie können durch Merkmale charakterisiert werden.

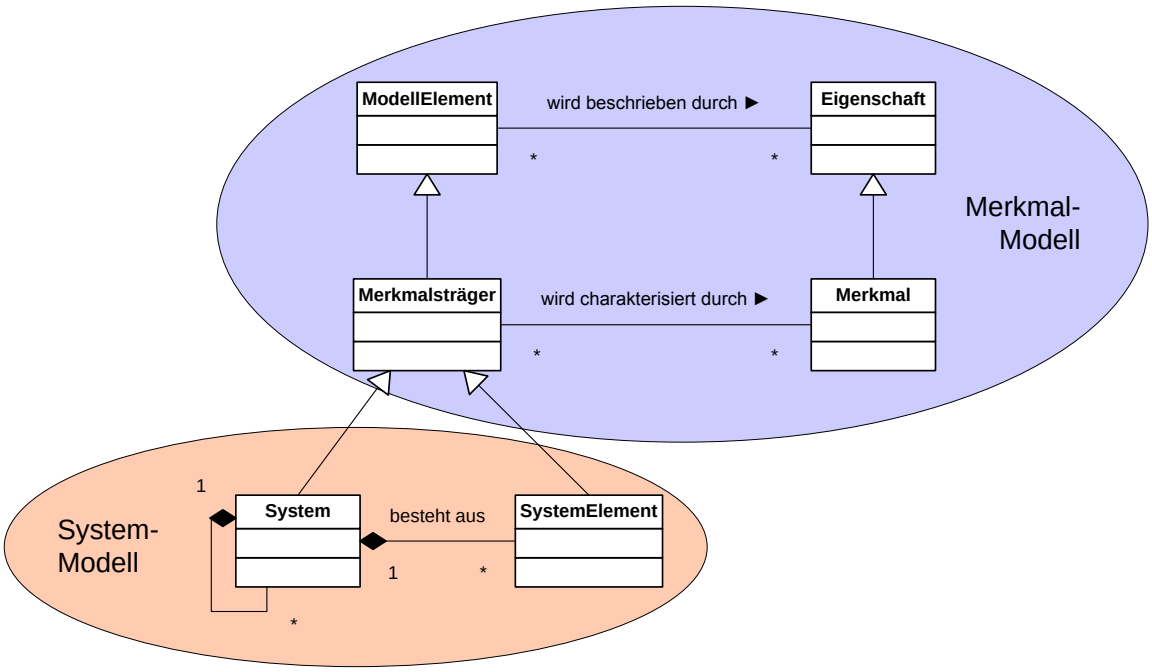


**Abbildung 4 – Beziehung zwischen Merkmalmodell und Systemmodell**

Wie in Abbildung 4 dargestellt, besteht ein System (z.B. das Auto aus Abbildung 3) aus mehreren Systemelementen (z.B. Motor und Räder). Sowohl System als auch Systemelemente können mit Merkmalen beschrieben werden. Ein Ziel dieser Arbeit ist es zu klären, wie diese Merkmale im System-Engineering verwendet werden können.

Um Merkmale zur Beschreibung und zum Engineering von Systemen nutzen zu können, muss ein eindeutig definiertes Systemmodell verwendet werden, mit dem unter anderem beschrieben werden kann, wie aus einzelnen Systemelementen ein System entsteht.

## 2.3 Systemmodell

Fast alle Modelle, mit denen ein Automatisierungsingenieur arbeitet, sind Modelle für Systeme. Auch das in Abschnitt 2.2 vorgestellte Merkmalmodell bezieht sich auf Systeme und auf ihre Systemelemente. Deswegen sollen in diesem Abschnitt das Systemmodell grundsätzlich dargestellt, grundlegende Begriffe geklärt und spezifische Eigenschaften von Systemen diskutiert werden.

### 2.3.1 Generelles Systemmodell

Es sind verschiedenste Arten von Systemen bekannt, z.B. das Sonnensystem, mathematische Systeme und verschiedenste technische Systeme. Dabei beruht die Wahrnehmung dieser Betrachtungsgegenstände als Systeme auf der Beobachtung von Zusammenhängen zwischen ihren Elementen, z.B. von Gravitationskräften zwischen den Planeten, von Ordnungsprinzipien für Zahlen oder von zu erfüllende technischen Funktionen. – D.h. das Modell „System“ wird als einheitliches, grundlegendes Modell verstanden, das allgemein und für verschiedenste Betrachtungsgegenstände verwendet werden kann.

In diesem Abschnitt wird ein entsprechendes generelles Systemmodell eingeführt.

#### *2.3.1.1 System*

Diese Arbeit stützt sich auf die Definition von ‚System‘ im Bereich des System-Engineering. Nach Patzak [Pat82, S. 19f.]gilt für das Modell ‚System‘:

- „besteht aus einer Menge von Elementen, welche Eigenschaften besitzen und welche durch Relationen miteinander verknüpft sind“;
- hat eine System-Grenze, die das System umhüllt bzw. von der Umwelt trennt [Pat82, S. 19f.].

Hitchins definiert in [Hit07, S. 28f.][Hit07, S. 28f.]: „A system is an open set of complementary interacting parts, with properties, capabilities and behaviors emerging, both from the parts and from their interactions, to synthesize a unified whole.“ – Bei dieser Definition wird hervorgehoben, dass das System ein abgeschlossenes Ganzes ist, das als Ganzes mehr bedeuten kann als die Summe der Teile, die sich gegenseitig ergänzen und die

miteinander kooperieren. Dieses System kann hergestellt (synthetisiert) werden. Die Eigenschaften, Fähigkeiten und Verhalten des Systems basieren nicht nur auf den Eigenschaften der Teile, sondern auch auf den Interaktionen zwischen den Teilen. D.h. Systeme können zusätzliche und andere Eigenschaften haben als ihre Elemente.

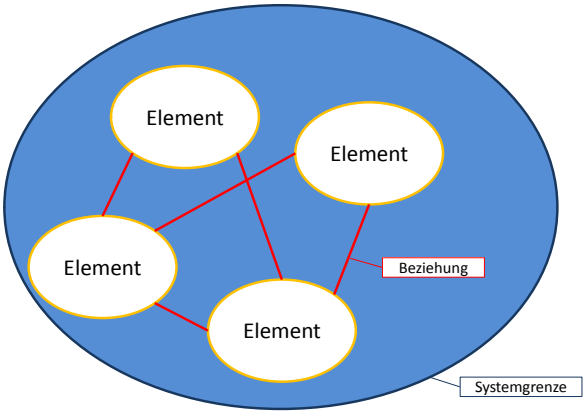


**Abbildung 5 – System**

An seiner Grenze kann das System mit seiner Umwelt interagieren.

In dieser Arbeit wird folgende grundlegende Definition von System verwendet:

> *Ein System besteht aus einer Menge von Elementen, welche durch Relationen miteinander verknüpft sind. Ein System hat eine Systemgrenze, die es von seiner Umwelt trennt, über die hinweg das System jedoch mit seiner Umwelt interagieren kann.*

Diese Definition trifft sowohl auf materielle Systeme zu (z.B. physikalische Systeme, biologische Systeme oder Automatisierungssysteme) als auch auf abstrakte Systeme (z.B. Zahlensysteme).

Im Allgemeinen wird bei der Betrachtung von Systemen vor allem auf die Aspekte ‚Struktur' und ‚Verhalten' fokussiert. Schnieder und Schnieder stellen in [SS10] dar, dass ein technisches System anhand seiner „zentralen Eigenschaften Zustand, Struktur, Funktion und Verhalten beschrieben wird". Deswegen werden diese Aspekte in den folgenden Abschnitten diskutiert. Für die Betrachtung technischer Systeme sind weitere Aspekte wichtig, die in Abschnitt 2.3.2 dargestellt werden.

#### *2.3.1.2 Struktur*

Ein System hat eine Struktur: „Die Struktur eines Systems [...] ist die Abbildung der Menge der die Komponenten eines Systems miteinander verbindenden Relationen." [Pat82, S. 39f.]

Laut Patzak können für Systeme zwei Strukturarten unterschieden werden [Pat82, S. 39f.]:

– Aufbaustruktur (Gefüge von Ordnungsrelationen), betrachtet statische Beziehungen der Systemelemente (z.B. materielle Beschaffenheit),
– Ablaufstruktur (Gefüge von Flußrelationen), betrachtet logische/zeitliche Kopplung der Systemkomponenten (z.B. Abläufe), Beispiele sind: Netzpläne, Balkenplan, Programmablaufplan, Funktionsablaufplan, Datenflußplan [Pat82, S. 48f.].

Die Aufbaustruktur wird durch die Elemente des Systems bestimmt. Dabei wird auf die Bestandsbeziehungen der Elemente fokussiert. D.h. die allgemeine Definition eines Systems (siehe 2.3.1.1) basiert auf den strukturellen Eigenschaften des Systems. – Können die Elemente des Systems in einer entsprechenden Struktur abgebildet werden, so wird von einem System gesprochen.

Die Ablaufstruktur eines Systems kann aus verschiedenen Aspekten betrachtet werden. Bei der Betrachtung der logischen Kopplung wird im Wesentlichen auf die funktionalen Verbindungen der Systemelemente fokussiert. Es wird betrachtet, welche Systemelemente miteinander interagieren und welche Systemelemente an der Bereitstellung einer Funktion des Systems beteiligt sind. Beispiele für solche Interaktionsdarstellungen sind Netzpläne oder Datenflusspläne.

Bei der Betrachtung des zeitlichen Ablaufs (z.B. Balkendiagramm, Programmablaufplan) betrachtet man eher die Struktur des Verhaltens. In diesem Fall betrachtet man nicht die Bau-Elemente des Systems, sondern die Verhaltenselemente des Systems. Für das Verhalten eines Systems gibt es zwei Sichten: eine Sicht von außen auf das System (Blackbox), bei der betrachtet wird, wie die Funktionen des Systems ausgeführt werden (d.h. es wird beschrieben, wie Eingangsgrößen angeregt werden und wie Ausgangsgrößen sich ändern) (z.B. Aktivitätsdiagramm, Prozess). Die andere Sicht (Whitebox) beschreibt das Systemverhalten basierend auf den Zuständen des Systems (z.B. Zustandsdiagramm) (siehe auch [IEE90]).

Schnieder und Schnieder definieren die Struktur als Gesamtheit der Beziehungen zwischen den Teilen eines Systems, und unterscheiden zwischen Beziehungen innerhalb des Systems (intrasystemische Relationen) und Beziehungen des Systems zu seiner Umgebung (extrasystemische Relationen) [SS10].

In dieser Arbeit werden folgende Definitionen verwendet:

*Ein System hat zwei Strukturaspekte:*

- *Kompositionelle Struktur (Patzak: Aufbaustruktur) in der die Elemente durch Bestandsbeziehungen (besteht aus) verbunden sind.*
- *Funktionale Struktur (Patzak: Ablaufstruktur) in der die Elemente durch funktionale Verbindungen verbunden sind.*

Die Elementbeziehungen (Bestandsbeziehungen und funktionale Verbindungen) sind in Abschnitt 2.3.1.2.1 beschrieben.

Die funktionale Struktur nach dieser Definition kann zeitliche Aspekte beinhalten (z.B. die zeitliche Folge der Interaktion zwischen den Systemelementen).

Andere Aspekte der von Patzak definierten Ablaufstruktur (z.B. Programmablaufplan, Funktionsablaufplan) werden in dieser Arbeit dem Verhalten zugeordnet (siehe 2.3.1.4).

### 2.3.1.2.1 Elementbeziehungen

Die Beziehungen zwischen den Elementen eines Systems werden als wesentliches Kennzeichen des Systems betrachtet.

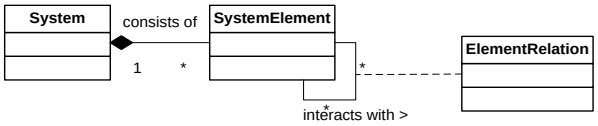


**Abbildung 6 - Modell für Beziehungen zwischen Systemelementen**

UML (Version 2) wird in dieser Arbeit als anerkanntes und erfolgreiches Sprachsystem für Systembeschreibungen betrachtet. Für verschiedene Systemaspekte werden in UML verschiedene Sprachen definiert.

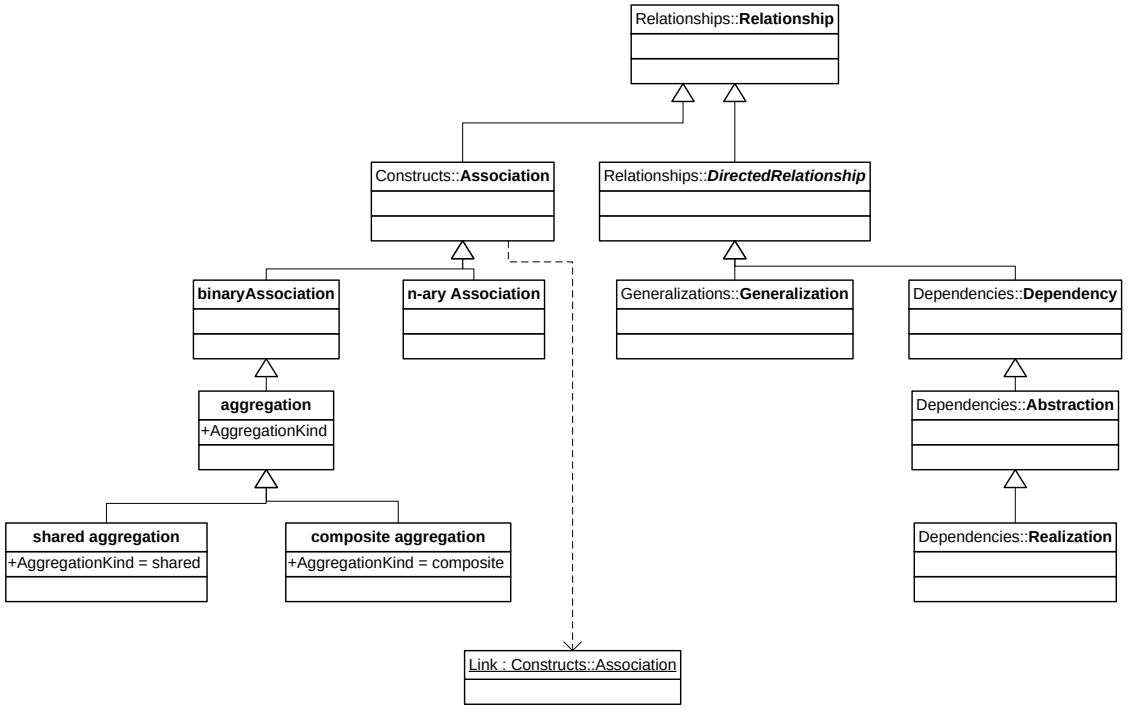


**Abbildung 7 – Beziehungstypen (basierend auf [OMG11b])**

Wie in Abbildung 7 dargestellt, berücksichtigt das Basismodell von UML eine Reihe von Beziehungen zwischen Systemelementen. Nach [OMG11b] ist die Unterscheidung zwischen Verhältnis (DirectedRelationship) und Verbindungen (Association) eine wesentliche Unterscheidung. Eine Verbindung ist eine Beziehung, die sich in der modellierten Realität erkennbar ausprägt, entweder dadurch dass die verbundenen Systemelemente miteinander interagieren oder dass eine Bestandsbeziehung existiert. Verhältnisse sind abstrakter Natur und werden in der Realität nicht erkennbar ausgeprägt. SysML benutzt die gleiche Definitionen für Beziehungen, leitet aber von den verschiedenen Beziehungstypen weitere Typen ab [OMG08].

Verbindungen kann man unterscheiden in bilaterale Verbindungen (engl.: *binary association*, zwischen zwei Elementen) und multilaterale Verbindungen (engl.: *n-ary association*, zwischen mehreren Elementen). Eine Bestandsbeziehung (engl.: *aggregation*, Teil-Ganzes-Beziehung) ist immer eine bilaterale Verbindung. Die Bestandsbeziehung kann unterteilt werden in Aggregation (engl.: *shared aggregation*, bei der das Teil selbständig existieren kann) und in Komposition (engl.: *composite aggregation*, bei der das Teil nicht unabhängig vom Ganzen existieren kann). Weitere Verbindungen sind im Sprachsystem von UML nicht definiert, es wird jedoch davon ausgegangen, dass in den jeweiligen Anwendungsmodellen entsprechende (fach-)spezifische funktionale Verbindungstypen definiert werden.

Im Weiteren werden sich die Betrachtungen von Elementbeziehungen in dieser Arbeit an dem Ordnungsschema von UML orientieren. In dieser Arbeit werden Elementbeziehungen wie folgt klassifiziert:

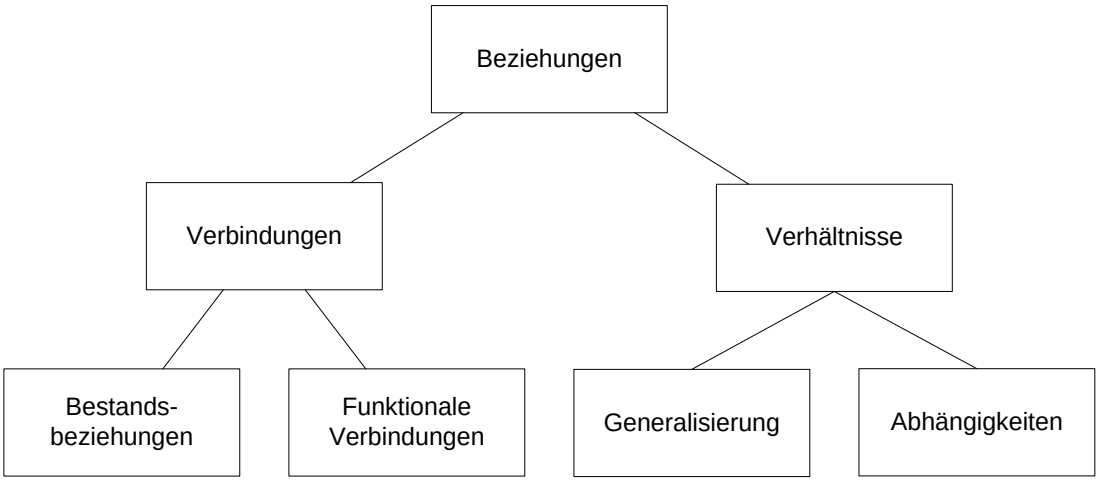


**Abbildung 8 – Kategorien von Beziehungen**

In technischen Systemen sind Bestandsbeziehungen, funktionale Verbindungen und Abhängigkeiten von besonderer Bedeutung.

Bestandsbeziehungen beschreiben die Zusammensetzung eines Systems (ist-Teil-von, besteht-aus). Da Systemelemente selbst als Systeme betrachtet werden können, ist es möglich, dass diese Art von Beziehungen zum mehrstufigen, hierarchischen Aufbau von Systemen führt. In UML wird diese Art von Beziehung durch strukturierte Klassen bzw. durch Aggregations-/Kompositions-Beziehungen beschrieben [OMG11a].

Funktionale Verbindungen sind Verbindungen, die einen funktionalen Zusammenhang herstellen. Über die funktionalen Verbindungen zwischen den Systemelementen erfolgen die Interaktionen zwischen den Systemelementen. Diese Interaktion kann sich auf Material, Energie oder Information beziehen, wobei diese jeweils als Ausgangsgrößen von Systemelementen über die funktionalen Verbindungen anderen Systemelementen als Eingangsgrößen zugeführt werden. Quellen der Eingangsgrößen von Systemelementen können Eingangsgrößen des Systems, Zustandsgrößen des Systems oder Ausgangsgrößen von anderen Systemelementen sein. Ausgangsgrößen von Systemelementen können auch Zustandsgrößen des Systems oder Ausgangsgrößen des Systems sein. Betrachtet man die Verkettung ausgehend von Eingangsgrößen des Systems bis hin zu den Ausgangsgrößen des Systems, so sind funktionale Verbindungen zwischen den Systemelementen eine notwendige Voraussetzung für die Bereitstellung von Funktionen des Systems. Interaktionen werden bei der Beschreibung der Systemstruktur selten explizit definiert, sondern sind implizit Bestandteil der funktionalen Verbindungen zwischen den Elementen. Eine Beschreibung der Interaktionen erfolgt oft mit Mitteln der Verhaltensbeschreibung.

Abhängigkeiten beschreiben Abhängigkeitsbeziehungen zwischen Systemelementen, Abstraktion (z.B. <derive>-Stereotyp, Realization) bzw. Vererbungsbeziehungen (Generalization). Abhängigkeitsbeziehungen werden genutzt um Beziehungen zwischen Elementen verschiedener Modelle darzustellen. Wenn beispielsweise ein Modell die Umsetzung eines abstrakteren Modells beschreibt, wird die ‚Realization'-Beziehung genutzt um darzustellen, wie die Modelle in Beziehung stehen und welche Modellelemente des Umsetzungsmodells welche Modellelemente des abstrakteren Modells umsetzen. Vererbungsbeziehungen werden im Allgemeinen genutzt um Beschreibungen zu vereinfachen bzw. zu verallgemeinern. Beispiele für diesen Beziehungstyp sind Typ-Instanz-Beziehungen bzw. Typ-Subtyp-Beziehungen. Die Vereinfachung erfolgt, indem Beschreibungen bzw. Definitionen für einen Typen erstellt werden und diese Beschreibungen dann für alle Instanzen des Typs gelten. Es kann Typen für Strukturen, Verhalten und Funktionen geben. Ein Typ wird als Menge aller Instanzen mit bestimmten gleichen Eigenschaften betrachtet. – Dieses Konzept ist relativ alt und wird schon in [Che76] als ‚entity-set' beschrieben. In UML wird ein Typ als ‚Klasse' bezeichnet [OMG11a, S. 48f.].

Mühlhause dokumentiert in [Müh12], dass Elementbeziehungen wichtig für die Beziehungen von Systemmodellen sind, deswegen soll hier eine Kategorisierung der dort verwendeten Elementrelationen erfolgen (siehe Tabelle 1).

**Tabelle 1 – Übersicht über Beziehungen nach [Müh12, S. 83f.]**

| Beziehungstyp | Kategorisierung | Richtungstyp |
|---|---|---|
| Assoziation | Funktionale Verbindung | Ungerichtet |
| typisierte Assoziation | Funktionale Verbindung | Ungerichtet / gerichtet |
| Komplementärbeziehung | Verhältnisbeziehung | Ungerichtet |
| Äquivalenz | Verhältnisbeziehung | Ungerichtet |
| Teiläquivalenz | Verhältnisbeziehung | Gerichtet |
| Aggregation | Bestandsbeziehung | Gerichtet |
| Komposition | Bestandsbeziehung | Gerichtet |
| Generalisierung | Generalisierung | Gerichtet |
| Realisierung | Verhältnisbeziehung | Gerichtet |
| Abhängigkeit | Abhängigkeit | Gerichtet |
| Regel (Implikation) | Abhängigkeit. | Gerichtet |
| Formel | Abhängigkeit / Funktionale Verbindung | Gerichtet |

#### *2.3.1.3 Funktion*

Für den Begriff ‚Funktion' gibt es unterschiedliche Definitionen. Es gibt mathematische Funktionen, Software-Funktionen und Funktionen im System-Engineering. Eine Reihe von weiteren Definitionen für den Begriff Funktion wurden in [Mat02] gesammelt.

Die Mathematik definiert eine Funktion wie folgt: „Die eindeutige Abbildung einer Menge *X* auf eine Menge *Y*, dargestellt als Menge *f* der geordneten Paare *[x; y]*, heißt Funktion. Jedem Element $x \in X$ ist genau ein Element $y \in Y$ zugeordnet." [Bar84, S. 135f.]

In der Wertanalyse wird eine Funktion definiert als „'... jede ... Wirkung eines ... Objektes' (unter bewußtem Einbeziehen der zweifachen Bedeutung des Wortes 'Wirkung' als 'das Wirken' und dessen Ergebnis, 'die Wirkung'." [Gie90, S. 20f.]

Im System-Engineering definiert Patzak die Gesamtfunktion eines Systems wie folgt:

„Wird ein System [...] als Ganzes betrachtet, ohne auf die innere Struktur einzugehen, so liegt eine sogenannte black-box-Betrachtung vor. Das Systemverhalten ist im vorliegenden Fall durch die Beziehung der Inputmenge X mit der Outputmenge Y definiert. Es handelt sich somit um eine Relation zwischen Flussgrößen (Materie, Energie, Information). Dieses Übertragungsverhalten des Systems/der Komponente wird bei kausalem Verhalten als Gesamtfunktion des Systems/ der Komponente bezeichnet." [Pat82, S. 57f.]

Und für die Funktion eines Systems gilt:

„Die Funktion eines Systems ist jener Ausschnitt des Systemverhaltens (Gesamtfunktion) der zur Zweckerfüllung herangezogen wird.
Sie wird erfasst durch Angabe

- eines Hauptwortes (das Objekt betreffend, auf welches eingewirkt wird)
- eines Zeitwortes (die Art betreffend, wie auf das Objekt eingewirkt wird)

Beispiel: Systemfunktion 'Lasten heben' " [Pat82, S. 164f.]

Schnieder und Schnieder beschreiben Funktion als die Überführung der Zustandsmengen der Eingangsgrößen in die Zustandsmengen der Ausgangsgrößen [SS10].

Matthiesen stützt in [Mat02] die Funktions-Definition auf die Definition in der VDI 2221[VDI93]. Nach Matthiesen definiert die VDI 2221 wie folgt: „Die Funktion ist eine ‚lösungsneutral beschriebene Beziehung zwischen Eingangs-, Ausgangs- und Zustandsgrößen eines Systems'". Matthiesen selbst nutzt folgende Definition: „Eine Funktion ist eine lösungsneutral beschriebene Beziehung zwischen Eingangs-, Ausgangsgröße eines Systems."

In dieser Arbeit wird (nach [Mat02]) folgender Funktionsbegriff verwendet:

*Eine Funktion ist eine lösungsneutral beschriebene Beziehung zwischen Eingangsgrößen und Ausgangsgrößen eines Systems.*

Diese Definition trifft auch auf mathematische Funktionen zu: mathematische Funktionen beschreiben die Relation zwischen Elementen eines mathematischen Zahlensystems. Dabei sind Eingangs- und Ausgangsgrößen (*x* und *y*) der Funktionen gleichzeitig auch die Elemente des Systems.

Ein System kann eine oder mehrere Funktionen bereitstellen, an deren Bereitstellung jeweils unterschiedliche Elemente des Systems beteiligt sein können. D.h. wenn eine Menge von Systemelementen $S_A$ an der Funktion A beteiligt ist und eine Menge $S_B$ an der Funktion B beteiligt ist, dann können diese Mengen identisch sein, sich überschneiden oder disjunkt sein. Wenn mehrere Systemelemente an der Bereitstellung einer Funktion beteiligt sind, so müssen diese Elemente miteinander über funktionale Verbindungen gekoppelt sein. Die Kette von den Eingangsgrößen einer Funktion über die beteiligten Elemente und funktionalen Verbindungen bis hin zu den Ausgangsgrößen der Funktion kann man als „funktionalen Pfad" bezeichnen. Überschneidungen von Funktionen und relevante funktionale Pfade müssen bei der Betrachtung von Eigenschaften des Systems in Bezug auf die Funktionen berücksichtigt werden (z.B. bei der Betrachtung der Verfügbarkeit einer Funktion).

In der VDI 2221 wird eine Klassifizierung der Funktionen in Gesamtfunktion, Teilfunktion und Elementarfunktion vorgenommen. In dieser Arbeit wird „Gesamtfunktion" als Funktion eines Systems entsprechend Patzak verwendet, während „Teilfunktion" jeweils die Funktion eines Systemelements beschreibt.

Funktionen können verschieden klassifiziert werden:

- Nach Zuordnung zu einem System, das diese Funktion bereitstellt. (z.B. Anlagenfunktion, Steuerungsfunktion)
- Nach Typ der Eingangs- bzw. Ausgangsgrößen (z.B. Material, Energie, Information)
- Nach dem zeitlichen Verhalten der Funktion (z.B. zustandsbehaftet/zustandsfrei, kontinuierlich/diskret)

D.h. ‚bereitstellendes System', ‚Typ der Eingangsgröße', ‚Typ der Ausgangsgröße' und ‚Zeitverhalten' können als Merkmale der Funktion aufgefasst werden.

#### *2.3.1.4 Verhalten*

Verhalten beschreibt die Reaktion eines Systems auf Eingangsgrößen. Es gibt verschiedene Verhaltensbeschreibungen für Systeme. Bei einer „Blackbox"-Beschreibung wird das äußere, sichtbare Verhalten eines Systems beschrieben. Diese Verhaltensbeschreibung stützt sich auf die Beschreibung der Ausführung von Funktionen des Systems. Bei einer „Whitebox"-Beschreibung wird auch das innere, von außen nicht immer sichtbare Verhalten des Systems beschrieben. Diese Verhaltensbeschreibung stützt sich auf die Beschreibung der Zustände und der Zustandsübergänge des Systems. In den folgenden Abschnitten wird auf beide Verhaltensbeschreibungen eingegangen.

Schnieder und Schnieder stellen in [SS10] dar, dass das dynamische Verhalten des Systems „durch die Zuordnung von Zuständen zu einer monotonen Folge von Referenzzuständen, etwa der Zeit, definiert" wird. Dabei werden die Referenzzustände durch die Umgebung des Systems definiert (siehe auch 2.3.1.3). – In dieser Arbeit werden bei der Beschreibung eines Systems nur die Zustandsgrößen des Systems selbst berücksichtigt. Aufgrund dieser Zuordnung und da eine Verhaltensbeschreibung eines Systems basierend auf den Zustandsgrößen des Systems der allgemeinen Praxis entspricht, wird der Zustand (bzw. die Zustandsgrößen) in dieser Arbeit dem Verhalten des Systems zugeordnet.

##### 2.3.1.4.1 Prozessbeschreibung

Die Ausführung von Funktionen eines Systems wird durch Prozesse beschrieben.

Osterloh beschreibt: „Ein Prozess beschreibt einen Ablauf, das heißt den Fluss und die Transformation von Material, Informationen, Operationen und Entscheidungen. Geschäftsprozesse sind durch die Bündelung und die strukturierte Reihenfolge von funktionsübergreifenden Aktivitäten mit einem Anfang und einem Ende sowie klar definierten Inputs und Outputs gekennzeichnet. Prozesse sind ‚structure for actions'. (Davenport 1993, S.5)" [OF06, S. 33f.]

Davenport (das Original) liest sich wie folgt: „A process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action"[Dav93, S. 5f.]. D.h. ein Prozess ist eine Anordnung von Arbeits-Aktivitäten in Zeit und Raum, mit einem Anfang, einem Ende und definierten Eingangsgrößen und Ausgangsgrößen.

Begreift man eine Arbeitsaktivität mit definierten Eingangs- und Ausgangsgrößen als eine System-Funktion, dann entspricht ein Prozess der Anordnung von (einer oder mehreren)

Funktionen in Zeit und Raum, mit einem Anfang (Anfangszustand) und einem Ende (Endzustand). Anfangs- und Endzustand beziehen sich dabei nicht auf den Zustand des Systems, sondern auf den Zustand der Systemumgebung (z.B. Zustand von Produkten). Basierend auf diesem Verständnis wird ein Prozess in dieser Arbeit verstanden als:

> *Ein Prozess beschreibt die Ausführung von einer oder mehreren Funktionen eines Systems mit zeitlichem und räumlichem Bezug, mit Anfangs- und Endzustand der Systemumgebung.*

D.h. eine Prozessbeschreibung stellt dar, welche Systemfunktionen in welcher Reihenfolge ausgeführt werden müssen, um von einer definierten Anfangssituation zu einer definierten Endsituation zu gelangen.

#### 2.3.1.4.2 Zustandsbeschreibung

Das Verhalten eines Systems wird charakterisiert durch die Zustandsvariablen (die Zustände) des Systems. Können die Zustandsvariablen eines Systems Werte aus einem zusammenhängenden Wertebereich einnehmen, so spricht man von einem kontinuierlichen System. Können die Zustandsvariablen eines Systems nur diskrete Werte annehmen, dann spricht man von einem diskreten System.

Das Verhalten kontinuierlicher Systeme kann mit Differentialgleichungen modelliert werden. Das Verhalten diskreter Systeme kann mit Zustandsdiagrammen modelliert werden.

Der Zustand eines Systems wird durch die Eingangsgrößen des Systems beeinflusst. D.h. durch die Eingangsgrößen kann eine Änderung des Zustands eines Systems bewirkt werden. Bei kontinuierlichen Systemen wird der Zusammenhang zwischen Eingangsgrößen und Zustandsvariablen mittels mathematischer Funktionen[4] beschrieben. Bei diskreten Systemen wird der Zusammenhang zwischen Eingangsgrößen und Zuständen mittels Zustandsmaschinen beschrieben. Zustandsmaschinen bestehen aus Zuständen und Zustandsübergängen, welche durch Eingangsgrößen des Systems angeregt werden können.

Der Zustand eines Systems kann Einfluss auf die Funktion des Systems haben. Z.B. sind verschiedene Funktionen eines Feldgerätes nur in bestimmten Betriebszuständen des Gerätes verfügbar oder die Ausgangsgrößen eines Systems hängen von Zustandsvariablen ab. D.h. die Zustandsvariablen eines Systems können als Eingangsgrößen für die Systemfunktion dienen.

[4] An dieser Stelle beachte: mathematische Funktionen sind Funktionen eines Zahlensystems, nicht Funktionen des betrachteten Systems. D.h. die mathematischen Funktionen sind an dieser Stelle nur ein Hilfsmittel zur Beschreibung des Zusammenhangs zw. Eingangsgrößen und Zustandsvariablen.

### *2.3.1.5 Zusammenfassung*

Im Rahmen dieser Arbeit wird davon ausgegangen, dass ein technisches System definiert wird durch: Struktur, Verhalten und Funktion.

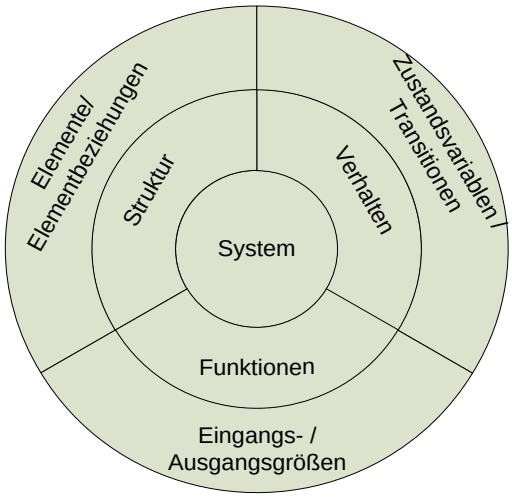


**Abbildung 9 – Die drei wesentlichen Systemaspekte**

Struktur wird durch die strukturellen Beziehungen der Systemelemente definiert (Bestandsbeziehungen und funktionale Verbindungen).

Das Verhalten eines Systems beschreibt die zu erwartende Reaktion (Zustandsänderung) eines Systems zu einem bestimmten Zeitpunkt (in einem bestimmten Zustand) auf mögliche Eingangsgrößen.

Funktion beschreibt den Zusammenhang zwischen Eingangsgrößen und Ausgangsgrößen des Systems. Ein System kann mehrere Funktionen haben, d.h. eine Funktion kann den Zusammenhang zwischen einen Teil der Eingangsgrößen und einem Teil der Ausgangsgrößen beschreiben.

Die Aspekte des Systems können nicht unabhängig voneinander betrachtet werden. Struktur, Verhalten und Funktion sind eng miteinander verknüpft. Wird zum Beispiel die Struktur eines Systems geändert, so hat das Auswirkungen auf das Verhalten. Die Ausführung von Funktionen kann wesentlich durch den aktuellen Zustand des Systems beeinflusst werden, wenn die entsprechende Zustandsgröße eine Eingangsgröße der Funktion ist.

### 2.3.2 Aspekte technischer Systeme

Neben den Aspekten des generellen Systemmodells, gibt es für technische Systeme (z.B. in der Automatisierungstechnik) weitere wichtige Aspekte. Basierend auf dem generellen Systemmodell sind detailliertere Definitionen für die Beschreibung technischer Systeme notwendig. In diesem Abschnitt werden die Aspekte technischer Systeme eingeführt, die für diese Arbeit wesentlich sind.

#### *2.3.2.1 Muster für Teilnehmer an einer Elementbeziehung*

An die Teilnehmer einer Elementbeziehung (z.B. einer funktionalen Verbindung) werden im Allgemeinen definierte Erwartungen in Bezug auf Eigenschaften und Verhalten gestellt. Diese Erwartungen werden durch Muster (Pattern) beschrieben, so dass sie leicht auf den jeweiligen Partner abgebildet werden können. Dabei werden die Erwartungen häufig jeweils durch das Ende der Beziehung definiert. Z.B. in einer Publisher-Subscriber-Beziehung gibt es unterschiedliche Erwartungen an den Teilnehmer auf der Publisher-Seite und an die Teilnehmer der Subscriber-Seite. Erwartungen an den Publisher sind z.B. dass er wesentliche Informationen mit definierten Wiederholungen bereitstellen kann, während die Erwartung an den Subscriber ist, dass er die Informationen empfangen kann. Wäre der Publisher z.B. nicht in der Lage, die Informationen in einem gewünschten Zyklus bereitzustellen, müsste evtl. eine andere Art der Kommunikationsbeziehung genutzt werden. Diese Erwartungen an den Teilnehmer einer Beziehung werden durch ein Muster beschrieben, dass häufig als ‚Rolle' bezeichnet wird (siehe auch [DV14]).

*Eine Rolle beschreibt das Muster (Pattern) für den Teilnehmer einer Elementbeziehung. Dabei kann eine Elementbeziehung unterschiedliche Rollen für die unterschiedlichen Endpunkte der Beziehung definieren. Ein solches Muster definiert Erwartungen in Bezug auf die Struktur, die Funktion und das Verhalten des Teilnehmers.*

Die Beschreibung der Rolle kann auch als Anforderungsdefinition für den Teilnehmer der Elementbeziehung verstanden werden. – Ist ein Beziehungsteilnehmer nicht in der Lage, die in die Rolle gesetzten Erwartungen zu erfüllen, kann er nicht an dieser Seite der Elementbeziehung teilnehmen.

#### *2.3.2.2 Grad der Kopplung*

Funktionale Verbindungen zwischen Systemelementen können in Bezug auf den Grad der Kopplung bewertet werden. Software-technisch wird im Allgemeinen zwischen enger Kopplung und loser Kopplung unterschieden [Kay03].

In der Elektrotechnik wird der Grad der Kopplung in Bezug zu Signalkopplungen (z.B. induktiv, kapazitiv) betrachtet. Der Grad der Kopplung entspricht dabei der Stärke der Signalübertragung bzw. der Stärke der Beeinflussung des Systemelements durch das übertragene Signal [Sch05].

Der Gedanke der Kopplung geht zurück auf die Organisationstheorie. Glassman beschreibt, dass der Grad der Kopplung von der Verwendung von Variablen abhängt:

„The degree of coupling, or interaction, between two systems depends on the activity of the variables they share. To the extent that two systems either have few variables in common or if the common variables are weak compared to other variables, which influence the system, they are independent of each other. It is convenient to speak of such a situation as one of loose coupling … In the fully joined system (...) a perturbation in any one variable would require readjustment of all the other variables of the system.“ [Gla73, S. 84f.]

Im Software-Engineering wird Kopplung im Zusammenhang mit dem Entwurf modularer Software betrachtet. Die Klassifikation der Kopplung richtet sich dabei nach der Art der Daten (bei Glassmann ‚Variable'), die zwischen den Modulen ausgetauscht werden. Stevens et al. führten eine Klassifikation von Kopplungen ein [SMC74], die von Felton und Pfleeger in [FP97, S. 310f.]wie folgt präzisiert wurde:

„
- No coupling relation $R_0$: x and y have no communication; that is, they are totally independent of one another.
- Data coupling relation R1: x and y communicate by parameters, where each parameter is either a single data element or a homogeneous set of data items that incorporate no control element. This type of coupling is necessary for any communication between modules.
- Stamp coupling relation $R_2$: x and y accept the same record type as a parameter. This type of coupling may cause interdependency between otherwise unrelated modules.
- Control coupling relation $R_3$: x passes a parameter to y with the intention of controlling its behavior; that is, the parameter is a flag.
- Common coupling relation $R_4$: x and y refer to the same global data. This type of coupling is undesirable; if the format of the global data must be changed, then all common-coupled modules must also be changed.
- Content coupling relation $R_5$: x refers to the inside of y; that is, it branches into, changes data in, or alters a statement in y.

The relations are listed from least dependent at the top to most dependent at the bottom, so that Ri > Rj for i > j. We say that x and y are loosely coupled if i is 1 or 2, and they are tightly coupled if i is 4 or 5.“

Das heißt, die Beziehungstypen R1-R2 werden als lose Kopplung eingeordnet, Beziehungstyp R3 (*control coupling relation*) wird als weder eng noch lose gekoppelt eingestuft und Beziehungstypen R4-R5 werden als enge Kopplung betrachtet.

Page-Jones hob als besonderen Fall ‚tramp'-Data hervor, die per *data coupling* Verbindungen durch unnötig viele Module transportiert werden, bevor sie verwendet werden [Pag88]. Diese Verbindungen (*data coupling relation* mit *tramp data*) werden auch als ‚*tramp coupling*' bezeichnet [OHK93].

Lose Kopplung ist ein wesentliches Element von serviceorientierten Architekturen (SOA) und ermöglicht eine flexible Verwendung der Systemelemente in unterschiedlichsten Anwendungen. Tabelle 2 zeigt einen Vergleich aus Sicht der Informatik.

**Tabelle 2 – Vergleich zwischen enger und loser Kopplung [Kay03, S. 133f.]**

| | Tightly Coupled | Loosely Coupled |
|---|---|---|
| Interaction | Synchronous | Asynchronous |
| Messaging Style | RPC | Document |
| Message Paths | Hard Coded | Routed |
| Technology Mix | Homogeneous | Heterogeneous |
| Data Types | Dependent | Independent |
| Syntactic Definition | By Convention | Published Schema |
| Bindings | Fixed and Early | Delayed |
| Semantic Adaptation | By Re-coding | Via Transformation |
| Software Objective | Re-use, Efficiency | Broad Applicability |
| Consequences | Anticipated | Unexpected |

Bei einer engen Kopplung besteht eine starke Abhängigkeitsbeziehung zwischen den Systemelementen. Eine enge Kopplung kann definiert werden durch einen kontinuierlichen Signalaustausch, eine starke Abstimmung der Schnittstellen aufeinander (z.B. durch Spezialisierung) bzw. eine starke Abhängigkeit im Verhalten der Elemente. Eine lose Kopplung kann durch einen sporadischen Signalaustausch über möglichst allgemeingültige Schnittstellen charakterisiert werden und durch eine Unabhängigkeit der Schnittstellen und im Verhalten.

#### *2.3.2.3 Schnittstelle*

Eine Schnittstelle (oder engl. ‚Interface') kann beschrieben werden als „Punkt einer Begegnung oder Kopplung zwischen zwei oder mehr Systemen" [Hal94, S. 168f.].

Laut IEV sind Schnittstellen die „Grenze zwischen zwei Funktionseinheiten, die durch funktionale Merkmale, Signalkenngrößen oder andere Merkmale als geeignet festgelegt ist [IEC 2382-9]" [DKE12a, 351-21-35].

UML2 definiert eine Schnittstelle als einen Classifier, der eine Deklaration einer Anzahl von zusammengehörigen öffentlichen (strukturellen bzw. Verhaltens-) Merkmalen und Verpflichtungen repräsentiert. Eine Schnittstelle beschreibt einen Vertrag, den jede Instanz einer Klasse, welche die Schnittstelle bereitstellt, erfüllen muss [OMG11a, S. 86f.].

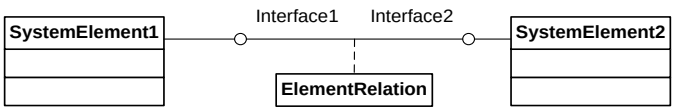


**Abbildung 10 – Schnittstellen und Elementbeziehung**

Schnittstellen werden durch Eingangs- und Ausgangsgrößen gekennzeichnet. Diese Größen können Material, Energie oder Information repräsentieren. Das IEC Wörterbuch für Steuerungstechnik (IEV control) definiert veränderliche physikalische Größen, deren Parameter Informationen übertragen als Signal [IEC06a, S. 29f.]. D.h. Schnittstellen können sowohl für die Übertragung von Material und Energie (welche keine Informationen übertragen) genutzt werden als auch für die Übertragung von Signalen (welche evtl. mit einer Material- oder Energie-Übertragung verbunden sind).
Um einen Signalaustausch zu ermöglichen, müssen die beteiligten Systemelemente über Signalanknüpfungspunkte verfügen, die zueinander passen und die miteinander verbunden sind. Verschiedene Bedingungen müssen dafür erfüllt werden: Die Eigenschaften der Schnittstellen müssen einander entsprechen – die Schnittstellen müssen den jeweils für die Verbindung notwendigen Rollen entsprechen (siehe 2.3.2.1). Dies kann bedeuten, dass einige Eigenschaften gleich sein müssen und dass einige Eigenschaften sich unterscheiden müssen. Z.B. muss jeweils eine passende Menge an Signalen ausgegeben bzw. empfangen werden. Die Richtung der Signale (Ausgangs- und Eingangssignale) muss sich unterscheiden. Die wesentlichen Eigenschaften der Signale (z.B. Datentyp, Wertebereich etc.) müssen zueinander passen.

Prinzipiell muss eine Ausgangssignal-Schnittstelle mit einer Eingangssignal-Schnittstelle verbunden werden. Es gibt die Möglichkeit, dass ein Signal an viele Eingangs-Schnittstellen übertragen wird (1→n) bzw. dass mehrere Signale an eine Eingangs-Schnittstelle übertragen werden (m→1).

Schnittstellen können sowohl als Bestandteil der Struktur als auch als Bestandteil der Funktionsbeschreibung dargestellt werden. Strukturell stellen Schnittstellen Interaktionspunkte mit der Umgebung dar. D.h. über Schnittstellen kann ein System Signale mit der Umgebung oder mit anderen Systemen austauschen. Je nach Richtung des Austauschs werden die Signale als die Eingangs- oder als die Ausgangsgrößen des Systems kategorisiert. Betrachtet man die Schnittstellen eines Systems, können diese entweder als rein funktionale Schnittstelle strukturiert sein (nur Eingangs- bzw. Ausgangssignale) (siehe Abbildung 11a) oder als Kopplung durch gemeinsame Systemelemente (siehe Abbildung 11b).

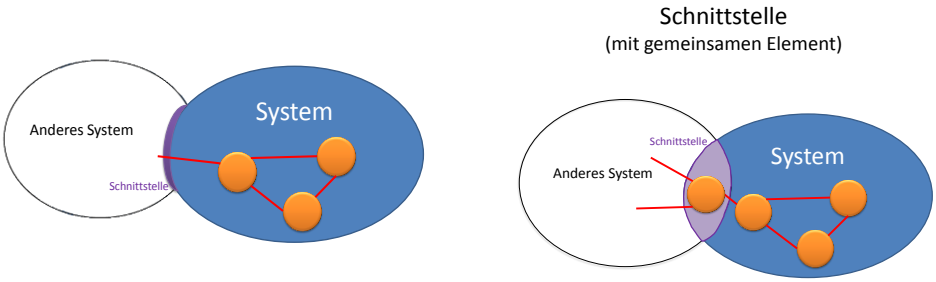


**Abbildung 11 – Schnittstelle mit gemeinsamen Element**

Ein Beispiel für die Kopplung von Systemen über ein gemeinsames Systemelement ist die doppelte Bedeutung von Feldbusanschaltungen für automatisierungstechnische Geräte (AT-Geräte). Solche Feldbusanschaltungen können sowohl als Bestandteil des Systems ‚Feldgerät' betrachtet werden als auch als Bestandteil des Systems ‚Feldbus'.

Eine Schnittstelle kann charakterisiert werden durch:

- Struktur
- Verwendete Signale
- Beziehung der Signale untereinander
- Charakteristiken der Signale (siehe 2.3.2.6)
- Richtung

In der Informatik (im Bereich der objektorientierten Architektur) werden Interfaces als Zusammenstellungen (*sets*) von Methoden bezeichnet [HM03]. – Ein Methodenaufruf kann als Steuersignal verstanden werden. Dabei entsprechen die Argumente der Methode dann Datensignalen, deren Werte bei einem bestimmten Wert des Steuersignals (beim Aufruf der Methode) übernommen werden.

Broy und Stølen beschreiben in [BS01] das syntaktische Interface eines Systems mit:

$$(I \triangleright O)$$

Nach Broy und Stølen steht I für die Menge aller Eingangskanäle und O für die Menge aller Ausgangskanäle, jeweils mit der Menge der möglichen Nachrichten (*messages*) auf diesen Kanälen. 'Kanäle' beschreiben dabei Signale, jeweils mit der Menge der möglichen Signalwerte (Nachrichten). Die durch Broy eingeführte Beschreibung des syntaktischen Interfaces entspricht nach automatisierungstechnischem Verständnis also einer Beschreibung der Systemfunktion mit ihren Eingangs- und Ausgangsgrößen.

In dieser Arbeit wird folgende Definition für Schnittstelle verwendet:

*Eine Schnittstelle ist der Punkt der Kopplung von einem Systemelement anderen Systemelementen. Sie wird charakterisiert durch die ihr zugeordneten Signale und ihren Signalcharakteristiken.*

Die Verwendung eines Port gibt die Möglichkeit mehrere Schnittstellen zusammenzufassen. UML2 beschreibt einen Port als Interaktionspunkt zwischen einer Klasse und seiner Umgebung. Die Schnittstellen, die mit einem Port assoziiert sind, definieren, welche Art von Interaktionen über den Port ausgeübt werden können [OMG11a, S. 186f.] .

Eine mögliche Verwendung von Port ist die Beschreibung einer Schnittstelle, die Definitionen auf verschiedenen Ebenen zusammenfasst, beispielsweise der LPT-Port eines Computers wird definiert durch ein mechanisches Interface (DB-25), elektrischem Interface (IEEE 1284 Electrical Interface) und logischem Interface (Protokoll).

#### *2.3.2.4 Kardinalität*

Kardinalität beschreibt die Fähigkeit einer funktionalen Verbindung mehrere Teilnehmer zu verbinden. Funktionale Verbindungen zwischen Systemelementen können zweiseitig (zwischen zwei Systemelementen) oder mehrseitig (zwischen mehreren Systemelementen) definiert werden. Diese Verbindungsarten müssen durch entsprechende Schnittstellen unterstützt werden:

- Bilaterale Schnittstellen unterstützen nur eine zweiseitige Verbindung (z.B. RS-232)
- Multilaterale Schnittstellen unterstützen Verbindungen zwischen mehreren Systemelementen (z.B. RS-485 oder Feldbus-Schnittstellen)

#### *2.3.2.5 Exklusivität*

Wenn Systemelemente miteinander durch bilaterale Schnittstellen verbunden sind, dann ist die Schnittstelle des Elements jeweils ‚besetzt' und kann nicht dazu genutzt werden, um weitere Verbindungen aufzubauen. Es handelt sich dann um eine exklusive Verbindung, die das Systemelement mit nur jeweils einem anderen Systemelement herstellen kann.

Wenn Systemelemente miteinander durch multilaterale Schnittstellen verbunden sind, dann können diese Schnittstellen für weitere Verbindungen genutzt werden. Beispiele für solche Schnittstellen sind Feldbusschnittstellen, welche es erlauben, dass ein Gerät mehrere Kommunikationsverbindungen aufbauen kann. Diese Verbindung repräsentiert dann eine gemeinsame Beziehung.

#### *2.3.2.6 Signal*

Die Interaktion von Systemelementen erfolgt mittels Signalen. D.h. über die funktionalen Verbindungen werden Signale ausgetauscht, die jeweils als Ausgangsgrößen (des Senders) bzw. als Eingangsgrößen (des Empfängers) dienen.

Beispielsweise sagt Frey in [FB08, S. 1f.]:

„Unter Signal verstehen wir allgemein eine abstrakte Beschreibung einer veränderlichen Größe. Die unabhängige Variable ist dabei in den meisten Fällen die Zeit, d.h. das Signal beschreibt den *zeitlichen* Verlauf der Größe. Wir unterscheiden zwischen einer *kontinuierlichen* und *diskreten* (diskontinuierlichen) Zeitvariable. [...} Zeitkontinuierliche Signale werden mathematisch durch Funktionen[5], zeitdiskrete Signale durch Folgen beschrieben."

Die deutsche Ausgabe des IEV beschreibt Signal als ein "physikalisches Phänomen, dessen Vorhandensein, Abwesenheit oder Änderung als Darstellung von Information betrachtet wird" [DKE12b], bzw. als "physikalische Größe, bei der ein oder mehrere Parameter Information über eine oder mehrere variable Größen tragen" [DKE12a].

Charakteristiken von Signalen sind:

- Zeitlicher Verlauf (kontinuierlich, diskontinuierlich)
- Signalträger (physikalische Größe, auf welche die Information abgebildet wurde)
- Wertevorrat (analog, diskret, binär)
- Wertebereich (Grenzen der erlaubten Werte)

Eine funktionale Verbindung zwischen Systemelementen kann durch ein oder mehrere Signale realisiert werden. Mehrere Signale können sowohl unabhängig voneinander übertragen werden als auch über ein Protokoll miteinander verknüpft sein. Wenn Signale mit einem definierten Wertevorrat in einer definierten zeitlichen Abfolge auftreten, spricht man von einem Protokoll [Kön12]. Ein Beispiel für solch eine Signalverknüpfung ist die Verknüpfung zwischen einem Datensignal (mit großem Wertevorrat) und einem Steuersignal (mit sehr kleinem Wertevorrat), bei dem die Gültigkeit des Datensignals vom Steuersignal signalisiert wird. Dieses Konzept wird z.B. bei der PC-Parallel-Schnittstelle verwendet [IEE00] oder bei Funktionsblöcken mit Daten-Eingang und zugeordnetem Event-Eingang [IEC03, S. 28f.].

Broy und Stølen beschreiben in [BS01] die Interaktion von Systemelementen über Messages. Diese Messages können als diskontinuierliche, diskrete Signale mit einem definierten Wertevorrat aufgefasst werden.

Signale werden über Links übertragen, die Implementierungen der funktionalen Verbindungen darstellen. Ein Systemelement muss jeweils ein entsprechender

[5] ‚Funktion' steht hier für mathematische Funktion. (Im Gegensatz zu Systemfunktion.)

Anknüpfungspunkt (Endpunkte) für diesen Link bereitstellen. Diese Endpunkte werden im Allgemeinen als Schnittstelle bezeichnet (siehe 2.3.2.3).

### 2.3.3 Verwendung verschiedener Systemmodelle

Von einem technischen System kann es verschiedene Modelle geben, die jeweils andere Eigenschaften des Systems darstellen (für eine Übersicht der verschiedenen Modelle siehe auch [Vog03] und [Sch99]). Dies ist zum Teil der historischen Entwicklung von unterschiedlichen Gewerken geschuldet. – Jedes Gewerk nutzt eigene Modelle (siehe auch [Müh12, S. 89f.] ). Bei der gegenwärtigen Entwicklung der Produktion zur stärkeren Arbeitsteilung und Spezialisierung kann erwartet werden, dass dieser Trend weiter anhält.

Die Nutzung verschiedener Modelle parallel zueinander hat den großen Nachteil, dass die Modelle oft disjunkt und nicht aufeinander abgestimmt sind [Müh12].

Vorteile der Nutzung verschiedener Modelle sind die Unterstützung von Arbeitsteilung und Spezialisierung auf verschiedene Fachgebiete sowie die Konzentration der Modelldarstellung auf wichtige Information jeweils für ein Fachgebiet.

[Müh12] untersucht wie verschiedene Modelle eines technischen Systems miteinander in Beziehung gesetzt werden können. Dabei werden bei verschiedenen Anwendungsfällen unterschiedliche Beziehungstypen zwischen den Modellelementen berücksichtigt (siehe Tabelle 3).

**Tabelle 3 –Beziehungstypen für Modellintegration basierend auf [Müh12, S. 83f.]**

| Anwendungsfall / Beziehungstyp | Combining | Relating | Matching | Mapping | Aligning | Integration | Merging | Mediation | Translating | Transforming | Häufigkeit Verwendung |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Assoziation | x | | | | | | | | | | 1 |
| typisierte Assoziation | x | x | | | | x | x | | | | 4 |
| Komplementärbeziehung | x | x | x | | | x | x | | | | 5 |
| Äquivalenz | x | x | x | x | x | x | x | | x | | 8 |
| Teiläquivalenz | x | x | | | x | | | | | | 3 |
| Aggregation | x | x | | | | x | x | | | | 4 |
| Komposition | x | x | | | | x | x | | | | 4 |
| Generalisierung | x | x | | | | | | x | | | 3 |
| Realisierung | x | x | x | | | | | x | | | 4 |
| Abhängigkeit | x | x | | | | | | | | | 2 |
| Regel (Implikation) | x | x | | | | x | x | | | x | 5 |
| Formel | x | x | | | | x | x | | | x | 5 |

Wenn also verschiedene Modelle eines Systems miteinander in Beziehung gesetzt werden, sind die Beziehungen der Elemente dieser Modelle zueinander wesentlich. Dabei ist

„Äquivalenz“ eine der am häufigsten genutzten Beziehungen der Modellelemente der verschiedenen Modelle. Andere Beziehungen werden weniger häufig genannt, sind aber beim Engineering ebenfalls von Bedeutung wie z.B. „Generalisierung“ und „Realisierung“ (siehe auch 2.3.1.2.1). Dementsprechend ist es sinnvoll im Verlauf des System-Engineering (z.B. bei der Ableitung eines detaillierteren Modells aus einem abstrakteren Modell) diese Beziehungen zwischen den Modellelementen unterschiedlicher Modelle zu dokumentieren.

### 2.3.4 Systeme von Systemen

In der Systemtheorie werden „*system of systems*“ beschrieben, bei denen verschiedenartige Systeme zu einem großen System integriert werden.

Hitchins definiert in [Hit13] ein System von Systemen wie folgt:

„A system of systems (SoS) is an open set of complementary, interacting systems with properties, capabilities and behaviours of the whole SoS emerging both from the systems and from their interactions to synthesise a whole operating with optimum effectiveness in its operational environment.“

Vergleicht man diese Definition mit der Definition für System (siehe S.8), fällt auf, dass die Definitionen sich ähneln, es wurde nur ‚Systemelement‘ durch ‚System‘ ersetzt.

Das heißt mit dem Begriff „*system of systems*“ wird die Möglichkeit betont, dass Systemen wiederum zu Systemen integriert werden.

Bei der Integration eines Systems in ein übergeordnetes System, übernimmt das System nun in Bezug auf die anderen Elemente des übergeordneten Systems die Rolle „Systemelement“ und alle oben beschriebenen Eigenschaften von Systemelementen treffen auf das untergeordnete System zu.

Wie in Abschnitt 2.3.2.3 beschrieben, stellen Systemelemente die Schnittstellen zur Verfügung um Beziehungen zu anderen Systemelementen aufzubauen. Dabei hängt die Anzahl und die Art der zur Verfügung gestellten Schnittstellen davon ab, welche Anwendungsfälle das Systemelement unterstützen soll.

Wird das Systemelement in einem System eingesetzt, so werden typischerweise die Schnittstellen des Systemelements genutzt. Dabei gelten die Exklusivitäts-Regeln entsprechend Abschnitt 2.3.2.5. Durch die Nutzung von bilateralen Schnittstellen zur Integration stehen diese Schnittstellen nicht mehr zur Verfügung. Multilaterale Schnittstellen und nicht für die Integration genutzte Schnittstellen stehen jedoch weiterhin zur Verfügung. Diese weiterhin verfügbaren Schnittstellen der Systemelemente können als Schnittstellen des Systems genutzt werden.

Zum Beispiel hat ein elektrischer Antrieb eine elektrische Schnittstelle (zur Energieversorgung) und eine mechanische Schnittstelle(zur Energieabgabe). Eine

Kreiselpumpe hat eine mechanische Schnittstelle (zur Energieaufnahme), einen Flüssigkeitseinlass und ein Flüssigkeitsauslass. Wenn die beiden Elemente miteinander zu dem System „Elektropumpe“ verbunden werden, dann werden die mechanischen Schnittstellen jeweils besetzt und stehen nicht mehr zur Verfügung. Die elektrische Schnittstelle des Antriebs sowie die Schnittstellen „Flüssigkeitseinlass“ und „Flüssigkeitsauslass“ der Kreiselpumpe sind jedoch weiterhin verfügbar und können als Schnittstellen des übergeordneten Systems „Elektropumpe“ betrachtet werden.

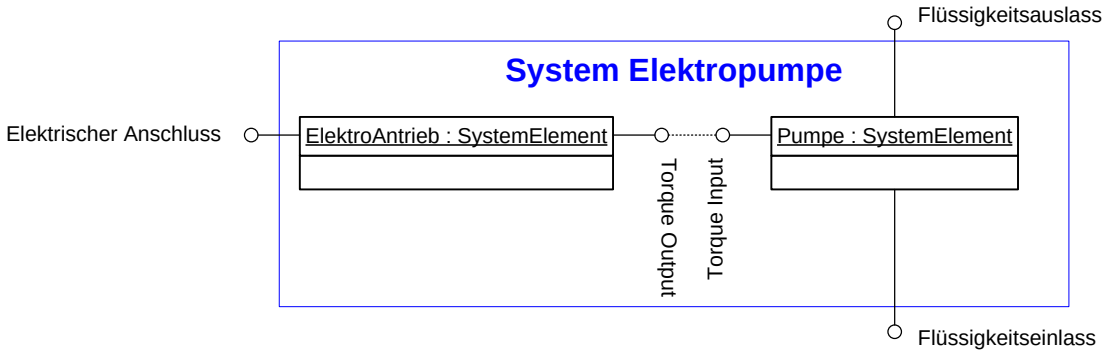


**Abbildung 12 – System „Elektropumpe“**

### 2.3.5 Eigenschaften von Systemen

Merkmale eines Systems können zumindest teilweise durch Kombination von Merkmalen der Systemelemente hergeleitet werden. Beispiele für solche herleitbare Merkmale sind:

- Verfügbarkeit
- Durchlaufzeit

Betrachtet man herleitbare Systemmerkmale in Zusammenhang mit den verschiedenen Systemstrukturen, so findet man unterschiedliche Regeln zur Herleitung der System-merkmale.

**Tabelle 4 – Herleitung von System-Merkmalen in Abhängigkeit von der Systemstruktur**

| Systemstruktur | Merkmal | Herleitungs-Regel |
|---|---|---|
| Reihe | Verfügbarkeit: p | $h(p) = \prod_{i=1}^{n} p_i$ |
| | Durchlaufzeit: t | $t_{sys} = \sum_{i=1}^{n} t_i$ |
| Parallel | Verfügbarkeit: p | $h(p) = 1 - \prod_{i=1}^{n} (1 - p_i)$ |
| | Durchlaufzeit: t | $t_{sys} = \min(t_i)$ |

D.h. bei der Herleitung von Systemmerkmalen aus den Merkmalen der Systemelemente müssen die funktionalen Verbindungen der Systemelemente berücksichtigt werden. Die Struktur des funktionalen Pfads hat zum Beispiel einen wesentlichen Einfluss auf die Verfügbarkeit oder Durchlaufzeit (Ausführungszeit) einer Funktion.

Nicht alle Systemmerkmale können allein aus der Kombination von Eigenschaften der Systemelemente hergeleitet werden, zum Beispiel ist die Flugfähigkeit von Sperrholz und von Leinen jeweils vernachlässigbar, aber das System ‚Flügel' bestehend aus Sperrholz-Spanten und Leinen-Bespannung ist durchaus flugfähig. Diese nicht aus den Eigenschaften von den Systemelementen herleitbaren System-Eigenschaften können als ‚Emergente' Eigenschaften bezeichnet werden.

Schnieder und Schnieder unterscheiden in [SS10] zwischen elementaren Eigenschaften des Systems (die auf Eigenschaften der Systemelemente basieren) und emergenten Eigenschaften des Systems (die aufgrund der Systemstruktur) entstehen:

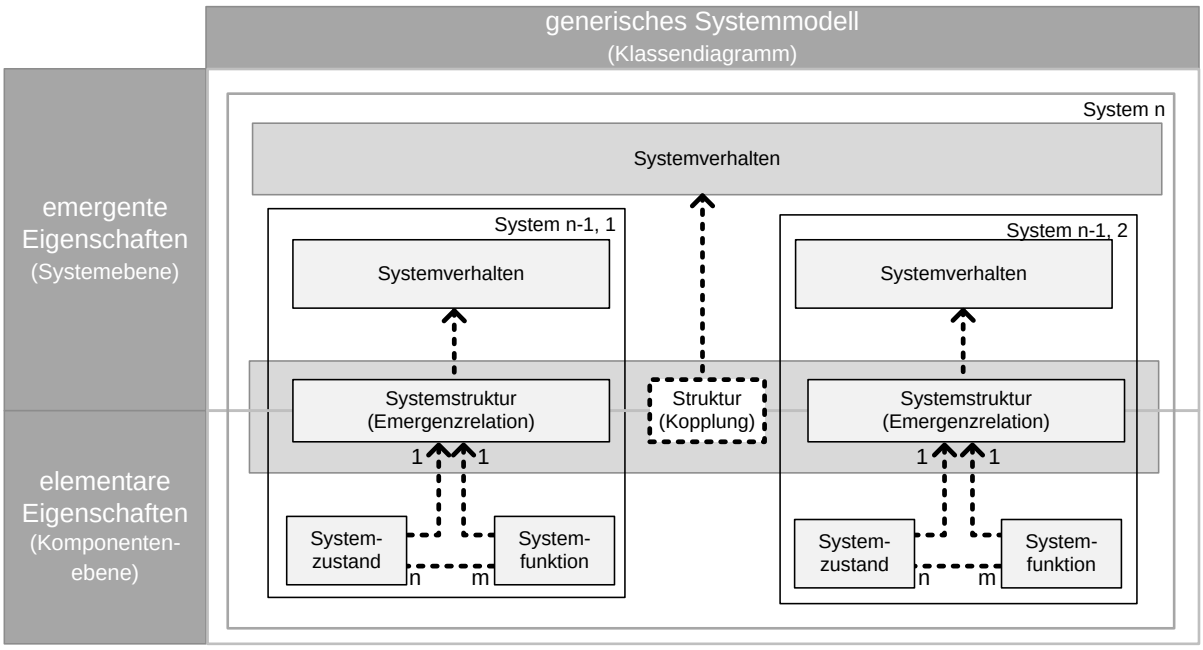


**Abbildung 13 – Emergenzrelation [SS10, S. 451f.]**

Emergente Eigenschaften eines Systems sind danach Eigenschaften, welche durch die jeweilige Struktur des Systems bedingt werden. Struktur bezieht sich dabei nicht nur auf die funktionalen Verbindungen zwischen den Elementen, sondern auch auf die Betrachtung der Systemelemente als untergeordnete Systeme (siehe Abbildung 13). Nach dieser Definition ist die Verfügbarkeit auch eine emergente System-Eigenschaft, obwohl sie aus der Verfügbarkeit der Systemelemente hergeleitet werden kann [SS10].

Generell wird Emergenz als Erscheinung betrachtet, bei der neue, „überraschende" Eigenschaften (Struktur, Verhaltensmuster, Properties) von Systemen auftreten. Diese

Betrachtungen erfolgen oft im Zusammenhang mit selbstorganisierenden, komplexen Systemen [Gol99]. Hitchins beschreibt dass Emergenz von den Interaktionen zwischen den Systemelementen stammt, wobei beide (sowohl die Interaktionen als auch die Systemelemente) Eigenschaften des Gesamtsystems bestimmen. Die Herausbildung emergenter Eigenschaften wird als mögliches Ziel beim Systementwurf beschrieben [Hit07, S. 124f.]

Betrachtet man diese Definition von Emergenz in Zusammenhang mit gezielt entwickelten, technischen Systemen, dann kann „überraschend“ nicht heißen, dass das System unerwartete Eigenschaften hat, sondern eher, dass das System neue Eigenschaften aufweist, welche nicht bei den Systemelementen allein beobachtet werden können. Basierend darauf werden in dieser Arbeit folgende Definitionen verwendet:

*Elementare Systemmerkmale sind Merkmale des Systems, welche bereits als Merkmale der Systemelemente ausgeprägt sind und deren Ausprägung im System sich aus den Elementmerkmalen und der Systemstruktur herleiten lassen.*

*Emergente Systemmerkmale sind Merkmale des Systems, welche nicht bei den Systemelementen ausgeprägt sind und deren Ausprägung im System auf eine spezifische Systemstruktur zurückzuführen ist.*

Nach dieser Definition ist Verfügbarkeit eher als elementare Eigenschaft einzuordnen. Aber auch für emergente Systemmerkmale sollte bei Anwendung elementarer Ingenieurkunst im Entwurf technischer nicht-komplexer Systeme der „Überraschungsfaktor“ gering sein.

### 2.3.6 Komplexität

Wissenschaftler verschiedenster Fachgebiete (z.B. Physik, Biologie, System-Engineering) beschäftigen sich mit dem Thema Komplexität, deswegen gibt es unterschiedlichste Definitionen von Komplexität und verschiedene Komplexitätsmetriken.

Generell werden drei Kategorien von Systemen betrachtet:

- Komplexe Systeme
- Komplizierte Systeme
- Einfache Systeme

Als komplexe Systeme werden im Allgemeinen Systeme bezeichnet, die eine nicht überschaubare Struktur haben, deren Verhalten nicht erklärbar, unvorhersagbar ist,

deren Verhalten sich ändert, das sich einer Analyse durch Reduktion (Dekomposition) entzieht bzw. deren Funktionen nicht analysierbar sind.

Nach [Mit09, S. 12f.] haben komplexe Systeme folgende gemeinsame Eigenschaften:

- Komplexes kollektives Verhalten:
  Die Systeme bestehen aus großen Netzwerken von individuellen Komponenten (jede mit einem einfachen Verhalten) und das kollektive Verhalten vieler Komponenten zusammen resultieren in einem komplexen Verhalten (schwer vorherzusagen, wechselndes Verhaltensmuster) – klassisches Beispiel für solche Systeme sind Ameisenkolonien
- Signal- und Informationsverarbeitung
  Die Systems erzeugen und verarbeiten Informationen und Signale von internen und externen Quellen
- Adaption
  Systeme ändern das Verhalten (z.B. durch Lernen oder evolutionäre Prozesse)

Mitchell selbst relativiert den letzten Punkt, weil [Mit09] auf adaptive Systeme fokussiert. Es kann also auch komplexe Systeme geben, die nicht adaptiv sind.

Es gibt einen Trend nach dem zwischen komplexen und komplizierten Systemen differenziert werden sollte:

- Komplizierte System sind Systeme mit einer schwer überschaubaren Struktur, aber durch Dekomposition ist es möglich aus dem Verhalten der Systemelemente das Gesamtverhalten des Systems abzuleiten (Reduktionismus).
- Komplexe Systeme sind Systeme deren Verhalten auch nicht durch Dekomposition in die Systemelemente erklärbar ist (Holismus). – Zum Beispiel ist der Aufbau eines menschlichen Gehirns im Allgemeinen bekannt und auch das Verhalten der Systemelemente (Neuronen) ist gut verstanden. Trotzdem ist es immer noch unklar wie sich in dieser Struktur Bewusstsein herausbilden kann.

Eine Definition für ein einfaches System lässt sich nicht finden. Offensichtlich sind alle nicht-komplexen und nicht-komplizierten Systeme wohl als einfache Systeme zu betrachten. Übliche Merkmale für einfache Systeme sind eine geringe Anzahl von Systemelementen und eine geringe Anzahl von Elementbeziehungen.

#### *2.3.6.1 Komplexität von Funktionen*

Für jede Funktion kann die Komplexität der Funktion bewertet werden. Dabei muss Komplexität nicht als absolute Metrik verstanden werden, sondern kann als relative Metrik für Funktionen betrachtet werden. D.h. man kann Funktionen anhand dieser Metrik vergleichbar machen.

Betrachtet man die Komplexität von Funktionen, haben Elementarfunktionen eine niedrige Komplexität. Die Komplexität von strukturierten Funktionen (z.B. Teilfunktionen und Gesamtfunktionen) ergibt sich aus der Struktur dieser Funktionen (z.B. nach [MB89, CRK08, Fre02]).

In [HSD13] und [Had12] wird dargestellt, wie die Komplexität von Funktionen anhand ihres funktionalen Pfads beurteilt werden kann.

## 2.4 Zusammenfassung

In diesem Kapitel wurden ein grundsätzliches Merkmalmodell und ein grundsätzliches Systemmodell eingeführt, das Gültigkeit nicht nur für technische Systeme hat, sondern prinzipiell auf alle Arten von Systemen angewendet werden kann. Basierend auf diesem grundsätzlichen Systemmodell wurde auf Besonderheiten von technischen Systemen eingegangen. Diese Modelle bilden nun die Grundlage für die Diskussion des aktuellen Stands der Technik (siehe Abschnitt 1) und für die Entwicklung von Methoden zur Verwendung des Merkmalmodells für das Engineering von Systemen (siehe Abschnitt 4 und folgende).

Es ist dargestellt worden, dass Merkmale dazu dienen, Systeme und Systemelemente zu beschreiben und dass Systeme durch Funktion, Verhalten und Struktur gekennzeichnet sind.

# 3 Stand der Wissenschaft und Technik

## 3.1 Zielstellung

Nachdem die grundsätzlichen Modelle für Merkmale und Systeme definiert wurden, soll in diesem Kapitel der aktuelle Stand der Wissenschaft und Technik dargestellt werden. Ziel dieser Arbeit ist es das Engineering von Systemen zu unterstützen. Deswegen werden ausgehend von dem zu unterstützenden Engineering-Prozess aktuelle Modelle für Merkmale und Systeme dargestellt und diskutiert. CAEX wird als Datenmodell für Systeme ausführlicher erörtert.

## 3.2 Systemmodelle in Wissenschaft und Technik

### 3.2.1 Systeme in der Automatisierungstechnik

In der Automatisierungstechnik werden Produktionssysteme mit automatisierungstechnischen Geräten (z.B. Sensoren, Aktuatoren, Steuerungsgeräte) ausgestattet um automatisierte Systeme zu erzeugen.

Dabei werden verschiedene Systemmodelle verwendet um unterschiedliche Teilsysteme zu beschreiben, zum Beispiel:

- System von Anlagenfunktionen
  betrachtet die Funktionen der Anlage mit dem Ziel die Funktionen in einem Ablauf (Produktionsablauf) anzuwenden
- Mechanisches System (Produktionssystem)
  betrachtet die (mechanischen) Komponenten des Systems
- Steuerungssystem
  betrachtet die automatisierungstechnischen Komponenten des System (E/A, Steuerung)
- Kommunikationssystem
  betrachtet die kommunikationstechnischen Komponenten des Systems (Feldgeräte, Steuerung, Kommunikationsinfrastruktur [Switch, Gateway, Repeater])
- Energieversorgungssystem
  betrachtet die Versorgung von Anlagenelementen mit Energie (z.B. elektrische Energie, pneumatischen Druck).

Diese verschiedenen Systemmodelle können integriert werden, um wiederum ein komplexes automatisiertes Produktionssystem vollständig zu beschreiben. Dieser Ansatz entspricht dem Ansatz des Systems von Systemen.

Das heißt, bei der Darstellung automatisierter Produktionssysteme werden verschiedene Beschreibungsmittel genutzt, um verschiedene Systemmodelle zu erstellen. Im Folgenden sollen verschiedene Beschreibungsmittel, welche bei der Beschreibung von Produktionssystemen verwendet werden, vorgestellt werden. Eine Wertung der Beschreibungsmittel erfolgt nicht. Ziel der Darstellung ist es den Stand der Technik und die Beziehung zwischen verschiedenen Beschreibungsmitteln zu dokumentieren. Für diese Arbeit wesentlich ist die Beziehung zwischen funktionalen Modellen und Strukturmodellen, deswegen wird hier auf eine Darstellung von Verhaltensmodellen verzichtet. Eine umfangreiche Übersicht auch in Bezug auf Verhaltensmodelle ist in [VV02] enthalten.

### 3.2.2 Funktionale Modelle

Ein funktionales Modell beschreibt die vom System bereitgestellten Funktionen aus der Sicht des Nutzers [RST09, TRS+10]. Mit wachsendem Detailierungsgrad können die Funktionen zu Unterfunktionen dekomponiert werden. Beispiele für Typen funktionaler Modelle sind Funktionsbäume (Darstellung der Abhängigkeit zwischen Funktionen und Unterfunktionen), Aktivitätsdiagramme und Fließdiagramme (d.h. Prozessbeschreibungen entsprechend 2.3.1.4.1). Diese unterschiedlichen Modelltypen werden durch verschiedene Beschreibungsmittel unterstützt.

Ein in der Automatisierungstechnik eingeführtes Beschreibungsmittel zur Darstellung von funktionalen Modellen ist die Formalisierte Prozessbeschreibung laut VDI/VDE 3682 [VV03]. Diese kann für die Beschreibung von funktionalen Modellen sowohl für die Prozessautomatisierung als auch für die Fertigungsautomatisierung eingesetzt werden [UGF09]. Deswegen wird die Formalisierte Prozessbeschreibung hier kurz eingeführt.

Wie in Abbildung 14 dargestellt, wird ein technischer Prozess mit seinen Eingangs- und Ausgangsgrößen (Produkt [P_1] und Energie [E_1]) an den Grenzen des Bilanzraums dargestellt. Das Element „Produkt“ wird zur Darstellung des Materialflusses genutzt. Der Bilanzraum wiederum ist in ein Netzwerk von Prozessoperatoren gegliedert, welche durch Fluss-Elemente (Produkt und Energie) verbunden sein können (im Bild ist nur ein Materialfluss dargestellt). Dabei wird jeweils dargestellt welche Formen von Produkt bzw. Energie zwischen den Prozessoperatoren transportiert wird. Einem Prozessoperator kann eine Technische Ressource zugeordnet werden, auf dem der Prozessoperator jeweils ausgeführt wird. Diese Zuordnung kann logischerweise erst erfolgen, nachdem ein Strukturmodell des Systems existiert und die verwendeten technischen Ressourcen definiert werden. D.h. durch die Zuordnung wird eine Beziehung zwischen Elementen des funktionalen Modells und Elementen des Strukturmodells definiert.

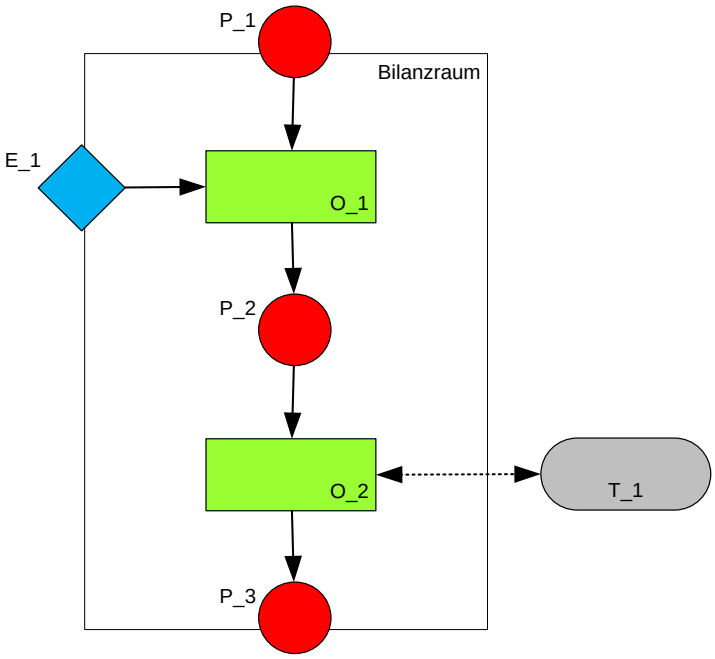


**Abbildung 14 – Formalisierte Prozessbeschreibung**

Ein Prozessoperator kann dekomponiert werden, in dem er wiederum als Bilanzraum dargestellt werden kann [VV03, UGF09].

Ulrich et.al haben in [UGF09] vorgeschlagen, die Formalisierte Prozessbeschreibung zur Darstellung von Informationen als Eingangs- bzw. Ausgangsgröße zu erweitern. Damit wäre es möglich, eine vollständige funktionale Beschreibung von Systemen bereitzustellen – siehe auch Patzaks Definition von Gesamtfunktion in 2.3.1.3.

[VV03] stellt dar, dass alle Elemente der Formalisierten Prozessbeschreibung als Objekte aufgefasst werden, deren Merkmale weitere Informationen zu dem beschriebenen Objekt (Produkt, Energie, Prozess, Ressource) bereitstellen. Der Merkmalbegriff laut [VV03] unterscheidet sich dabei von dem Merkmalbegriff laut [IEC09b] und entspricht laut [IEC09b] den Attributen einer Item-Klasse.

**Tabelle 5 - Bestandteil einer Merkmaldefinition laut VDI 3682**

| Verwendung | Attribute | Bestandteile der Attribute |
|---|---|---|
| Kennzeichnung | Eindeutiges Ident | |
| | Langname | |
| | Kurzname | |
| | Versionsnummer | |
| | Revisionsnummer | |
| | Referenzen | Produkte |
| | | Energien |
| | | Technische Ressource |
| | | Bilanzraum |
| Andere | Kategorie (Name) | |

| Verwendung | Attribute | Bestandteile der Attribute |
|---|---|---|
| Beschreibender Bestandteil | Wertfeststellungsverfahren | |
| | Repräsentativität | |
| | Sollwert | |
| | Gültigkeitsgrenzen | Gutbereich |
| | | Erlaubten Fehlbereich |
| | | Unerlaubten Fehlbereich |
| | Istwerte | |
| Beziehungsherstellender Bestandteil | Sicht | |
| | Modell | |
| | Vorschriften zur Relationsgenerierung | |

In einem Vergleich zwischen VDI/VDE 3682 und den PROLIST Merkmalleisten [Ahr10b, S. 33] werden die Unterschiede zwischen den Beschreibungsmodellen betont. Leider wird dabei die wesentliche Übereinstimmung nicht dargestellt: die Objekttypen der Formalisierten Prozessbeschreibung (Prozessoperator, Produkt, Energie, Bilanzraum und Technische Ressource) können als Klassen entsprechend PROLIST (siehe 3.4.8) betrachtet werden, die über die entsprechende Attribute identifiziert und beschrieben werden. Diesen Klassen können nun Merkmale zugeordnet werden, d.h. sie können als Merkmalträger genutzt werden (siehe 2.2).

In [BMQ+10] wird als Alternative zur Verwendung von Formalisierten Prozessbeschreibungen die Verwendung von Workflow-Modellen für die Beschreibung von fertigungstechnischen Anlagen gegenübergestellt.

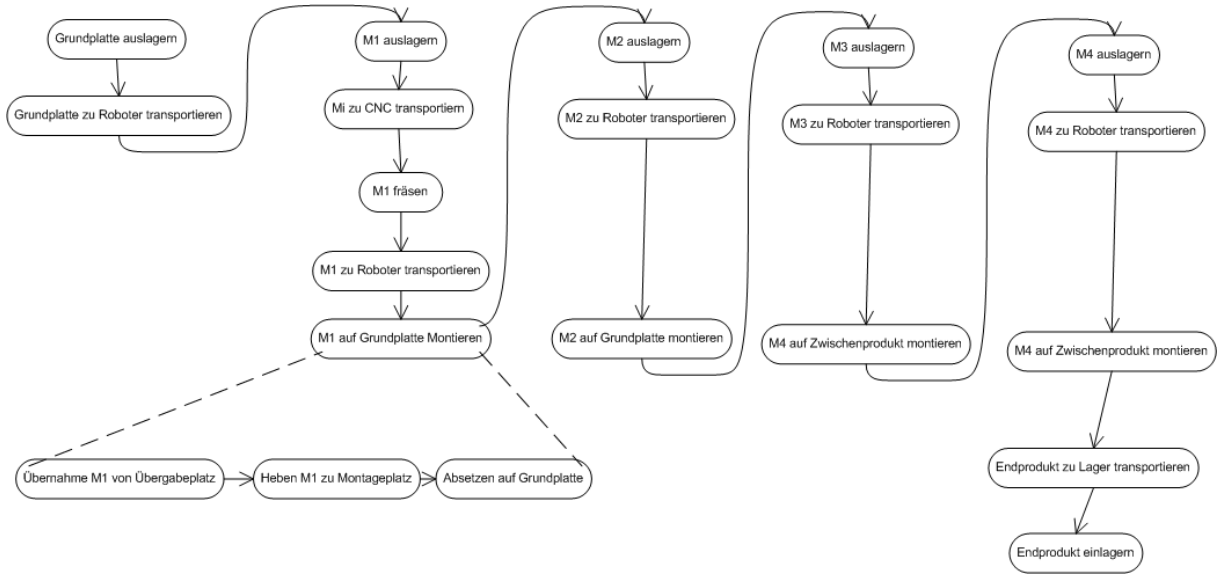


**Abbildung 15 – Aktivitätsdiagramm**

UML bzw. SysML nutzen als Mittel zur Beschreibung von Prozessen das in Abbildung 15 dargestellte Beschreibungsmittel „Aktivitätsdiagramm“ [Wei08]. Eine Aktion entspricht der Ausführung einer Systemfunktion, um die Reihenfolge der Ausführung darzustellen, werden Aktionen durch gerichtete Kanten verbunden. Diese Kanten können den Kontrollfluss oder

als Objektfluss die Übergabe von Objekten (z.B. Material, Energie, Information) darstellen und mit entsprechenden Objekten annotiert werden. Es ist möglich eine Aktivität weiter zu dekomponieren (z.B. in einem separaten Aktivitätsdiagramm).

Auch in einem Aktivitätsdiagramm kann eine Zuordnung der Aktivitäten zu Elementen eines Strukturmodells vorgenommen werden. Die Zuordnung einer Aktivität zu technischen Ressourcen kann durch Schwimmbahnen (engl.: *swim lanes*) dargestellt werden.

Die beschriebenen funktionalen Modelle können mit Information bezüglich der zwischen den Funktionen ausgetauschten Objekte (Informationen, Materialien, Energie) ausgestattet werden. Diese erlauben dann z.B. Rückschlüsse auf die Anforderungen an die Funktionen zu ziehen. So können z.B. bei einer Beschreibung des Fertigungsablaufs wie in Abbildung 15 bereits die Dimensionen des zu fertigenden Produkts berücksichtigt und als Anforderung an die Funktionen erfasst werden (z.B. für die maximalen Größen des Werkstücks).

### 3.2.3 Strukturmodelle

Strukturmodelle beschreiben die Struktur des Systems (siehe 2.3.1.2). Die Beschreibung kann mit unterschiedlichem Detaillierungsgrad erfolgen. Komponentenmodelle weisen geringen Detaillierungsgrad als Topologie-Modelle auf.

#### *3.2.3.1 Komponentenmodell*

Technische Systeme werden typischerweise in Komponenten gegliedert. Komponenten stellen dabei wiederverwendbare System-Elemente dar, die eine Funktion erbringen. Die in einem Komponentenmodell dargestellte Information entspricht der funktionalen Struktur entsprechend 2.3.1.2. Dabei werden typischerweise allgemeine Betriebsmittel-Typen referenziert (z.B. Fräse, Drehbank). Die Darstellung erfolgt dabei durch Blockdiagramme (auch bezeichnet als: Blockschaltbild, Strukturbild, Blockschema), in dem die Komponenten durch einfache geometrische Figuren (Rechtecke, Kreise) dargestellt werden und die funktionellen Beziehungen zwischen den Komponenten durch Kanten dargestellt werden (siehe Beispiel in Abbildung 16)[Kla68, S. 113f.].

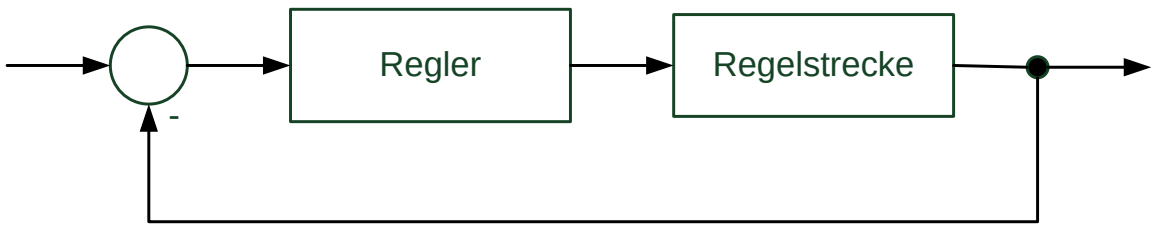


**Abbildung 16 – Beispiel für ein Blockdiagramm**

In UML wird eine Komponente als ein physisches Element (Hardware) oder als logisches Element (Software) betrachtet. Es wird davon ausgegangen, dass durch weitere Spezifikationen (Profile) spezifische Komponententypen und Regeln für diese Typen definiert werden. Wesentlicher Aspekt der komponentenbasierten Entwicklung ist die Wiederverwendbarkeit einer Komponente [OMG11a, S. 145f.]. Das UML Metamodell versteht eine Komponente als eine Klasse, die ein modulares Systemelement repräsentiert, das in unterschiedlichen austauschbaren Erscheinungsformen vorliegt. Das Verhalten einer Komponente wird über Schnittstellen definiert. Komponenten mit gleichen Interfaces sind austauschbar [OMG11a, S. 149f.].

Eine Komponente kann aus anderen Komponenten zusammengesetzt werden (Teil-Ganzes-Beziehung, kompositionelle Struktur nach 2.3.1.2). Dabei ist es möglich eine komplexe einzigartige Komponente aus verschiedenen wiederverwendbaren Komponenten zu erstellen und so das Konzept der Wiederverwendbarkeit zu unterstützen.

Ein Komponentenmodell kann aus einem funktionalen Modell eines Produktionssystems abgeleitet werden, indem z.B. für jede auszuführende Funktion des Systems eine entsprechende Komponente eingefügt wird und die Flussverbindungen des funktionalen Modells als funktionale Verbindungen der Komponenten nachgebildet werden. Dabei erfolgt ein Übergang von einer funktionsorientierten Darstellung zu einer strukturorientierten Darstellung des Systems. Die Zuordnung der Funktionen auf die Komponenten kann auch anders erfolgen, z.B. ist es ist möglich, dass eine Funktion auf mehrere Komponenten aufgeteilt wird (z.B. durch Unterfunktionen) bzw. dass eine Komponente mehrere Funktionen erbringt.

#### *3.2.3.2 Topologie-Modell und Netzwerkmodell*

Die Begriffe Topologie-Modell [FHP+11] und Netzwerkmodell [WV08] werden in dieser Arbeit als synonym betrachtet. Ein Topologie-Modell stellt das System aus Sicht der Kommunikation dar.

Das Netzwerkmodell eines Produktionssystems kann aus dem Komponentenmodell des Produktionssystems abgeleitet werden. Bei der Ableitung des Netzwerkmodells aus dem Komponentenmodell werden die abstrakten Komponenten durch konkrete Produkttypen (z.B. Geräte) ersetzt, definiert über welche Netzwerke die konkreten Produkttypen miteinander kommunizieren und zusätzliche Infrastrukturelemente (z.B. Switch, Router) hinzugefügt.

### 3.2.4 Modell- und Elementbeziehungen

Mühlhause hat in [Müh12] dargestellt, dass es verschiedenartige Beziehungen zwischen den Systemmodellen gibt. – Diese Beziehungen existieren sowohl für Systemmodelle als auch für die Systemelemente.

Beschreibungsmittel in der Automatisierungstechnik unterstützen ein unterschiedliches Niveau für die Darstellung von Beziehungen zwischen den Elementen eines Systems. Tabelle 6 bietet eine Übersicht über die Beziehungen in verschiedenen Beschreibungsmitteln.

**Tabelle 6 – Beziehungen in verschiedenen Beschreibungsmitteln**

| | **Beziehungen (Allgemein)** | **Bestandsbeziehungen** | **Funktionale Verbindungen** | **Generalisierung** |
|---|---|---|---|---|
| **UML Komponentendiagramm / Klassendiagramm** | Relationship, Association | Aggregation / Composition; Strukturierte Klasse | Associations zwischen Klassen (Interfaces) | Generalisation |
| **Entity-Relationship** | Relationship | Is-part-of | (Relationship mit definierter Bedeutung) | Is-a relationship |
| **CAEX** | <InternalLink><br><br>Äquivalenz: Mirror-Konzept | XML-Hierachy von <InternalElement > | <InternalLink> | XML-Typ-Vererbung, Referenzen auf <SystemUnitClass> und <RoleClass> |
| **P&ID** | Linie | Hierarchie von Diagrammen | Fließlinien, Steuer- und Signallinien | |
| **IEC 61987** | | LOP/Block-Hierarchie, Properties mit Klassenreferenzen | - | Klassenhierarchie (Superklasse, Subklasse), Polymorphismus für Klassenreferenzen |

Es kann auch Beziehungen zwischen Modellelementen unterschiedlicher Systemmodelle geben. UML definiert zum Beispiel eine <Realization>-Abhängigkeit (siehe Abbildung 7), mit der dargestellt werden kann, dass Elemente eines Modells die Elemente eines anderen Modells realisieren. Ein anderes Beispiel ist in der Formalisierten Prozessbeschreibung die Referenz von einem Prozessoperator auf die Technische Ressource, die den Prozessoperator realisiert. Der Zusammenhang zwischen den Technischen Ressourcen wird mit der Formalisierten Prozessbeschreibung nicht beschrieben, sondern muss in einem separaten Strukturmodell definiert werden. Durch die Referenz auf die Technische Ressource wird also ein Bezug zwischen Prozessbeschreibung und Strukturbeschreibung des Systems hergestellt. Generell können diese Beziehungen auf Ebene der Modellelemente als Basis zur Definition der Beziehungen zwischen den Systemmodellen genutzt werden (siehe [Müh12, S. 76]).

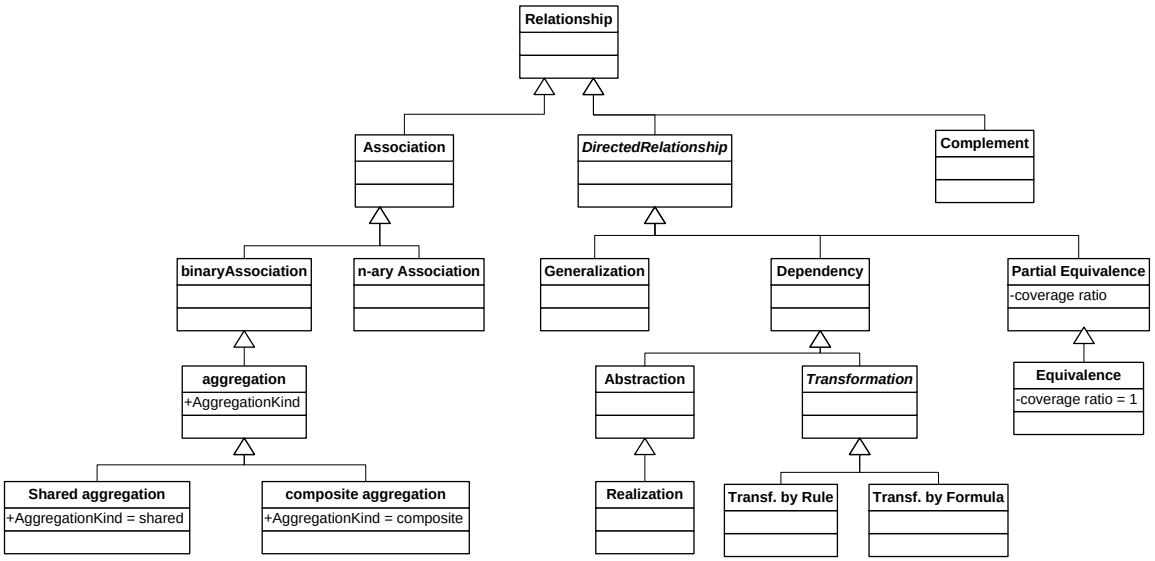


**Abbildung 17 – Modellbeziehungstypen (basierend auf [Müh12])**

### 3.2.5 Zusammenfassung

Für die Beschreibung der verschiedenen Aspekte von Systemen werden verschiedene Systemmodelle verwendet. Diese Systemmodelle beschreiben sowohl unterschiedliche Sichten auf das Produktionssystem (z.B. aus Sicht der Funktion: Funktionsmodell und aus Sicht der Struktur: Komponentenmodell) als auch unterschiedliche Detaillierungsstufen einer Sicht (z.B. abstraktes Komponentenmodell und detailliertes Komponentenmodell beschreiben die Systemstruktur mit unterschiedlichem Detail). Unterschiedliche Sichten können sowohl die gleichen Systemelemente als auch unterschiedliche Systemelemente beinhalten.

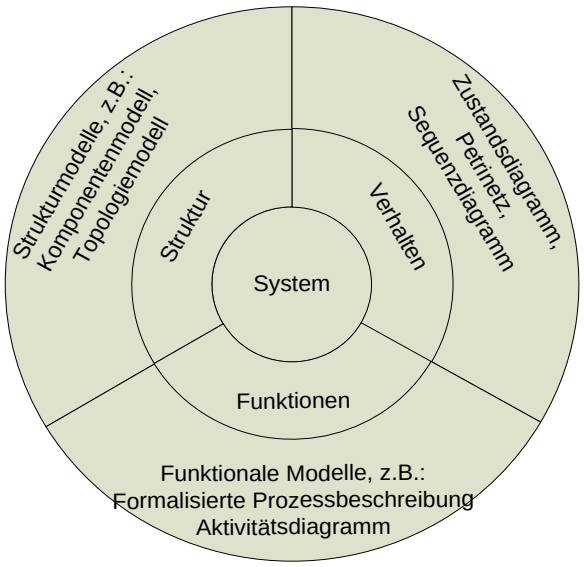


**Abbildung 18 – Beschreibungsmittel für die verschiedenen Systemaspekte**

Daraus ergibt sich ein Problem, weil der Zusammenhang zwischen den verschiedenen Sichten eine wichtige Information ist, die dokumentiert werden muss. Diese Dokumentation erfolgt sowohl in den jeweiligen Modellen (z.B. in der formalisierten Prozessbeschreibung durch Referenz auf die technische Ressource), durch Namenskonventionen für die verschiedenen Beschreibungsmittel (z.B. Bezeichnungen von Verhaltensdarstellungen referenzieren das jeweilige Systemelement) als auch durch Dokumentation außerhalb der Beschreibungsmittel.

## 3.3 CAEX Systembeschreibungssprache

Betrachtet man die Modelllandschaft für Beschreibung von Produktionssystemen, dann ragt die CAEX-Systembeschreibungssprache (im folgenden kurz „CAEX“) als grundlegendes Datenformat heraus, das allgemein als Austauschformat für die Beschreibung von Produktionssystemen akzeptiert wird. Aus diesem Grund wird in diesem Abschnitt CAEX diskutiert.

CAEX wird hier als Beispiel für Strukturbeschreibungssprachen in der Automatisierungstechnik eingeführt, weil es leicht erweiterbar ist und die in Kapitel 4 eingeführten Vorschläge zur Nutzung von Merkmalmodellen im Engineering basierend auf CAEX gut demonstriert werden können (siehe Kapitel 5). Prinzipiell können die diskutierten Vorschläge auch auf andere Beschreibungssprachen angewendet werden.

### 3.3.1 Einführung

CAEX [IEC08] wurde als ein Datenformat zum Austausch von Daten zwischen verschiedenen Engineering-Werkzeugen im Engineering-Prozess entwickelt (CAEX steht für „Computer Aided Engineering eXchange“). Als solches wurde es möglichst allgemein formuliert. Es ermöglicht verschiedenartige Systemmodelle zu repräsentieren, indem es ein Strukturmodell als Basis nutzt, das mit verschiedenen spezifischen Modellen (z.B. mit Rollenbibliotheken) erweitert werden kann. Beispielsweise diskutiert Helgermann anhand von AutomationML, wie CAEX-Elemente genutzt werden können, um verschiedene Modelle der Automatisierungstechnik darzustellen [Hel12].

Aufgrund dieses generellen Basis-Modells und aufgrund der Flexibilität des Modells erscheint CAEX als passendes Format zur Darstellung einer Integration des Systemmodells mit dem Merkmalmodell, deswegen wird CAEX in diesem Abschnitt diskutiert. Die Diskussion orientiert sich an den in Abschnitt 2.3 dargestellten Gesichtspunkten.

Die aktuell gültige und allgemein genutzte Version von CAEX ist CAEX v2.15. Gegenwärtig erfolgt eine Weiterentwicklung vom CAEX zu einer neuer Version CAEX v3.0 [Dra13, IEC14b].

AutomationML [IEC11a] wurde basierend auf CAEX zum Einsatz für die Entwicklung von fertigungstechnischen Systemen entwickelt. Dabei sind Erweiterungen und abweichende Semantiken definiert worden. Eine Übersicht über solche Änderungen und Erweiterungen in Vergleich zwischen den verschiedenen Versionen von CAEX und AutomationML wird in Tabelle 7 dargestellt. – Es kann davon ausgegangen werden, dass die Weiterentwicklungen in CAEX auch für AutomationML übernommen werden.

**Tabelle 7 – Vergleich von CAEX und AutomationML**

| Konzept | CAEX v2.15 | CAEX v3.0 | AutomationML |
|---|---|---|---|
| Referenz auf externe Dokumente | - <ExternalReference> verweist auf anderes CAEX-Dokument | - <ExternalReference> verweist auf anderes CAEX-Dokument | - <ExternalReference> verweist auf anderes CAEX-Dokument<br>- PLCopenXMLInterface verweist auf PLCOpen-Dokument<br>- COLLADAInterface verweist auf Collada-Dokument [Dra10, S. 64f.] |
| Verweis mittels refBaseSystemUnitPath auf <SystemUnitClass>-Element in <SystemUnitClassLib> | - "lebendige Vererbung", Änderungen an der Klassendefinition bewirken Änderungen an den referenzierenden Instanzen [Dra10, S. 54f.] | - Klassen dienen als "Kopiervorlage", beim Anlegen der Referenz werden alle Eigenschaften übernommen, spätere Änderungen an der Klassendefinition wirken sich nicht an den referenzierenden Instanzen aus, sondern müssen explizit neu übernommen werden [IEC14b, S. 73f.]. | - Klassen dienen als "Kopiervorlage", beim Anlegen der Referenz werden alle Eigenschaften übernommen, spätere Änderungen an der Klassendefinition wirken sich nicht an den referenzierenden Instanzen aus [Dra10, S. 54f.] |
| Unterstützte Rollen | - Aktuell unterstützte Rolle (Einsatzzweck) wird durch eine Referenz auf RoleClass definiert (RefBaseRoleClassPath als Pfad) [Dra10, S. 56f.] | - Aktuell unterstützte Rolle (Einsatzzweck) wird durch eine Referenz auf RoleClass definiert (RefBaseRoleClassPath als Pfad) [IEC14b, S. 73f.]. | - Aktuell unterstützte Rollen werden durch <supportedRoleClass>-Liste beschrieben [Dra10, S. 57f.] |
| Definition von Anforderungen | - Verweis auf eine RoleRequirements mit optionalem Attribut RefBaseRoleClassPath<br>- Mapping separat | - Verweis auf mehrere RoleRequirements mit optionalem Attribut RefBaseRoleClassPath<br>- Mapping integriert in RoleRequirements [IEC14b, S. 81f.]. | - |
| Allgemein genutzte Attribute | - | - AttributeLibrary zur Definition von Attribute-Klassen [IEC14b, S. 52f.]. | - |
| Datentypen | - | - Verwendung von 'string' Datentypen statt 'anyType' | - |
| Zusammenfassung mehrerer Schnittstellen | - (nicht vorgesehen) | - Unterstützung von hierarchischen Interface-Definitionen (nested Interface) [IEC14b, S. 63f.] | - Port-Konzept, zur Zusammenfassung von Interfaces [Dra10, S. 74] |
| Versionsinformation für abgeleitete Standards | - Kein abgeleiteter Standard vorgesehen | - Unterstützung für Versionsinformationen von abgeleiteten Standard (e.g. AutomationML) [IEC14b, S. 53f.] | - Versionsinformation zu AutomationML als Annotation im XML-Dokument |

| Konzept | CAEX v2.15 | CAEX v3.0 | AutomationML |
|---|---|---|---|
| | - ‚anonymer' Namensraum | - Definition eines Namensraums für CAEX [Dra13, S. 38f.] | - |
| Verwendung der Objekt-ID | - Optionale ID, Identfikation erfolgt über das Name-Attribut [IEC08, S. 41f.] | - ID (als GUID) ist obligatorisch für alle Instanzen (<InternalElement> und <ExternalInterface>) | - ID (als GUID) ist obligatorisch für <InternalElement> |
| Tracking | - | - Information über Quelle der Daten<br>- (generell für das Dokument und für Elemente des Modells) | - |
| Mirror-Konzept | - | - diverse Klarstellungen (siehe auch 3.3.4) | - |
| InternalLink | - Referenzen zu Enden des Links optional | - Referenzen zu Enden des Links obligatorisch | - |
| Zusätzliche Konzepte in AutomationML | - | - | - Facetten-Konzept, zur Beschreibung unterschiedlicher Sichten auf Systemelemente [Dra10, S. 77]<br>- Gruppen-Konzept, zur Gruppierung von Systemelementen [Dra10, S. 79f.]<br>- Ressource-Produkt-Prozess-Konzept [Dra10, S. 83] zur Überstützung des Produkt-Prozess-Ressource- Konzepts |

Im weiteren wird bei allen Aussagen zu CAEX in diesem Dokument davon ausgegangen, dass sie sowohl für CAEX auch für AutomationML gelten. Andernfalls wird AutomationML explizit genannt.

Ein CAEX-Dokument (nach CAEX v2.15) gliedert sich in vier Knotentypen: die <InstanceHierarchy>, welche eine Abbildung der betrachteten Anlage enthält; die <InterfaceClassLib>, welche Beschreibungen von Interfaces enthält; die <RoleClassLib>, welche Rollen als abstrakte Objekte definiert. Eine Rolle kann ein Element oder eine Funktion des referenzierenden Elements beschreiben. Der vierte Knotentyp ist die <SystemUnitClassLib>. Diese Bibliothek enthält einen Katalog der Strukturelementtypen, welche in der Anlage genutzt werden können. Alle diese Knotentypen können mehrfach definiert werden [GF08, S. 35f., IEC08] (siehe Abbildung 19).

Mit CAEX 3.0 kann ein weiterer Bibliothek-Typ bereitgestellt werden: <AttributeTypeLib>. Diese Bibliothek enthält Definitionen von Attribute-Typen, welche bei der Definition von Daten referenziert werden können. Attribute-Typen sind Klassen, die in der <AttributeTypeLib> voneinander abgeleitet werden [Dra13, IEC14b].

Viele CAEX-Elemente sind von dem Basis-Element <CAEXObject> abgeleitet. Ein entsprechendes Element kann über eine ID und muss mit einen Namen identifiziert werden.

Prinzipiell kann man die abgeleiteten Klassen als eindeutig identifizierbare ‚Konzepte' im Sinne von ISO/IEC 11179 verstehen.

Ein CAEX-Dokument kann weitere CAEX-Dokumente mittels <ExternalReference>-Elementen referenzieren. Dabei wird ein lokaler Datei-Verweis verwendet [IEC11a, S. 28f., IEC08, S. 59f.].

Für AutomationML wurde zusätzlich zur <ExternalReference> die Möglichkeit eingeführt OpenPLC-Dokumente und Collada-Dokumente in entsprechenden Interface-Definitionen zu referenzieren. Für diese Referenzen in Interface-Definitionen wurde die Verwendung einer URI spezifiziert [IEC11a, S. 28f.]. Die Verwendung von Interface-Definitionen zur Referenzierung auf externe Dokumente führt zu einer Mischung zweier Systembeschreibungen: üblicherweise wird mit Interface das technische System beschrieben. Wenn das Konstrukt „Interface" für Referenzen auf Dateien verwendet wird, wird jedoch das Engineering-System (bzw. seine Umgebung) beschrieben.

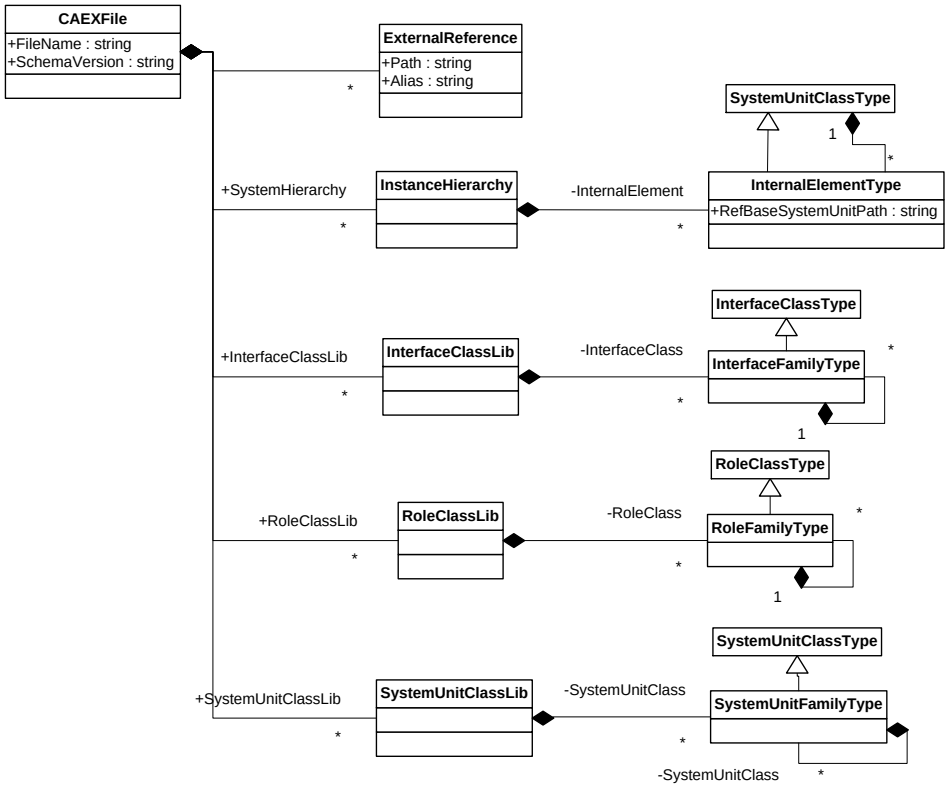


**Abbildung 19 – Inhalt eines CAEX-Files**

Alle Inhaltselemente eines CAEX-Dokumentes können hierarchisch strukturiert sein, wobei nur die Struktur der <InstanceHierarchy>-Elemente eine festgelegte semantische Bedeutung trägt. Die Strukturierung von <InterfaceClassLib>-, <RoleClassLib>- und <SystemUnitClassLib>-Elementen können eine anwendungsspezifische Bedeutung haben (z.B. wie durch die Namensgebung ‚Family' angedeutet zur Darstellung von

zusammengehörigen Mengen von Bibliothekselementen). Die hierarchische Struktur der <InstanceHierarchy>-Elemente wird zur Darstellung der Aufbaustruktur eines Systems genutzt.

### 3.3.2 Darstellung der Aufbaustruktur

Die Darstellung der Aufbaustruktur in CAEX basiert auf einer XML-Hierarchie und besteht aus dem Systemmodell (repräsentiert durch das <InstanceHierarchy>-Element) und dem Systemelement (repräsentiert durch <InternalElement>-Element).

Ein Systemmodell ist jeweils hierarchisch strukturiert, ein <InstanceHierarchy>-Element kann ein oder mehrere <InternalElement>-Elemente enthalten, welche wiederum <InternalElement>-Elemente enthalten können. D.h. Aggregations- und Kompositionsbeziehungen werden in CAEX durch die Hierarchie von Systemelementen (<InternalElement>) beschrieben [IEC08, S. 29f.].

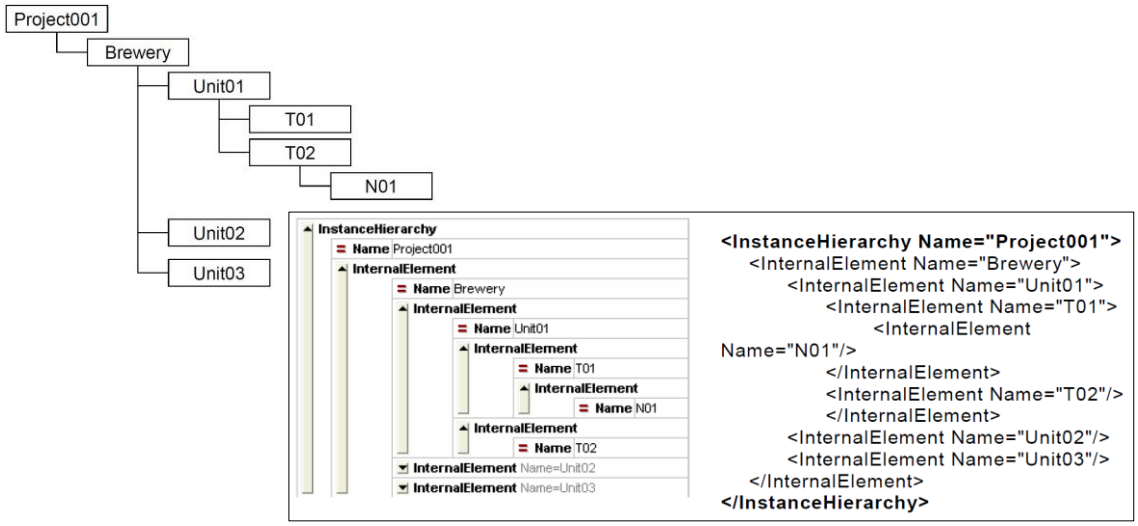


**Abbildung 20 – Darstellung der Aufbaustruktur eines Systems mit CAEX [IEC08, S. 53f.]**

Die Eigenschaften von Systemelementen (<RoleClassType>, <InterfaceClassType>, <SystemUnitClassType> und abgeleitete Instanzen) werden durch Attribute repräsentiert. Ein <AttributType> wird vom <CAEXObject> abgeleitet und kann Informationen über den Defaultwert, einen aktuellen Wert, Wertebereich, Maßeinheit, Datentyp und eine Referenz auf eine semantische Definition bereitstellen. In veröffentlichten Beispielen werden Attribute von Systemelementen oft nur mit Namen und Wert repräsentiert.

Ein CAEX-Dokument kann mehrere solcher Hierarchien darstellen [IEC08, S. 55f.].

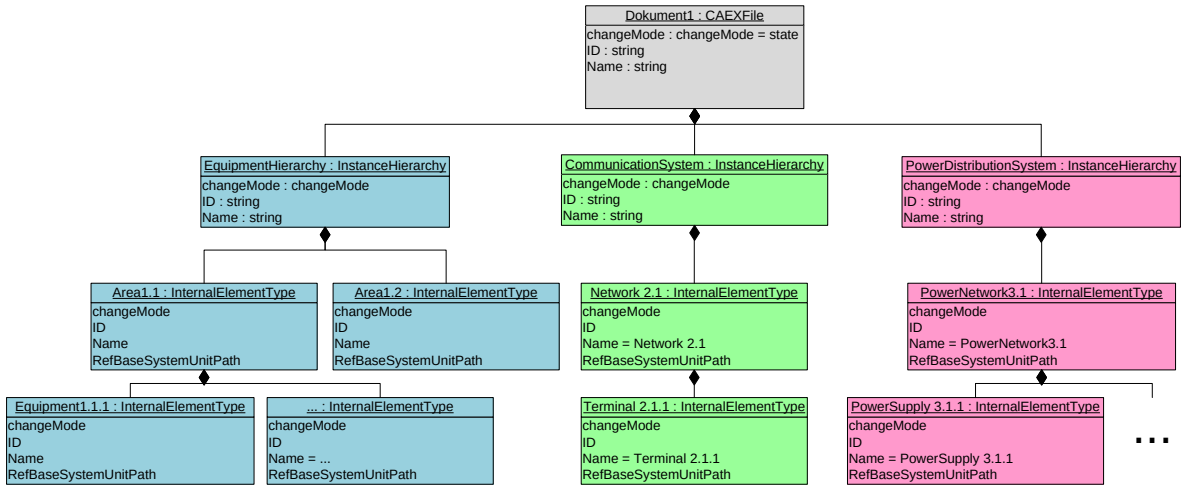


**Abbildung 21 – Beispiel für mehrere Hierarchien in einem CAEX-Dokument**

Diese Fähigkeit kann zur Darstellung mehrerer Modelle des gleichen Systems in einem Dokument genutzt werden (siehe Abbildung 21).

### 3.3.3 Darstellung funktionaler Verbindungen

Funktionale Verbindungen, welche in einem realen System als Verbindungen zwischen den Systemelementen existieren, werden in CAEX durch das Element <InternalLink> repräsentiert. Dabei unterstützen die <InternalLink>-Elemente nicht die Darstellung von Eigenschaften (z.B. Attribute), sie können also nur für die Darstellung des Vorhandenseins einer Verbindung genutzt werden. Eine Beschreibung der Verbindungseigenschaften ist somit nicht möglich, bzw. erfolgt implizit über Eigenschaften der verbundenen <InterfaceClassType>-Elemente [IEC08, S. 31f.] (siehe Abbildung 22).

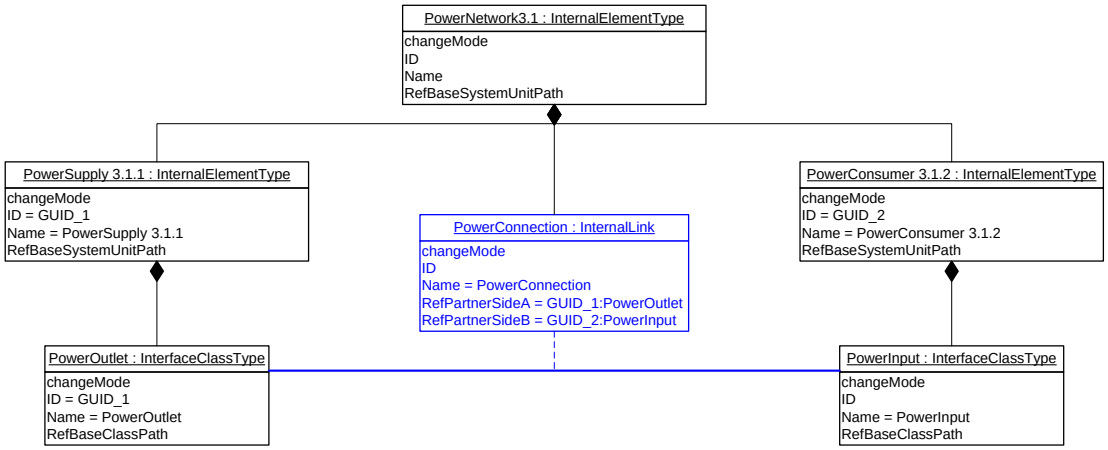


**Abbildung 22 – Beispiel für <InternalLink>-Element**

Die Darstellung von Verbindungen mit Eigenschaften ist jedoch durch die Verwendung von <InternalElement>-Elementen möglich. Die Verbindung wird dabei als separates Objekt mit eigenen Attributen, Rolle usw. betrachtet. Das entsprechende <InternalElement> muss dabei

mit einer entsprechenden <RoleClass> assoziiert werden, welche die entsprechenden Eigenschaften verwaltet. Die Beziehung zwischen dem Verbindungsobjekt und den Systemelementen wird wieder über <InternalLink>-Elemente dargestellt [IEC08, S. 31f.]. Diese Art der Darstellung ist hauptsächlich für physische Verbindungselemente vorgesehen (z.B. Kabel, Leitungen). Abbildung 23 zeigt ein entsprechendes Beispiel, bei dem aber auf die Darstellung der Attribute verzichtet wurde.

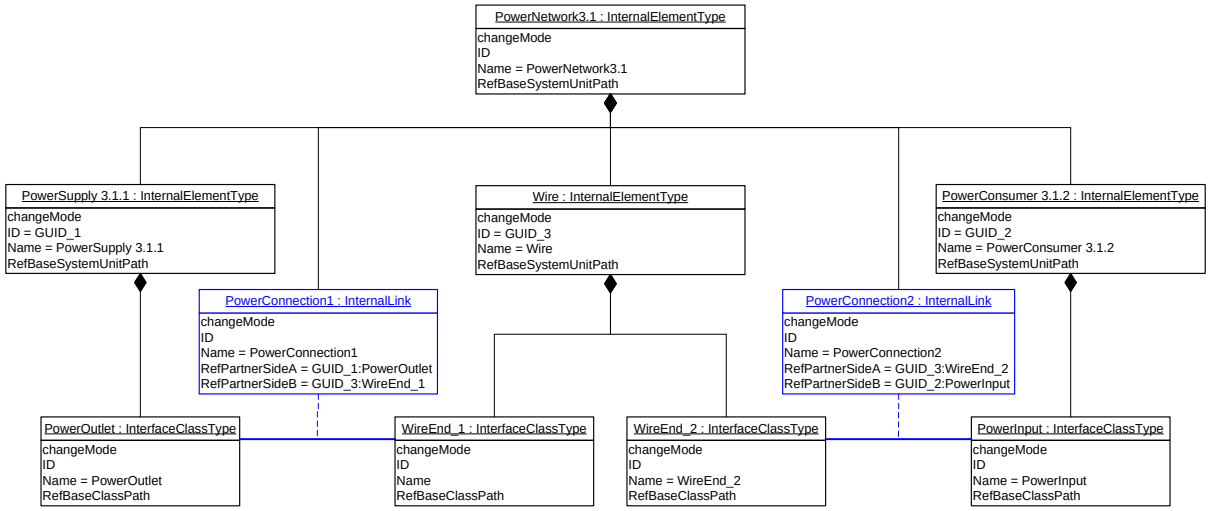


**Abbildung 23 – Beispiel für Verbindung mittels <InternalElement>**

Als Endpunkte eines <InternalLink> werden immer definierte Schnittstellen (<ExternalInterface>-Elemente) referenziert.

### 3.3.4 Darstellung von Verhältnis-Beziehungen

Untersucht man CAEX in Hinblick auf die in Abschnitt 2.3.1.3 dargestellten Elementbeziehungen, so ergibt sich die in Tabelle 8 dargestellte Übersicht.

**Tabelle 8 – Übersicht über Elementbeziehungen in CAEX**

| **Bezeichnung lt. [Dra10]** | **Vater-Kind-Relation** | **Vererbungsbeziehungen / Klassen-Instanz-Relation** | **Instanz-Instanz Relation** | |
|---|---|---|---|---|
| **Einordnung entspr. 2.3.1.3** | **Bestandsbeziehung** | **Generalisierung** | **Funktionale Verbindung** | **Äquivalenz-Verhältnis** |
| Rollenbibliothek (<RoleClassLib>) | | <RoleClass<br>RefBaseClass … ><br></RoleClass> | | |
| Interfacebibliothek (<InterfaceClassLib>) | | <InterfaceClass<br>RefBaseClass … ><br></InterfaceClass> | | |
| SystemUnit-Bibliothek (<SystemUnitClassLib>) | <SystemUnitClass><br><SystemUnitClass><br></SystemUnitClass><br></SystemUnitClass> | <SystemUnitClass><br><Name>Parent</Name><br></SystemUnitClass><br><SystemUnitClass<br>RefBaseClassPath = Parent<br>… ><br><Name>Child</Name><br></SystemUnitClass> | | |

| Bezeichnung lt. [Dra10] | Vater-Kind-Relation | Vererbungsbeziehungen / Klassen-Instanz-Relation | Instanz-Instanz Relation | |
|---|---|---|---|---|
| **Einordnung entspr. 2.3.1.3** | **Bestandsbeziehung** | **Generalisierung** | **Funktionale Verbindung** | **Äquivalenz-Verhältnis** |
| Instanzhierarchie (<InstanceHierarchy>) | <InternalElement><br><InternalElement><br></InternalElement><br></InternalElement> | **Klassen-Instanz-Beziehung**<br>SystemUnit-ObjectInstance<br><InternalElement<br>RefBaseSystemUnitPath=””<br>RoleClass-ObjectInstance<br><InternalElement ><br><RoleRequirements<br>RefBaseRoleClassPath=""<br>InterfaceClass-ObjectInstance<br><InternalElement><br><ExternalInterface<br>RefBaseClassPath=”” | **Funkt. Verbind.**<br><InternalLink> | **Mirror-Konzept**<br>Eine Instanz spiegelt die andere (“Mirror” referenziert “Master” in refBaseSystemUnitPath) |

Eine in der Rollenbibliothek(<RoleClassLib>) definierte Rollenklasse (<RoleClass>) kann über das Attribut „RefBaseClassPath“ eine Referenz auf eine Rollen-Superklasse bereitstellen. Das heißt, es ist möglich Rollenklassen durch Ableitung zu verfeinern und zu konkretisieren. Der gleiche Mechanismus ist für Interfaceklassen innerhalb der Interfacebibliothek definiert. Diese Ableitung erfolgt unabhängig von der Hierarchie der Elemente innerhalb der Bibliotheken, d.h. die Hierarchie innerhalb einer Bibliothek kann eine von der Klassenhierarchie unabhängige Ordnungsstruktur repräsentieren (z.B. eine Sortierung nach Quelle der Definition). Die Struktur der beiden Bibliotheken repräsentiert keine definierte Semantik.

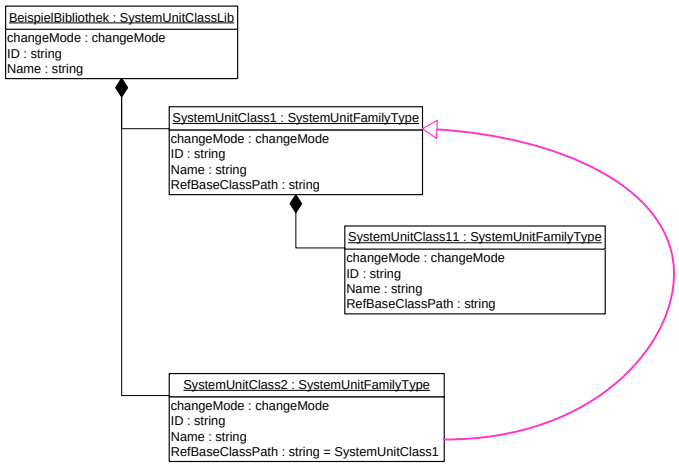


**Abbildung 24 – Hierarchie und Vererbung in einer CAEX-Bibliothek[6]**

Innerhalb der SystemUnit-Bibliothek wird durch die Hierarchie der SystemUnit-Klassen eine Bestandsbeziehung repräsentiert. Dabei ist es nicht möglich zu unterscheiden, ob die Bestandsbeziehung eine Komposition oder eine Aggregation definiert. Für jede SystemUnit-

[6] Bei Abbildung 24 ist zu beachten: Die dargestellten Objekte sind XML-Elemente, die Klassen repräsentieren. Die dargestellt Vererbungsbeziehung bezieht sich nicht auf die XML-Elemente, sondern auf die repräsentierten Klassen.

Klasse kann über das Attribut „RefBaseClassPath“ die Referenz auf eine Superklasse definiert werden (siehe Abbildung 26).

Wie in 3.3.1 dargestellt, wird innerhalb der Instanzhierarchie durch die Hierarchie der Elemente die Bestandsbeziehung der Elemente repräsentiert. Auch hier fehlt die Möglichkeit zwischen Komposition und Aggregation zu unterscheiden.

Die Elemente in der Instanzhierarchie stellen Objekte dar, für die mittels des Attributs „RefBaseSystemUnitPath“ die Vererbung von einer Klasse definiert werden kann. Für das Element kann ebenfalls definiert werden, welche Rolle es im System übernimmt, indem mit dem XML-Element <RoleRequirements> das Attribut „RefBaseRoleClassPath“ definiert wird, das auf eine Rollenklasse in der Rollenbibliothek verweist (siehe Abbildung 25).

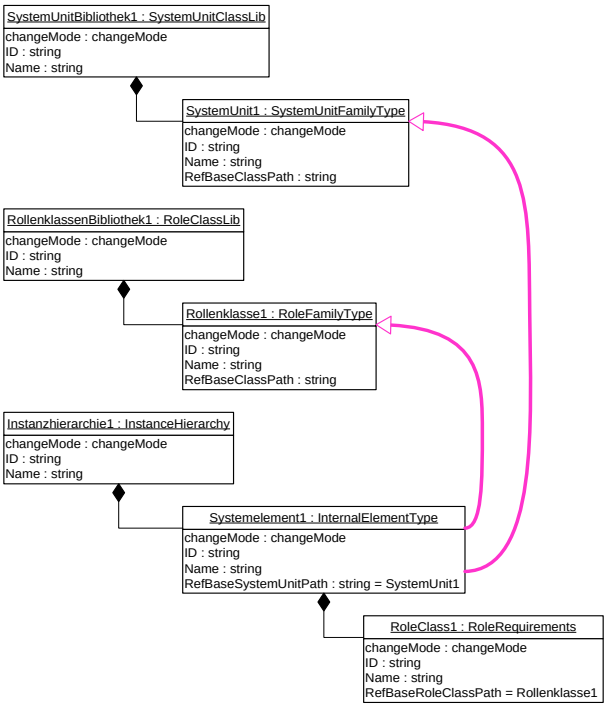


**Abbildung 25 – Definition von SystemUnitKlasse und Rolle für ein Systemelement[7]**

Für die Schnittstellen des Objektes (die es evtl. von der SystemUnit-Klasse geerbt hat), können ebenfalls Referenzen auf die jeweilige Schnittstellenklasse definiert werden. Verbindungen zwischen den Elementen des Systems werden durch <InternalLink>-Elemente repräsentiert. Diese <InternalLink>-Elemente können nur verwendet werden, um Verbindungen zwischen Schnittstellen (<ExternalInterface ) der Elemente darzustellen. Das heißt um funktionale Verbindungen herzustellen, muss ein Element entsprechende Schnittstellen bereitstellen (siehe Abbildung 22).

[7] Bei Abbildung 25 ist zu beachten: Die dargestellten Objekte sind XML-Elemente, die Klassen oder Systemelemente repräsentieren. Die dargestellte Vererbungsbeziehung bezieht sich nicht auf die XML-Elemente, sondern auf Ableitung des Systemelements von den repräsentierten Klassen.

Außerdem ist es möglich mittels Mirror-Konzept auch Äquivalenz-Verhältnis zwischen Elementen (<InternalElement>) darzustellen. Dabei wird das Attribut „RefBaseSystemUnitPath“ genutzt um von einem Element auf ein anderes Element (und nicht auf eine Klasse) zu referenzieren. Das referenzierende Element ist dem referenzierten Element äquivalent und hat die gleichen Attribute wie das referenzierte Element. Mit dem Mirror-Konzept wird das referenzierte <InternalElement> zum Mirror-Master deklariert, auf den verwiesen wird. Das referenzierende Element wird zum Mirror-Slave deklariert, das nur als Referenz dient. Alle Daten und Informationen zu dem <InternalElement> werden als Eigenschaften des Mirror-Masters abgelegt. Es ist möglich, dass Mirror-Master und Mirror-Slave in unterschiedlichen < Instanzhierarchien> verwendet werden, dann repräsentieren trotzdem beide Elemente das gleiche Systemelement. D.h. mit dem Mirror-Konzept ist es möglich Äquivalenzen zwischen Elementen verschiedener Modelle (Instanzhierarchien) darzustellen.

In Abbildung 26 sind als Beispiel eine SystemUnit-Bibliothek „ClassLib 1“ und zwei separate Systemmodelle („Hierarchy 1“ und „Hierarchy 2“) dargestellt, es wird eine Vererbung der Klasse „A“ auf die Elemente „A1“ und „A2“ dargestellt, sowie die Vererbung von Klasse „B“ auf Element „B1“. Als Mirror-Objekt wurde Element „B2“ als äquivalent zu dem Element „B1“ definiert (dargestellt durch gestrichelte Linie). Da die Klasse „B“ in der Bibliothek mit einer Komponente „b1“ definiert wurde, erbt Element „B1“ diese Komponente und das äquivalente Element „B2“ erbt diese Komponente ebenfalls. Weiterhin wurde in „Hierarchy 1“ eine Komponente „B3“ für das Element „B1“ definiert. Diese Komponente wird nicht an das Element „B2“ vererbt, da es nicht Bestandteil der Klassendefinition ist, sondern Bestandteil des Modells „Hierarchy 1“ ist.

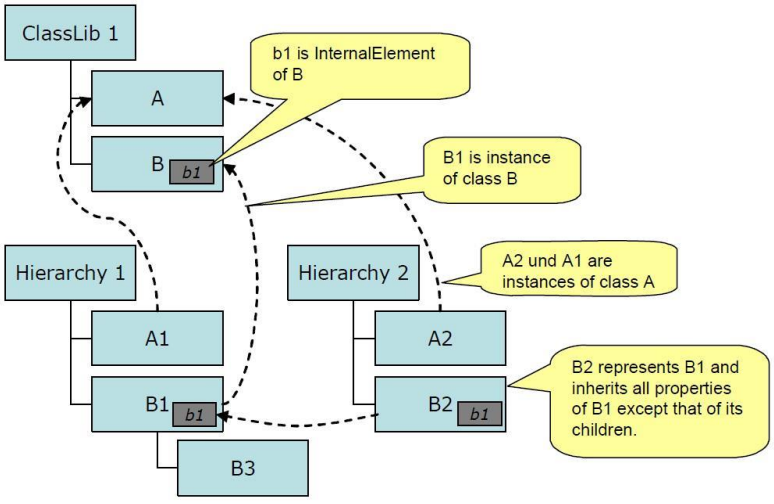


**Abbildung 26 – Darstellung von Vererbung und Äquivalenz mit CAEX v2.15 [IEC08, S. 56f.]**

Mit CAEX3.0 wurde das Mirror-Konzept dahingehend geändert, dass die äquivalenten Objekte (auch Spiegelobjekte genannt, engl. *mirror object*) alle Eigenschaften, Daten und

untergeordneten Objekte der ursprünglichen Objekte (auch Spiegelmaster genannt, engl. *mirror master*) erben und dass die Spiegelobjekte keine untergeordneten Objekte mehr haben können (siehe Abbildung 27) [Dra13, S. 43f.].

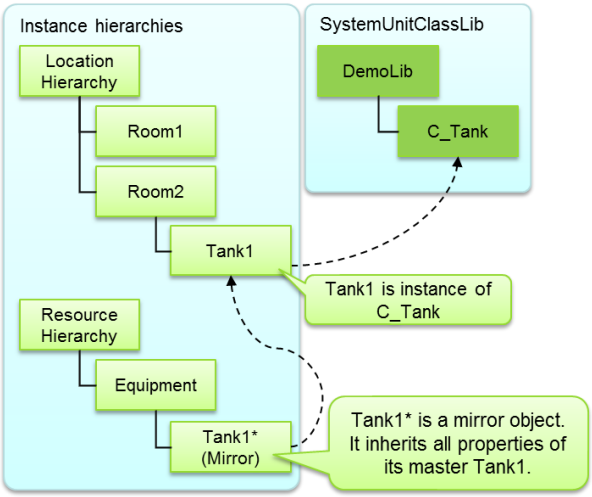


**Abbildung 27 – Darstellung von Vererbung und Äquivalenz mit CAEX v3.0 [Dra13]**

In CAEX ist die Verwendung von Referenzen limitiert. Es kann nur eine Referenz auf eine <SystemUnitClass> und auf eine <RoleClass> (ab CAEX 3.0: mehrere <RoleClass>-Elemente) definiert werden. D.h. es ist möglich darzustellen, dass ein Ventil (Instanz) einem bestimmten Betriebsmitteltyp (<SystemUnitClass>) entspricht und in einer bestimmten Rolle (z.B. Einlaufventil, Ablaufventil) eingesetzt wird. Es ist nicht vorgesehen weitere Verhältnis-Beziehungen darzustellen. D.h. es ist nicht möglich darzustellen, dass eine Klasse die Eigenschaften von zwei Klassen erbt, dass ein Element von zwei SystemUnit-Klassen erbt oder dass eine Beziehung zwischen Elementen in verschiedenartigen Modellen (Funktionales Modell, Komponentenmodell, Topologie-Modell) existiert (siehe 3.2.4). Dies ist ein wesentliches Defizit.

Abstrakte Elemente (Funktionen, Komponenten) können in einer <RoleClassLib> beschrieben werden. Referenzen zwischen <RoleClass>-Elementen werden benutzt um Vererbung darzustellen. Die Darstellung anderer Verhältnis-Beziehungen (z.B. Äquivalenz) innerhalb von <RoleClassLib> wird nicht unterstützt. Es ist auch nicht möglich innerhalb der <RoleClassLib> funktionale Verbindungen zu beschreiben.

Wird von einem <InternalElement> auf eine Rollenklasse in einer Rollenbibliothek referenziert, so wird von einem Strukturelement auf eine Funktionsbeschreibung bzw. auf ein Rollenbeschreibung verwiesen, welche Anforderungen an das Anlagen-Strukturelement definiert. Das heißt die bereitgestellte (bzw. die angestrebte) Funktion eines Strukturelements wird in einer Bibliothek ohne Zusammenhang dargestellt und ohne Einordnung im Rahmen eines funktionalen Modells. - Nach Gierse kann solch eine

Darstellung unter anderem dazu führen, dass der Rahmen einer Entwicklung zu eng oder zu weit gefasst wird, Funktionen in ihrer Bedeutung falsch eingestuft oder vergessen werden. Besser ist es die Beziehung, Zuordnung und Abhängigkeiten der Funktionen darzustellen [Gie90].

Eine Möglichkeit zur Darstellung anderer Verhältnis-Beziehungen (z.B. Teiläquivalenz, Generalisierung, Realisierung, siehe 2.3.3) ist nicht gegeben (siehe Tabelle 9). Es existiert ein Vorschlag zur Verwendung von Interfaces und <InternalLink>-Elementen zur Darstellung von Verhältnisbeziehungen [SF10, JCS+12]. Eine klare Trennung zwischen funktionalen Verbindungen und Verhältnis-Beziehungen ist jedoch im Sinne einer klaren Semantik erstrebenswert, insbesondere da ein <InternalLink>-Element nur innerhalb eines <InternalElement>-Elements (und nicht zwischen verschiedenen <SystemHierarchy>-Elementen) verwendet werden kann. Außerdem kann für ein <InternalLink>-Element keine weitere semantische Information (z.B. Verbindungstyp, Attribute der Verbindung) hinterlegt werden.

**Tabelle 9 – Unterstützung von Elementbeziehungen in CAEX**

| Beziehung | Unterstützung | Kommentar |
|---|---|---|
| Funktionale Verbindungen | | |
| Binary Association | <InternalLink> | Keine Eigenschaften oder Typisierung |
| n-ary Association | <InternalElement> mit spezifischer Role und <InternalLink> | |
| Bestandsbeziehungen | | |
| Aggregation | XML-Hierarchie | Keine Unterscheidung zwischen Aggregation und Komposition. |
| Komposition | XML-Hierarchie | |
| Verhältnisbeziehungen | | |
| Realisierung | <SupportedRoleClass> reference | Referenz nur auf eine Bibliothek, nicht auf eine andere <SystemHierarchy> |
| Äquivalenz | Mirror-Konzept | Äquivalenz nur zwischen Instanzen, nicht zwischen Klassen |
| TeilÄquivalenz<br>Generalisierung<br>Abhängigkeit<br>Komplementärbeziehung | | Nicht unterstützt. |

### 3.3.5 Darstellung von Verhaltensmodellen

Die Darstellung von Verhaltensmodellen ist prinzipiell bei CAEX nicht vorgesehen. Im Kontext von AutomationML werden jedoch zwei Methoden zur Verhaltensmodellierung dargestellt:

- Modellierung von Verhalten mit CAEX-Elementen
- Modellierung von Verhalten mit PLCopen XML

#### *3.3.5.1 Modellierung von Verhalten mit CAEX-Elementen*

Bei dem in [Dra10, S. 46f.] beschriebenen Entwurfsprozess wird im ersten Schritt ein funktionales Modell erstellt, in dem die Funktionen (Prozesse) als Systemelemente (<InternalElement>) mit entsprechenden Rollen modelliert werden. Funktionale Verbindungen (Prozessfluss) werden durch <InternalLink>-Elemente dargestellt. Dieser Ansatz ist Ausgangspunkt für alle weiteren in diesem Abschnitt diskutierten Ansätze. Eine Persistenz des funktionalen Modells ist jedoch nicht vorgesehen, vielmehr wird davon ausgegangen, dass das bestehende Modell in weiteren Schritten in ein Strukturmodell überführt wird.

[Dra10, S. 83] beschreibt die Abbildung des Produkt-Prozess-Ressource-Konzepts in AutomationML. Prozesse werden als <InternalElement>-Elemente dargestellt und mit Hilfe von <InternalLink>-Elementen werden Prozessabläufe modelliert. Darauf basierend beschreibt Keibel in [Dra10, S. 195] die Modellierung von Bearbeitungsprozessen mittels AutomationML. Es wird als Anforderung beschrieben, dass basierend auf der Prozessbeschreibung ein ausführbarer Programm-Quellcode für Roboter generiert werden soll. Es werden die zu modellierenden Objekte der Prozessbeschreibung dargestellt und beschrieben, dass diese Objekte „mit Hilfe des CAEX Rollenkonzeptes“ modelliert werden. Die dort beschriebenen Attribute beziehen sich auf das Werkzeug ([Dra10, S. 208f.]) und auf den Prozess ([Dra10, S. 213f.]). Das Werkzeug wird in einer <SystemUnitClassLib>-Bibliothek modelliert und die Prozessdarstellung in einer <RoleClassLib>-Bibliothek. In dem Anlagenmodell (einer <InstanceHierarchy>) werden dann Prozess, Produkt und Ressource jeweils in eigenen Sub-Hierarchien (<InternalElement>) modelliert. Verweise zwischen Prozessmodell und Strukturmodell der Anlage erfolgen über <InternalLink>-Elemente. Der modellierte Prozess beschreibt allerdings nicht das Verhalten des Gesamtsystems, sondern den Steuerungsprozess, der mit Ablaufsprache (AS, englisch: *sequential function chart* (SFC)) modelliert wird [Dra10, S. 217].

Strube in [SF10] und Jäger in [JCS+12] beschreiben die Darstellung eines funktionalen Modells (basierend auf Formalisierter Prozessbeschreibung) mit CAEX. Die Operatoren der Formalisierten Prozessbeschreibung werden als <SystemUnitInstance>-Elemente mit entsprechender Rollendeklaration dargestellt. Die funktionalen Verbindungen (Fluss) werden mit <InternalLink>-Elementen dargestellt. Verhältnis-Beziehungen zwischen Prozessoperator und technischer Ressource werden ebenfalls mit <InternalLink>-Elementen dargestellt. Eine Trennung von funktionalem Modell und Systemmodell erfolgt durch die Darstellung in unterschiedlichen <InternalElement>-Elementen (siehe Abbildung 28).

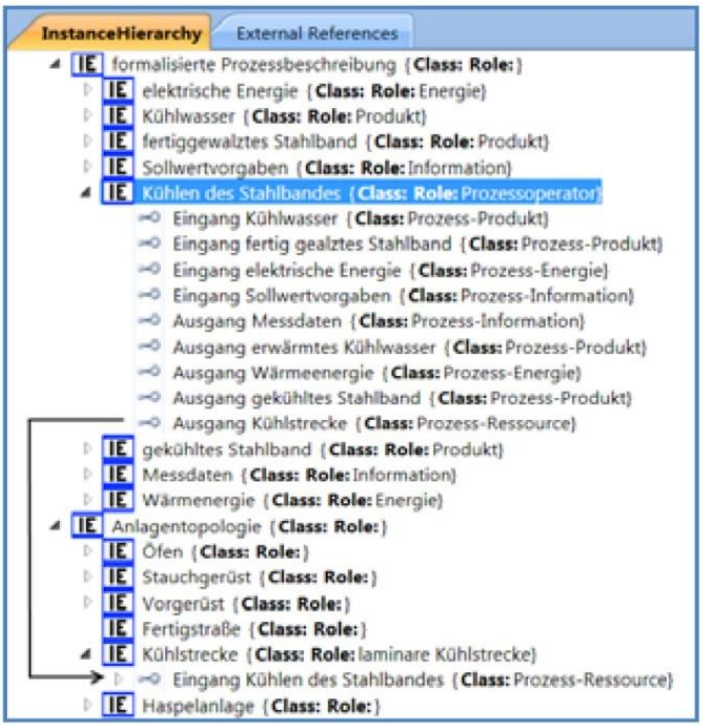


**Abbildung 28 – Verknüpfung von Anlagen- und Prozessbeschreibung [JCS+12]**

Bei beiden Ansätzen wird nicht begründet, wieso funktionales Modell und Komponentenmodell in der gleichen Instanzhierarchie beschrieben werden. – Wahrscheinlich liegt das daran, dass für die Darstellung des Verhältnisses ein <InternalLink>-Element verwendet wurde. <InternalLink>-Elemente können nur innerhalb einer Instanzhierarchie oder innerhalb eines <InternalElement> verwendet werden. Eine Separierung der Modelle in unterschiedliche Instanzhierarchien ist diesem Ansatz vorzuziehen, da dadurch eine Trennung der Modell-Domänen erfolgt.

#### *3.3.5.2 Modellierung von Verhalten mit PLCopen XML*

AutomationML definiert eine Interfaceklasse „PLCopenXMLInterface“, die genutzt werden kann, um PLCopen-XML-Dokumente zu referenzieren [IEC11a, S. 29f.]. Eine Standardisierung für die Verwendung von PLCopen-XML-Dokumenten zur Beschreibung von Logik ist in Teil 4 des AutomationML-Standards geplant, aber noch nicht veröffentlicht. Eine informelle Beschreibung des Ansatz findet sich in [IEC11a, S. 52f.].

Die Vorgehensweise zur Nutzung der Referenz auf ein PLCopen-Dokument zur Verhaltensbeschreibung ist in [Dra10, S. 135] bzw. in [Hun12] beschrieben. In dem referenzierten PLCopen-XML-Dokument können verschiedene Arten von Informationen in den 5 Sprachen der IEC 61131 (Funktionsblockdiagramme (FBD), Strukturierter Text (ST), Anweisungslisten(AWL), Ablaufsprache(AS) und Kontaktplan (KOP)) bereitgestellt werden.

Um das Verhalten einer Komponente zu beschreiben wird die Ablaufsprache genutzt. Andere Verhaltensbeschreibungen (z.B. Ganttchart, Impuls-Diagramme) können in dieses Format konvertiert werden [Dra10, S. 135].

Der dargestellte Ansatz ist gut geeignet, um das Verhalten einer einzelnen Komponente bzw. einer Steuerung zu beschreiben. Die Nähe zu Programmierwerkzeugen für speicherprogrammierbare Steuerungen (SPS) erlaubt einen effektiven Datenaustausch zwischen diesen Werkzeugen und Engineering-Programmen.

Für die Beschreibung des Verhaltens eines Produktionssystems und für die Entwicklung eines Produktionssystems basierend auf solch einer Verhaltensbeschreibung ist der Ansatz eher nicht geeignet. Auch wenn es möglich ist, das Systemverhalten mit Ablaufsprache zu beschreiben, besteht bei einer Beschreibung des Verhaltens außerhalb des CAEX-Dokuments anschließend das Problem, dass es nicht möglich ist, Beziehungen zwischen dem extern beschriebenen Verhalten und den im CAEX-Dokument entwickelten, darauf basierenden Struktur-Modellen zu dokumentieren.

### 3.3.6 Integration mit Merkmalmodellen

Lüder et al. präsentieren in [LSG+14] verschiedene Ansätze zur Integration von Merkmalbeschreibungen in AutomationML bzw. CAEX. Es werden 3 Ansätze diskutiert:

- „Referenzierung in eine eCl@ss <RoleClassLib>“,
- „Referenzierung in eine eCl@ss <SystemunitClassLib>“,
- „Referenzierung auf Attribute“[LSG+14].

Die ersten beiden Ansätze bestehen darin, dass die eCl@ss-Klassifizierungshierarchie jeweils in eine Bibliothek mit Klassenhierarchie übersetzt wird. Der Unterschied besteht darin, dass die Ziel-Bibliothek einmal Rollen (bzw. Funktionen in einem System) beschreibt und das andere Mal die Betriebsmittel beschreibt. Der dritte Ansatz besteht darin, dass die Attribute mittels dem Attribut RefSemantik die entsprechenden Merkmaldefinition referenzieren.

Entsprechend der Präsentation durch Lüder auf der Automation Konferenz 2014 wird der Ansatz zum Import von Merkmalbeschreibungen als Rollenbeschreibungen präferiert [Lüd14]. Der Import von Merkmalbeschreibungen als Betriebsmittelbeschreibung (in eine <SystemunitClassLib>) wird bewertet mit einer „.. sich daraus ergebene höhere Komplexität der Semantikabbildung“ [LSG+14]. Diese Bewertung wird leider nicht erklärt und ist nicht nachvollziehbar.

Der Ansatz zum Import als <RoleClassLib> ist unzureichend, weil er nicht die Diversität und die Erweiterungsmöglichkeiten der existierenden Gerätebeschreibungen berücksichtigt. Auch ist die Konvertierung der Beschreibung eines Betriebsmittels zu einer

Rollenbeschreibung fragwürdig. – Der Autor würde eher die Konvertierung einer Betriebsmittelbeschreibung zu einer Systemunit-Klasse erwarten, wobei dieser Ansatz mit dem 3. Ansatz (Referenz der Attribute) kombiniert werden sollte. Um diese Kritikpunkte zu illustrieren werden im nächsten Abschnitt die existierenden Merkmalmodelle diskutiert. Ein Vorschlag zur Integration von Merkmalbeschreibungen in Systemmodelle wird in 4 und 5 dargestellt.

## 3.4 Merkmalmodelle in Wissenschaft und Technik

### 3.4.1 Übersicht

In Abschnitt 2.2 wurde dargestellt, dass bereits seit einiger Zeit an Merkmalmodellen gearbeitet wird. Inzwischen gibt es eine Reihe von Standards, die sich mit Merkmalmodellen beschäftigen (siehe Abbildung 29). In diesem Abschnitt sollen diese Merkmalmodelle vorgestellt werden.

Abbildung 29 stellt eine schematische Übersicht der verschiedenen Standards nach Heeg dar [Hee05]. Auf die Darstellung der Standards, die sich auf Messgrößen beziehen wurde verzichtet, weil sie für diese Arbeit nicht relevant sind. Die Einordnung wurde etwas geändert um die unterschiedlichen Typen der Standards zu reflektieren.

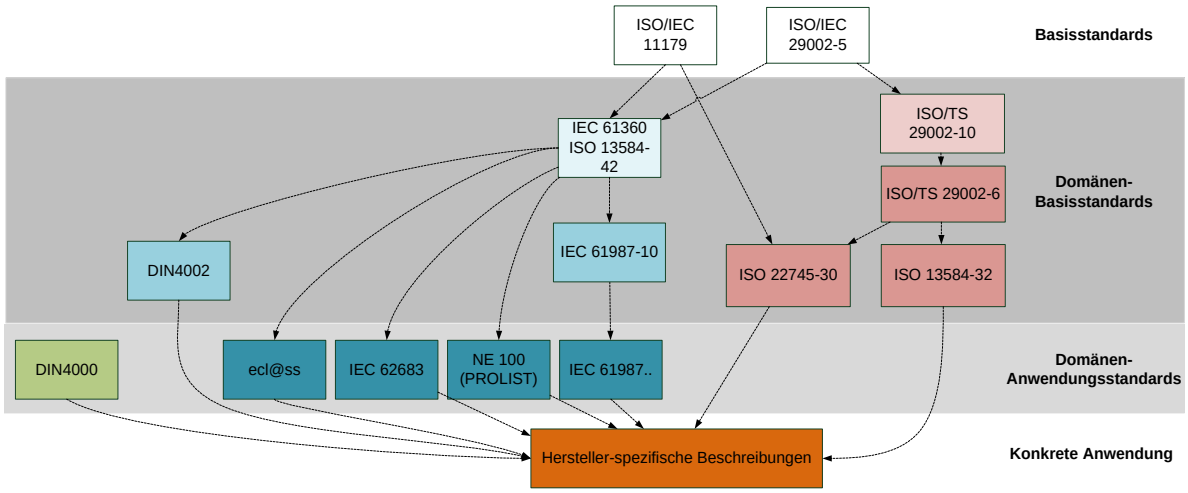


**Abbildung 29 – Übersicht über Standards zu Merkmalmodellen**

ISO/IEC 11179 und ISO/IEC 29002-5 werden als allgemeine Basisstandards betrachtet, weil sie in anderen Anwendungsdomänen verwendet werden. Domänen-Basisstandards definieren Konzepte, welche allgemein zur Erstellung von Merkmalbeschreibungen verwendet werden. Die dargestellten Domänen-Anwendungsstandard definieren Standard-Beschreibungen für Betriebsmittel, welche zum Teil direkt oder in einer angepassten Form für Beschreibungen von Betriebsmitteln verwendet werden.

Die dargestellten Standards sind nicht vollständig. Es fehlen einige Anwendungsstandards (z.B. ETIM, profiClass). Diese Industriestandards entsprechen aus Sicht dieser Arbeit im Prinzip dem dargestellten eCl@ss-Standard. Für eine Darstellung weiterer Standards wird auf [Ges10, Hee05] verwiesen.

### 3.4.2 Identifikation

Ein wesentlicher Aspekt von Merkmalmodellen ist, dass die Definition eines Merkmals typischerweise getrennt von der Verwendung des Merkmals erfolgt. Die Beziehung zwischen der Definition des Merkmals und der Ausprägung des Merkmals wird über den Identifikator hergestellt.

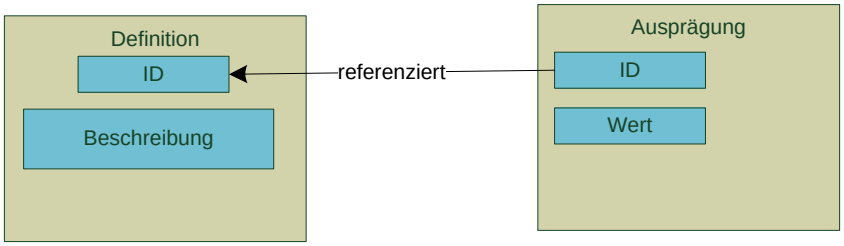


**Abbildung 30 – Beziehung zw. Merkmaldefinition und Merkmalbeschreibung**

Dieser Ansatz entspricht dem in der IEC 62507-1 definierten Identifizierungssystem, bei dem das Objekt jeweils mit einer Identifikationsnummer identifiziert wird und über diese Identifikationsnummer dem zu beschreibenden Objekt jeweils die entsprechenden Beschreibungen (in dargestellt durch Metadaten) zugeordnet werden können (siehe auch die Diskussion in [HGH+15]).

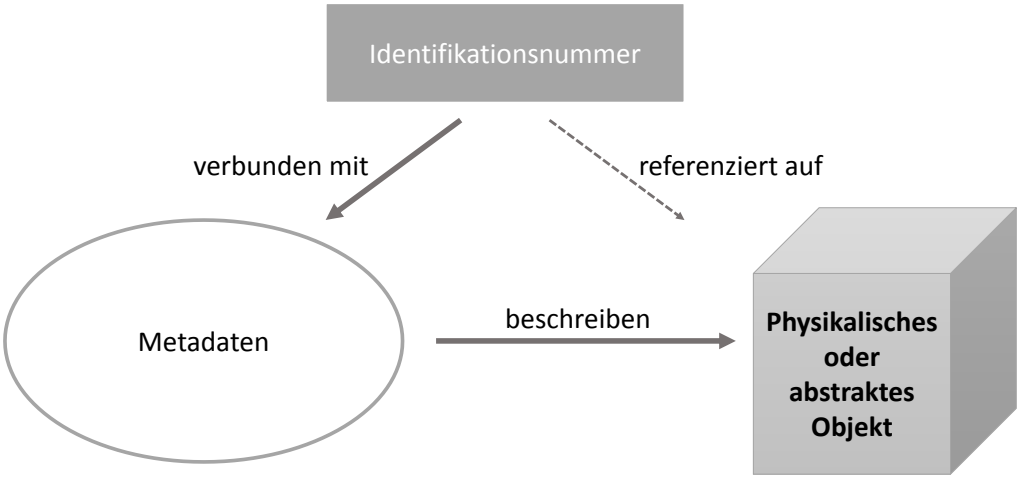


**Abbildung 31 – Darstellung des Referenzmechanismus nach [IEC10]**

D.h. in der Ausprägung eines Merkmals wird die Definition des Merkmals referenziert (d.h. die Metadaten) und Ausprägung und Merkmalsdefinition zusammen beschreiben ein physikalisches oder abstraktes Objekt (bzw. nur einen Aspekt dieses Objektes).

Verschiedene Merkmalmodelle verwenden unterschiedliche Arten von Identifikatoren, beispielsweise numerische Identifikatoren wie im IBM part number System, GUIDs oder URI. Ein wichtiges Identifikationssystem wurde mit der ISO 290002-5 definiert (siehe Abbildung 32).

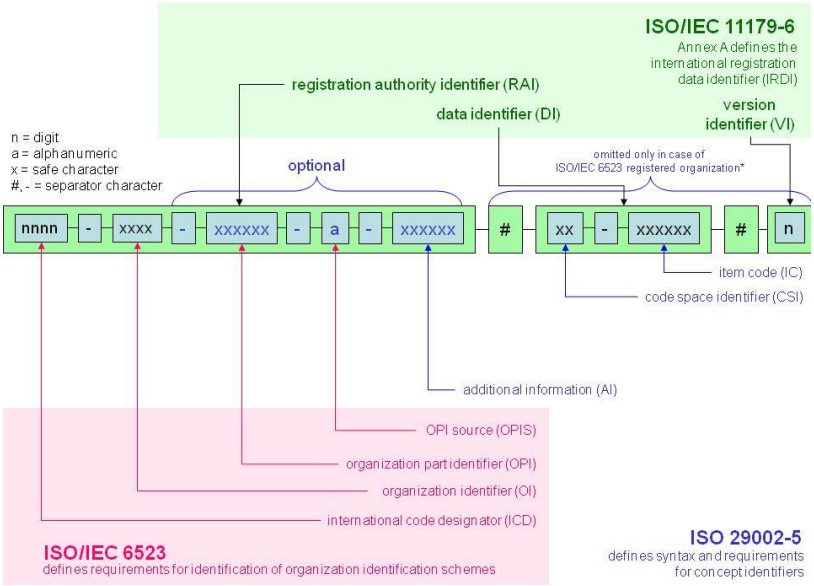


**Abbildung 32 – Identifier Format [ISO09b, S. 14f.]**

Wesentliche Eigenschaft eines Identifikators nach ISO 29002-5 ist, dass er strukturiert ist und dass verschiedene Teile der Struktur jeweils unterschiedliche Bedeutungen haben. Wesentliche Teile der Struktur sind RAI (registration authority identifier), DI (data identifier) und VI (version identifier). Der RAI identifiziert weltweit eindeutig den Herausgeber und die zugeordnete Datensammlung (z.B. Dictionary, Library) des Identifikators. Der DI identifiziert eindeutig eine Konzeptbeschreibung innerhalb der Datensammlung. Konzept kann dabei eine Klasse, ein Merkmal, eine Darstellung, eine Masseinheit, ein Messqualifizier, ein Wert, eine Währung, ein Datentyp, ein Dokument oder eine Ontologie. Der VI identifiziert die Version der Beschreibung. RAI und DI wiederum sind strukturiert wie in Abbildung 32 dargestellt. Die RAI enthält beispielsweise eine ICD (international code designator), welche die Organisation eindeutig identifiziert von der ein Identifikator stammt (z.B. 0172 für PROLIST, 0173 für eCl@ss) . Die DI enthält z.B. einen IC (item code), welcher ein Konzept innerhalb des jeweiligen Namensraumes eindeutig identifiziert.

### 3.4.3 ISO/IEC 11179

Der Standard ISO/IEC 11179 führt auf der Basis grundsätzlicher Konzepte (‚metamodel construct' z.B. Klasse, Assoziation, Relationship) die Konzepte (‚metadata object') zur Verwaltung von Daten ein. ISO/IEC 11179 führt das Konzept von Klassifikationsschemata ein, es werden zentrale Konzepte wie Metamodell, Datenmodell, Registry, Charakteristik,

Datenelement, Property und Attribut eingeführt, Prinzipien zur Bezeichnung und Identifikation definiert und das Konzept der MetaDaten-Registry spezifiziert, in der zentral die Datenformate für eine Organisation gespeichert werden können [ISO04a]. Basierend auf diesen Konzepten wurden Anwendungen entwickelt (z.B. [RS12, War12]) aber auch weitere Standards (für Merkmalmodelle usw.).

### 3.4.4 IEC 61360

Die IEC 61360 bildet eine wichtige Grundlage zur Verwendung von Merkmalen in der internationalen Standardisierung. Während IEC 61360-1 [IEC09b] ein allgemeines Merkmalmodell definiert, wird in der IEC 61360-2 [IEC04a] ein entsprechendes Datenmodell definiert. Dieses Datenmodell ist mit der ISO abgestimmt, die Basisnorm IEC 61360-2 [IEC04a] entspricht der ISO 13584-42[ISO10].

IEC 61360-1 definiert ‚*data element type*' (Datenelementtyp, DET) und ‚*item class*' (Itemklasse) Datentypen, beide sind Konzepte entsprechend ISO/IEC 11179. Datenelementtyp können entweder ‚*property*' (entspricht dem Merkmal) oder ‚*condition*' sein. Diese Datenelementtypen werden den abstrakten Itemklassen (*item class*) zugeordnet. Es gibt zwei Klassentypen, die von Itemklasse abgeleitet werden: ‚*super class*' (Superklasse) und ‚*sub class*' (Unterklasse). Basierend auf diesen Klassentypen kann eine Klassifizierungshierarchie definiert werden, bei der abgeleitete Klassen (Unterklassen) die Zuordnung von DET von den übergeordneten Klassen (Superklassen) erben und ihnen weitere DET zugeordnet werden können. Die Unterscheidung von Unterklassen erfolgt anhand von Werten der ‚*classifying DET*' (klassifizierende Datenelementtypen). Instanzen dieser Klassen werden in ‚*dictionaries*' (Wörterbücher) gespeichert. Es wird ein Standard-Wörterbuch für die IEC mit dem Identifikator „IEC CDD" definiert [IEC09b][IEC09b]. Dieses Wörterbuch ist allgemein als *IEC Common Data Dictionary* (IEC CDD) bekannt. Die RAI für das IEC CDD lautet „0112/2///61360_4#IECCDD#".

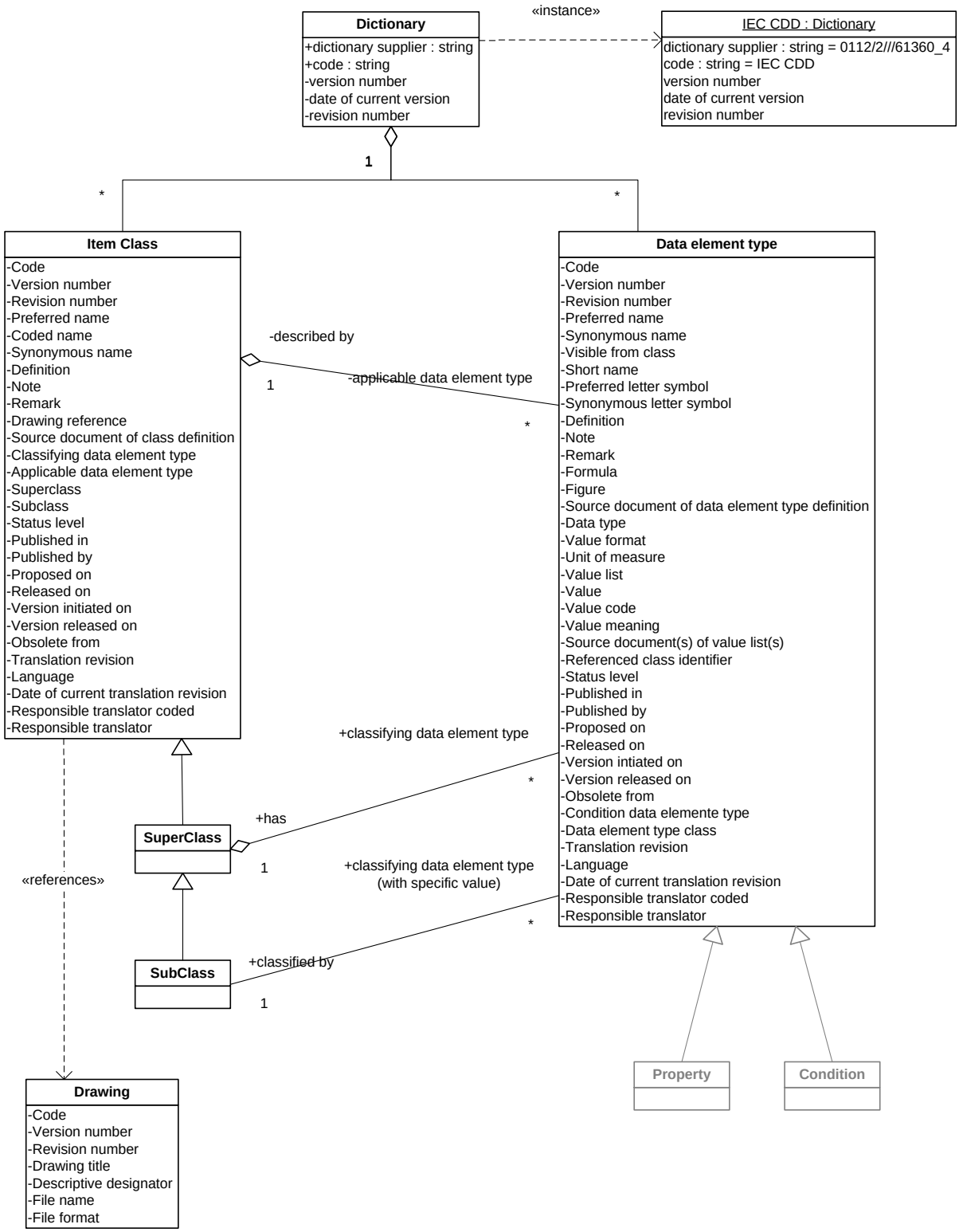


**Abbildung 33 – IEC 61360 Datentypen**

Diese IEC 61360 Datentypen sind eine wesentliche Grundlage zur Kommunikation von Informationen über Komponenten von Prozessleitsystemen zwischen Anbietern und Nutzern dieser Komponenten [ADK+06]. Interessanter Aspekt dabei ist, dass alle String-Attribute der IEC 61360 Datentypen(z.B. Definition für Itemklasse und DET) mehrsprachig definiert sein

können. D.h. man kann alle Konzepte in verschiedenen Sprachen darstellen und eindeutig über ihre Identifikatoren (hier dargestellt als *code*) identifizieren. Dadurch ist es einfach Missverständnisse im internationalen Handel zu vermeiden.

### 3.4.5 DIN 4000 Serie

DIN 4000 Sachmerkmal-Listen basieren nicht auf der IEC 61360 und nutzen ein anderes Identifier-Format als durch ISO 29002 definiert. Die Norm ist jedoch Basis für die Beschreibung von Gerätestammdaten, welche noch häufig in der Industrie genutzt werden [Ins10]. Die DIN 4000 umfasst derzeit ca. 120 Teile. Die ersten Dokumente wurden 1975 veröffentlicht [DIN12], die letzten Dokumente 2013 [DIN13]. Die DIN 4000 definiert Sachmerkmal-Listen für verschiedenste konstruktive Elemente des Maschinenbaus. Die Norm unterscheidet zwischen einem Merkmal, das definiert ist als „Datenelementtyp [DIN EN 61360-1][…] Eigenschaft, die zum Beschreiben und Unterscheiden von Gegenständen dient“ und einem Sachmerkmal, das definiert ist als „Beschreibung eines Gegenstandes, so wie dieser sich dem Betrachter darstellt“ [DIN12]. Das heißt, das Konzept „Sachmerkmal“ kann als Super-Klasse verstanden werden, welches zur Beschreibung von Gegenständen dient, und eine Teilklasse von Sachmerkmal ist das Konzept „Merkmal“, welches auch zur Klassifizierung des Gegenstandes genutzt werden kann.

### 3.4.6 DIN 4002 Serie

Die DIN 4002 Serie „Merkmale und Geltungsbereiche zum Produktdatenaustausch“ wurde basierend auf der IEC 61360 entwickelt. Teile 1 bis 7 der Norm definieren eine Arbeitsanleitung zur Erstellung von Merkmalbeschreibungen. Teile 100 und 101 definieren Merkmale und Merkmalleisten [Poh04]. Die definierten Merkmale sollen in Klassifikationsstandards wiederverwendet werden [Ins10].

Merkmale werden entsprechend DIN 4002 auf einem Merkmalserver (www.DinSML.net) [DIN00] veröffentlicht.

Die Definitionen der DIN 4002 sind Grundlage für die Überarbeitung der Sachmerkmalleisten nach DIN 4000, so dass die überarbeiteten Sachmerkmale nun Attribute entsprechend DIN 4002-3 (z.B. Identifikator) haben (z.B. [DIN07]).

### 3.4.7 DIN PAS 1040 Reihe

Die DIN PAS umfasst derzeit 3 Dokumente (PAS 1040 bis PAS 1042), welche Sachmerkmallisten für die Verfahrenstechnik definieren. „PAS“ steht für öffentlich verfügbare Spezifikation (publicly available specification) und markiert die Dokumente als Vorschlag, der außerhalb des DIN e.V. erarbeitet wurde. Der Vorschlag wurde von einer Firmengruppe in Zusammenarbeit mit der DIN erarbeitet [Zgo12].

### 3.4.8 PROLIST

Im April 2003 wurde die Projektgruppe ‚Merkmalleisten' (PROLIST) als Non-Profit-Organisation gegründet [Nam05] und im Jahr 2008 in den PROLIST International e.V. überführt. Diese Gruppe hatte das Ziel Merkmalsleisten für Geräte und Systeme der Prozessleittechnik zu erstellen und eine Standardisierung im Rahmen der IEC vorzubereiten. Zu den Zielen der Gruppe gehörte immer eine Unterstützung der Arbeit von eCl@ss. Diese Zusammenarbeit mündete 2012 in die Integration von PROLIST in den eCl@ss e.V. [Ahr10a].

Die Projektgruppe arbeitete sehr erfolgreich, während Version 1.0 der NE100 bereits 41 Gerätespezifikation unterstützte [Nam03], werden mit Version 3.2 der NE100 nun Gerätespezifikationen für 110 Gerätetypen unterstützt [PRO12].

PROLIST Merkmallisten basieren auf der IEC 61360 [IEC04b]. Es wird eine Klassifizierungshierarchie basierend auf Superklassen und Subklassen genutzt. PROLIST führte neue Konzepte für Merkmalbeschreibungen ein: Blöcke, Aspekte, Kardinalität und Polymorphismus.

Blöcke dienen der Gruppierung von Merkmalen nach Betrachtungsgegenständen. Ein solcher Betrachtungsgegenstand kann zum Beispiel die Zusammenfassung der Kontaktinformationen für den Lieferanten eines Gerätes sein, oder die Beschreibung eines Moduls eines Gerätes.

Kardinalität beschreibt die Möglichkeit, dass bestimmte Elemente (z.B. Module, Schnittstellen) mehrfach in einer Beschreibung vorhanden sein können. Die Anzahl der Module wird über ein Zählmerkmal gesteuert, die Beschreibung der Elemente erfolgt durch eine Blockdefinition, die mehrfach in der Gerätebeschreibung eingefügt wird (entsprechend der durch das Zählmerkmal vorgegebenen Anzahl) [Zgo07, S. 11f.].

Es ist möglich für unterschiedliche Aspekte eines Gerätes unterschiedliche Gerätemerkmalleisten bereitzustellen. In der NE 100 werden Geräte-Merkmalleiste (GML, für die technische Beschreibung des Gerätes), Betriebs-Merkmalleiste (BML, für die Beschreibung der Anforderungen an ein Gerät), Administrative Merkmalleiste (AML, für die Verwaltung des Merkmalleisten-Dokuments) und Kommerzielle Merkmalleiste (KML, für die Beschaffungsdaten des Gerätes) definiert. Es ist möglich, weitere Merkmalleisten zu definieren [Zgo07, S. 12f.]. Dieses Konzept ist als „*list of properties*" in die IEC 61987-10 eingegangen [IEC09a, S. 22f.].

Polymorphismus ermöglicht die Beschreibung von Varianten für einen Aspekt der Geräte. Dabei wird für jede Variante jeweils ein Block mit einer entsprechenden Beschreibung bereitgestellt. Über den Wert eines Steuermerkmals wird in der Gerätebeschreibung jeweils der gültige Block definiert, der zu der Beschreibung des Gerätes genutzt wird [IEC09a, S. 20f.].

Durch PROLIST erfolgte auch eine Definition eines Workflows zur Kommunikation zwischen Nutzern und Lieferanten von Geräten unter Nutzung von Merkmalleisten (siehe Abbildung 34).

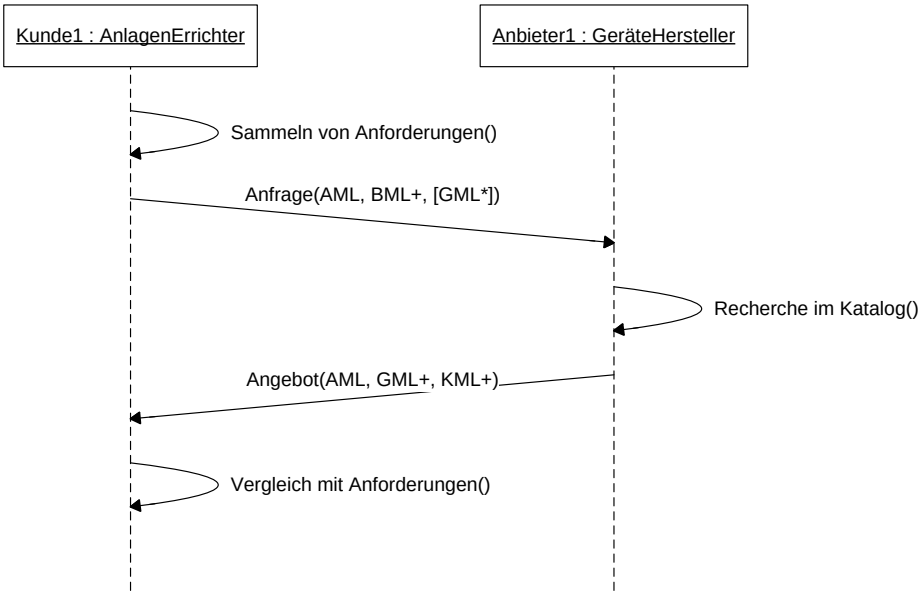


**Abbildung 34 – Kommunikation nach PROLIST nach [Zgo07, S. 14f.]**

Dabei erfolgt beim Kunden (in diesem Beispiel ein Hersteller von Produktionssystemen) eine Sammlung von Anforderungsinformationen (z.B. Prozessbedingungen, Hilfsenergieversorgung, Umgebungsbedingungen) für die benötigten Automatisierungsgeräte. Diese Sammlung kann mit einem CAE-Tool ausgeführt werden. Diese Informationen werden für jede Messstelle in eine BML zusammengefasst. Es ist auch möglich, dass ein Kunde bereits gerätespezifische Eigenschaften vorgibt, die in einer GML beschrieben werden. Bei der Anfrage werden die BML (bzw. GML) zusammen mit einer AML an die Anbieter übermittelt. (Abbildung 34 beschreibt, dass die Anfrage genau eine AML, mindestens eine BML und optional GML enthält.) Der Anbieter überprüft nun welche Geräte die Anforderungen erfüllen und stellt für jede angefragte Messstelle eine oder mehrere GML bereit (falls mehrere Gerätetypen die Anforderungen erfüllen können). Die AML wird angereichert mit herstellerspezifischen Angaben (z.B. Kontaktperson für dieses Angebot). AML und GML werden mit KML ergänzt, in der kommerzielle Aspekte des Angebots beschrieben werden (z.B. Lieferzeit). (Abbildung 34 beschreibt, dass das Angebot genau eine AML, mindestens eine GML und mindestens eine KML enthält.)

### 3.4.9 eCl@ss

Der eCl@ss e.V. wurde im November 2000 gegründet. Ziele des eCl@ss e.V. sind es einen Standard für die Datenstrukturen und Datenformate von Stammdaten sowie Schnittstellendefinitionen für unternehmensübergreifende Prozesse zu definieren [eCl14].

eCl@ss definiert einen Industriestandard zur Klassifikation von Betriebsmitteln. Diese Klassifikation ist hierarchisch gegliedert. Ziel ist es Materialien, Produkte und Dienstleistungen in eine logische Struktur einzuordnen, in deren untersten Detaillierungsebene diese Elemente mit Standard-Merkmalen beschrieben werden können. Diese logische Struktur gliedert sich in 4 Ebene: Segment, Hauptgruppen, Gruppen und Klassen [eCl13].

Diese Klassifikationshierarchie stellt keine Klassenstruktur im Sinne einer Vererbungshierarchie dar, sondern eine Einteilung der Klassen(auf der 4. Ebene) in eine andere Gruppierung. Wesentliches Element ist die Zuordnung der Klassen zu den mit einer ID versehenen Strukturelementen (jeweils 2 Ziffern), so dass sie über eine zusammengesetzte Klassen-ID (8 Ziffern) identifizierbar sind (siehe Abbildung 35).

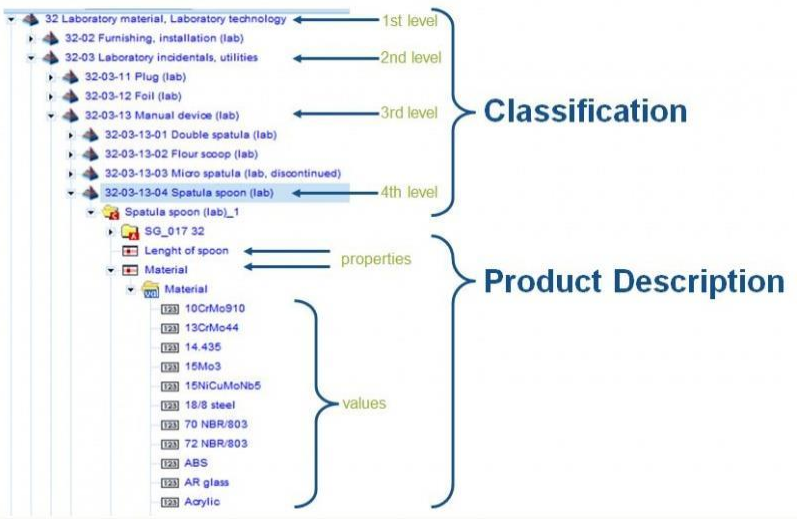


**Abbildung 35 – Struktur von eClass [eCl13]**

Seit der Version 4.0 von eCl@ss, die im Jahr 2003 veröffentlicht wurde, erfolgt die Beschreibung der Produkte mittels Merkmalen. Die Merkmale werden entsprechend der ISO 29002 identifiziert. Seit der Version 7.0 werden Beschreibungen auf unterschiedlichem Niveau angeboten – „Basic“ und „Advanced“. Eine „Advanced“-Beschreibung kann in Blöcke strukturiert werden und enthält mehr Informationen als die „Basic“-Beschreibung [eCl13].

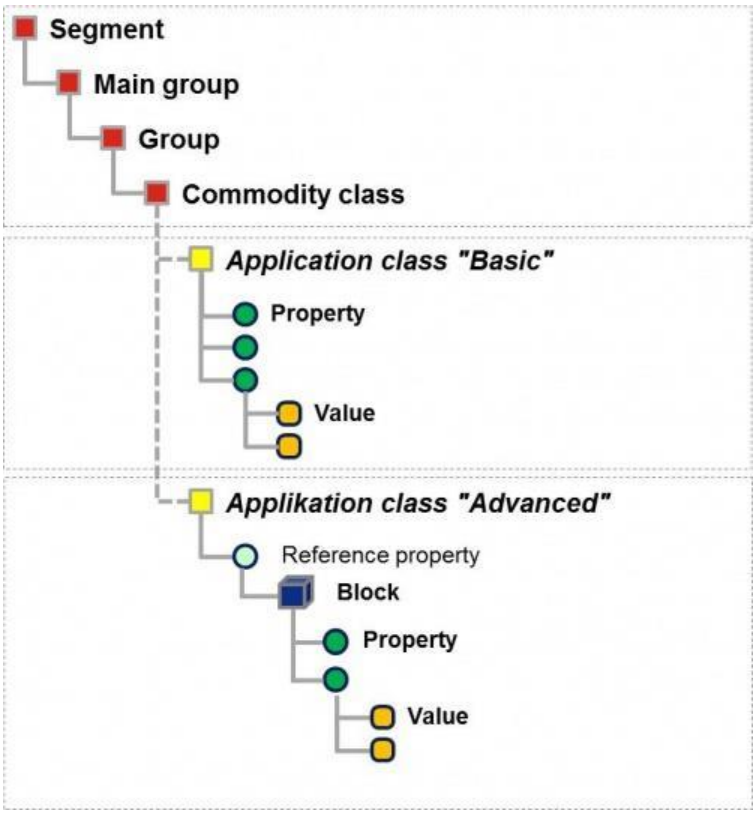


**Abbildung 36 – Unterscheidung in Basic und Advanced [eCl13]**

Die Klassifikation mit Basic-Beschreibung kann über www.eclasscontent.com abgerufen werden.

### 3.4.10 Auf IEC 61360 basierende internationale Standards

Die Definition nach IEC 61360 ist Ausgangspunkt für weitere Normen zur Beschreibung von Mitteln der Automatisierungstechnik, z.B. IEC 61987, IEC 62683. IEC 62683 basiert auf der Klassendefinition von IEC 61360. Es werden formell keine eigenen Itemklassen definiert, eine „*list of properties*“(LOP) wird nur informell definiert [IEC11b]. Die NNorm IEC 61987-10, welche basierend auf den PROLIST-Definitionen entwickelt wurde, führt verschiedene weitere Itemklassen ein und definiert Regeln für die Beschreibung von AT-Geräten. Das Klassenkonzept schließt die Möglichkeit der Vererbung ein und ermöglicht Polymorphismus [IEC09a, S. 30]. Die Klassen „*block of properties*“ und „*list of properties*“ (LOP) werden formell von der Itemklasse abgeleitet (siehe Abbildung 37).

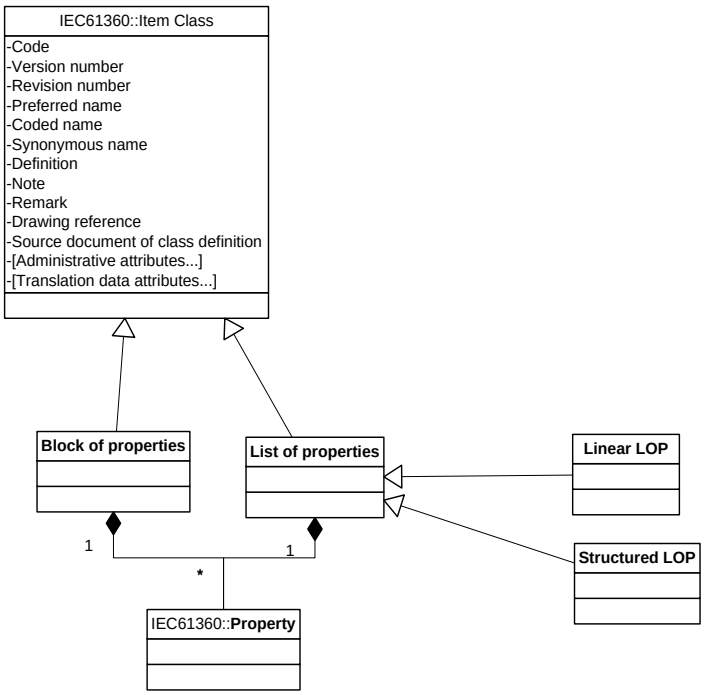


**Abbildung 37 – IEC 61987-10 Datentypen**

Mit den verschiedenen Itemklassen können verschiedene Betriebsmittel und ihre Komponenten, z.B. Geräte, Module, Schnittstellen, Teilfunktionalitäten und jeweils auch ihre Typen, repräsentiert und durch entsprechende Merkmale beschrieben werden. Repräsentiert eine Klasse einen Typen (z.B. Gerätetyp), dann können weitere Typen (z.B. Produkttyp) davon abgeleitet werden. Diese Ableitung erfolgt durch eine ‚is_a' Beziehung.

In der IEC 61987-10 wird ein Beschaffungsprozess definiert, der dem PROLIST-Beschaffungsprozess entspricht.

### *3.4.10.1 Darstellung von Schnittstellen laut IEC 61987-10*

Schnittstellen von Systemen werden im Allgemeinen durch Blöcke von Properties dargestellt.

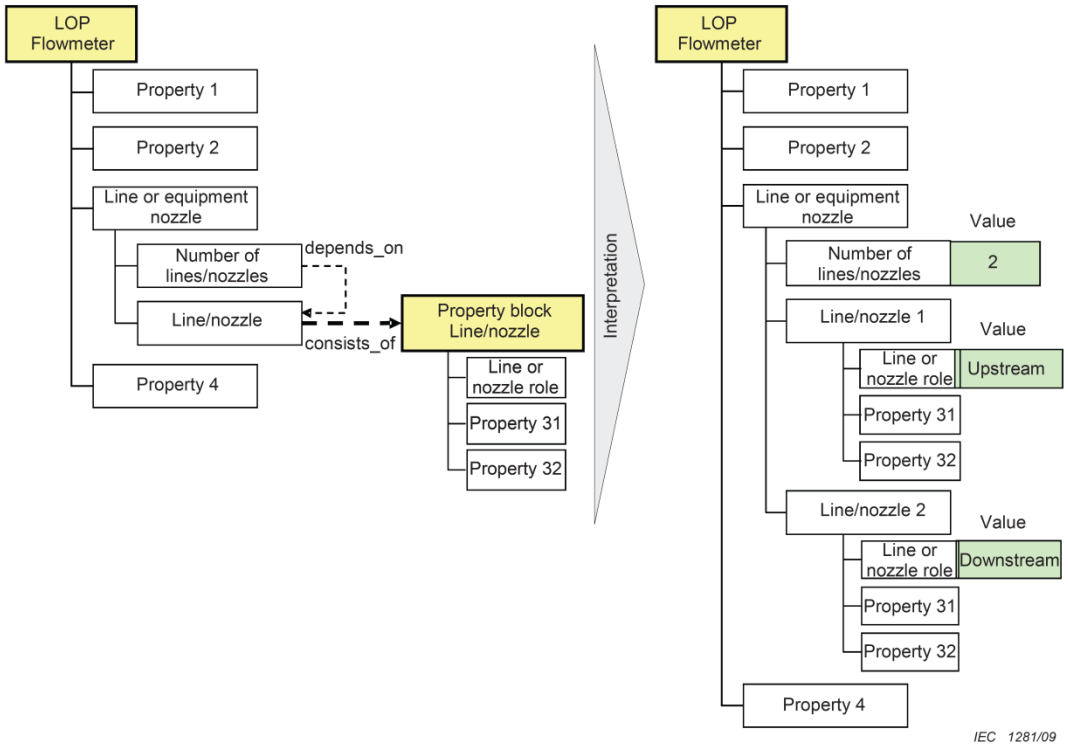


**Abbildung 38 – Erläuterung von Kardinalität in [IEC09a]**

Dabei dient ein Property als Angabe für ‚Kardinalität' des Interfaces (in Abbildung 38 das Property „*Number of lines/nozzles*"). Der Property-Block für das Interface wird entsprechend dieser Kardinalität mehrmals bereitgestellt um jeweils ein Interface zu beschreiben.

### *3.4.10.2 Angestrebte Verwendung von Properties und LOPs nach IEC 61360*

Betrachtet man die Verwendung von Merkmalbeschreibungen für Automatisierungsgeräte in der Industrie, so werden Properties entsprechend IEC 61360-2/ISO 13584-42 allgemein als Basis verwendet. Basierend darauf werden Standard-LOPs definiert, die Geräteprofilen entsprechen (z.B. IEC 61987, IEC 62683). Diese Standard-LOPs sind mit wenigen Property-Werten definiert und werden typischerweise in „Dictionaries" (dt. Wörterbüchern) wie z.B. dem IEC CDD zur Verfügung gestellt. Standard-LOPs dienen als Vorlagen für produktspezifische LOPs. Diese produktspezifischen LOPs werden in Rahmen des Beschaffungsprozess für Produktionssysteme (siehe Abbildung 40) verwendet, um die Fähigkeiten eines Betriebsmittels (Gerät) zu beschreiben. Dabei sind Teile der LOP bereits mit Property-Werten belegt, die den Fähigkeiten des Betriebsmittels entsprechen. Beispiele

für solche Property-Werte sind Messbereichsgrenzen, Zusicherungen über die Zuverlässigkeit usw. Produktspezifische LOPs werden in Hersteller-Bibliotheken (engl. Libraries) zur Verfügung gestellt. Nach dem Kauf des Gerätes (d.h. für eine Geräte-Instanz, die durch eine Seriennummer identifiziert werden kann), sind alle Properties der LOP mit Werten belegt.

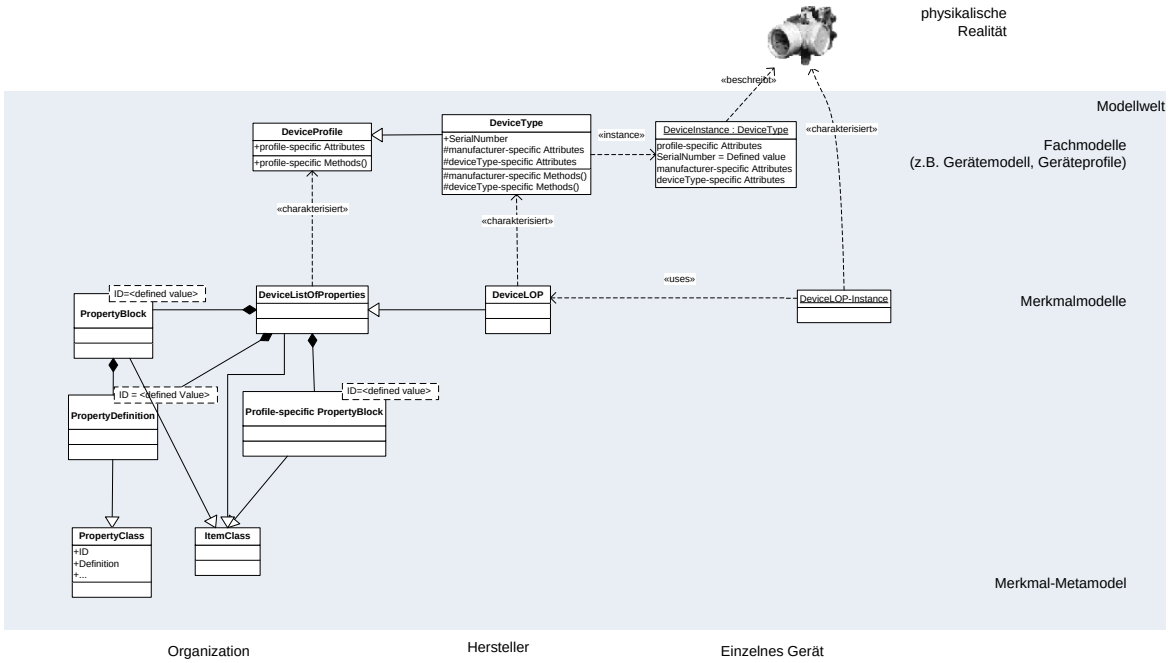


**Abbildung 39 –Wie Properties und LOPs genutzt werden**

Es gibt verschiedene Möglichkeiten diese mit Werten belegte LOP zu implementieren:

- Als LOP (d.h. vollständige Definition der Properties) bei denen auch die Werte der Properties definiert sind.
  Die ausgelieferte Beschreibung beinhaltet dabei die Beschreibung (z.B. Definition) der Properties zusammen mit den entsprechenden Werten, so dass es möglich ist, die Beschreibung auch ohne Rückgriff auf ein Dictionary zu verstehen.
- Als Key-Value-Liste, bei denen der Key jeweils ein Property mittels ID referenziert und der entsprechende Wert des Properties durch den Value dargestellt wird.
  Die ausgelieferte Beschreibung beinhaltet nicht die Beschreibung der Properties, um die Bedeutung der Properties zu verstehen ist ein Rückgriff auf ein entsprechendes Dictionary notwendig um die Informationen zu verstehen.

Löffelmann et al. beschreiben in [LPZ07], wie LOPs im Beschaffungsprozess und im Engineering genutzt werden können (siehe Abbildung 40). Dabei werden unterschiedliche Typen von LOP verwendet, um unterschiedliche Kategorien von Informationen zu transportieren. LOPs werden hauptsächlich in der Beschaffung genutzt um Anforderungen

an Betriebsmittel zu beschreiben, die entsprechenden Fähigkeiten der Betriebsmittel zu dokumentieren und eine Auswahl von Betriebsmitteln auf diesen Informationen zu ermöglichen. Die Verwendung von Property-Beschreibungen im eigentlichen Design-Prozess ist nicht methodisch beschrieben, es wird erwartet, dass ein Ingenieur diese Daten basierend auf seinen Erfahrungen nutzt.

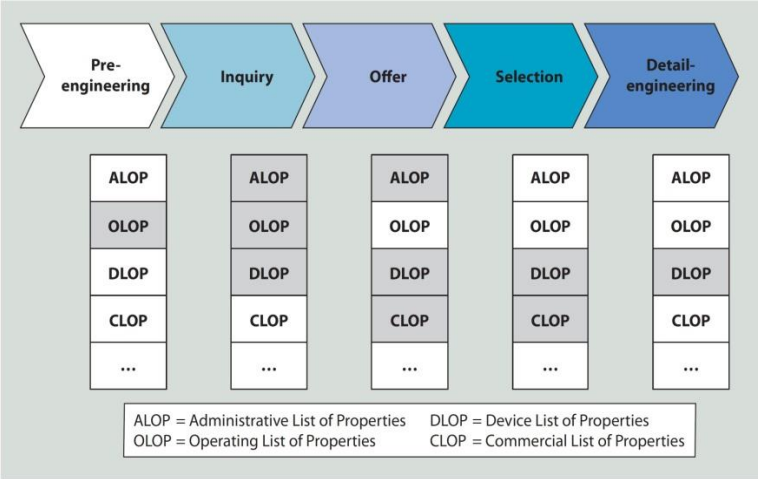


**Abbildung 40 – Nutzung von Merkmalleisten im Engineering-Prozess [LPZ07]**

Für diese Arbeit interessant sind die LOP-Typen OLOP (Operating LOP) und DLOP (Device LOP). Eine OLOP beschreibt Einsatzbedingungen und die Anforderungen an ein Gerät. Eine OLOP beschreibt zum Beispiel eine Mess- bzw. Wirkstelle eines R&I-Diagramms, für welche das passende Betriebsmittel gesucht wird. Eine DLOP beschreibt das Betriebsmittel mit seinen Strukturelementen (z.B. Prozessanschlüsse, Module) und Eigenschaften.

### 3.4.11 ISO 22745

Die Definitionen der ISO 22745 basieren auf den Konzepten der ISO 29002 (siehe Abbildung 41).

ISO 22745 definiert eine Methode zur Beschreibung von Produktdaten, die auf Ontologien basiert. Ein *electronic Open Technical Dictionary* (eOTD) enthält Konzepte (entsprechend der Definition nach ISO/IEC 11179), welche durch ein oder mehrere Bezeichner(Term), einer oder mehreren Definitionen und optional Bildern(Image) beschrieben sind. Konzepte können sein: Klassen, Property, Darstellungsformate(*representation*), Maßeinheiten, Messwertqualifikation(*qualifier of measure*), Property-Werte, Währungen und Datentypen.

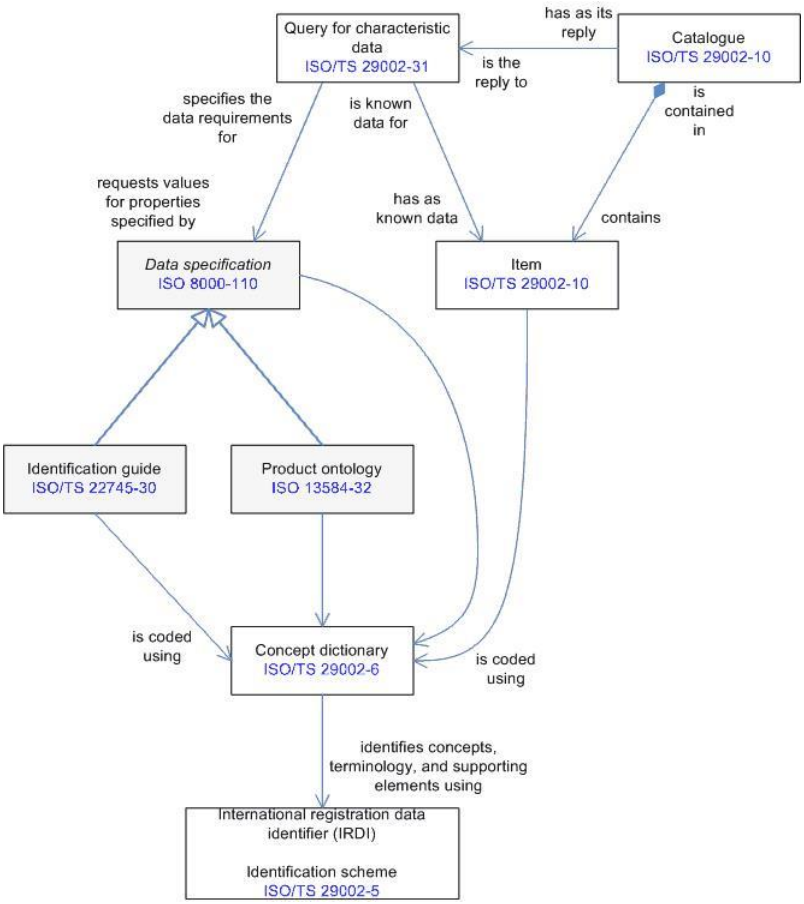


**Abbildung 41 – High-level planning model [ISO09a, S. 9f.]**

Ein eOTD enthält eine Sammlung von Konzepten, die weitgehend unabhängig voneinander definiert werden. Die Beziehungen zwischen Klassen und Properties wird durch „*Identification Guide*“-Dokumente entsprechend ISO 22745-30 [ISO09d] definiert. D.h. dass die Merkmalbeschreibung einer Klasse aus der Sammlung von Klassen und Properties in einem (oder mehr) eOTD und einem entsprechenden *Identification Guide* besteht (siehe Abbildung 42).

Die ECCMA stellt auf ihrer Webseite (http://www.eccma.org/) die Möglichkeit bereit herstellerspezifische eOTD zu registrieren und zu durchsuchen [ECC14a].

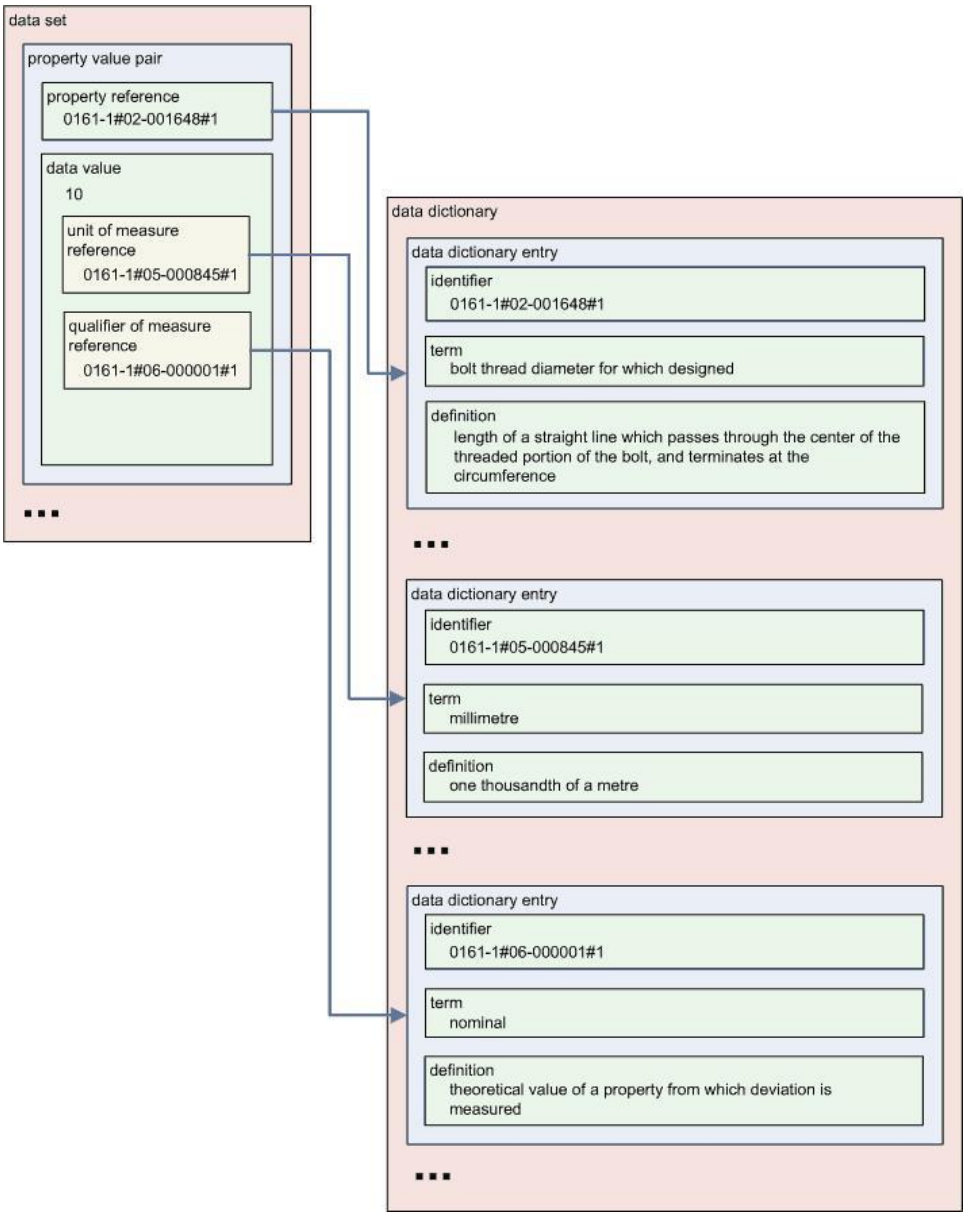


**Abbildung 42 – Beispiel für Referenzierung von Datenwert und Einheit [ISO07]**

Identification Guides werden durch Kunden (Einkäufer) definiert und definieren mit welchen Properties eine Klasse beschrieben werden soll, sowie welche Beschränkungen es hinsichtlich der Properties gibt (z.B. Wertebereiche, gegenseitigen Ausschluss). Für den Austausch von Informationen zwischen Anbietern und Einkäufern wurde ein Prozess zur Kommunikation von Informationen zwischen Anbietern von Equipment und Nutzern von Equipment definiert. Dabei treten die Nutzer als „data requester“ und die Anbieter als „data provider“ auf (siehe Abbildung 43). Bei der Anfrage definiert ein Nutzer, nach welchem „Identification Guide“ die Antwort (engl. *reply*) formatiert sein soll. Der Anbieter (engl. *data provider*) ist dann dafür verantwortlich seine Daten entsprechend zu formatieren. In der Darstellung wird auch gezeigt, dass Daten eines Sub-Lieferanten in die Antwort integriert werden können.

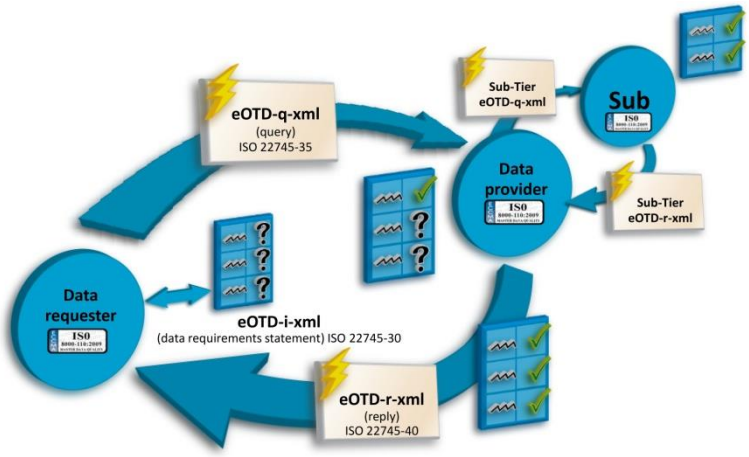


**Abbildung 43 – Informationsaustausch zwischen Lieferanten und Nutzer [Ben12, S. 14f.]**

Ein interessanter Aspekt ist die Unterstützung von Mehrsprachigkeit. Werden gleiche Gegenstände mittels eOTD in unterschiedlichen Sprachen beschrieben, heißt das, dass sie durch unterschiedliche Konzepte repräsentiert werden. Zum Beispiel führt die Suche in der ECCMA-eOTD [ECC14b] nach dem Begriff „Valve“ zu einer Liste von Klassen (siehe Tabelle 10).

**Tabelle 10 – Liste von Konzepten zur Repräsentation von „Valve“ [ECC14b]**

| Term | TermID | Language | Organization | Status |
|---|---|---|---|---|
| ADAPTER ASSEMBLY,VALVE | 0161-1#TM-012074#1 | en - US | DLIS | Active |
| SOVITINYHDISTELMÄ, VENTTIILIN | 0161-1#TM-1016582#1 | fi - FI | NCB FINLAND | Active |
| ADAPTER ASSEMBLY,VALVE | 0161-1#TM-1061937#1 | el - GR | NCB GREECE | Active |
| SESTAVA PŘECHODOVÉHO KUSU, VENTIL | 0161-1#TM-228015#1 | cs - CZ | NCB Czech | Active |
| ADAPTERBAUGRUPPE, VENTIL | 0161-1#TM-299667#1 | de - DE | NCB Germany | Active |
| […] | | | | |

D.h. wenn man das gleiche Gerät in verschiedenen Sprachen beschreibt, muss man nicht nur die Äquivalenz für die Geräteklassen kennen, sondern auch die Äquivalenzen für die Properties die zu der Gerätebeschreibung gehören (siehe Tabelle 11).

**Tabelle 11 – Properties zur Repräsentation von „Anzahl von Sperrklappen“ [ECC14b]**

| Term | TermID | Language | Organization | Status |
|---|---|---|---|---|
| BUTTERFLY VALVE BOX QUANTITY | 0161-1#TM-091268#1 | en - US | DLIS | Active |
| POÈET SKØÍNÍ ©KRTÍCÍHO VENTILU. | 0161-1#TM-285424#1 | cs - CZ | NCB Czech | Active |
| NOMBRE DE CHAMBRES A PAPILLON | 0161-1#TM-391167#1 | fr - FR | DLIS | Active |

### 3.4.12 Vergleich von IEC 61360-basierten und ISO 22745-basierten Ansätzen

Wenn man die Ansätze von IEC 61360 und ISO 22745 vergleicht, erscheint der ISO 22745-Ansatz mächtiger und flexibler, es können verschiedene herstellerspezifische Beschreibungen von Betriebsmitteln unterstützt werden, das Datenformat ist flexibler, es besteht die Möglichkeit Daten von Sub-Lieferanten zu integrieren. Der ISO 22745-Ansatz erlaubt, dass alle von den IEC 61360-Klassen (z.B. ItemClass, DET) abgeleitete Klassen (z.B.

LOP entsprechend IEC 61987) als eindeutig identifizierbare Konzepte im Sinne von ISO 22745 verstanden werden können. Auf der anderen Seite können die Merkmallisten-Definitionen in IEC 61987, IEC 62683 und ähnliche als allgemeingültige „Identification Guide“-Dokumente verstanden werden, die von allen Kunden und Anbieter gleichermaßen eingesetzt werden. Damit entsprechen sich die Ansätze prinzipiell, wobei der Ansatz basierend auf der IEC 61360 durch seine standardisierten Formate schneller von mehr Beteiligten (Anbietern und Kunden) umgesetzt werden kann, zum Beispiel weil gleiche Werkzeuge genutzt werden.

Hervorzuheben ist bei beiden Ansätzen die Fokussierung auf den Beschaffungsprozess. Die in Abbildung 40 dargestellte Nutzung von Daten im Detailed Engineering besteht darin, das ein Ingenieur in der Lage ist, die bereitgestellten Informationen zu verstehen, zu überblicken und sie bei Entwurfsentscheidungen zu berücksichtigen. Die existierenden Werkzeuge unterstützen nur den Beschaffungsprozess. Werden Vergleiche von Merkmalen bzw. von Merkmalausprägungen ausgeführt, so ist das Ziel verschiedene Betriebsmittel miteinander zu vergleichen. Es gibt keine Werkzeuge um Entwurfsentscheidungen im Engineering automatisiert zu unterstützen.

## 3.5 Unterstützung des Engineering-Prozess

Diese Arbeit beschäftigt sich mit dem Engineering von Systemen. Engineering wird dabei als Prozess verstanden, in dessen Verlauf basierend auf einer Anforderungsbeschreibung ein System entwickelt wird. Beim Engineering bewirkt ein soziotechnisches System, bestehend aus Menschen und deren Arbeitsmitteln, die Erstellung eines technischen Systems.

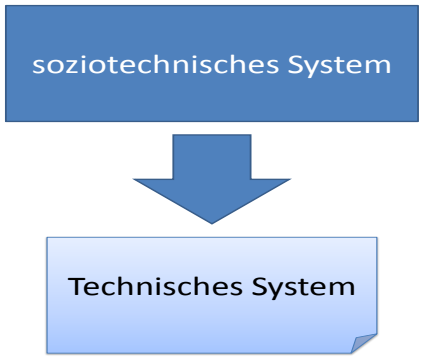


**Abbildung 44 – Beziehung zwischen soziotechnischem und technischem System**

Engineering Prozesse sind Prozesse, die in diesem soziotechnischen System ablaufen, um ein technisches System zu erzeugen. Einen solchen Prozess kann man auch mit der Formalisierten Prozessbeschreibung darstellen (siehe Abbildung 45). Das soziotechnische System besteht dabei im Wesentlichen aus Personen und den eingesetzten Werkzeugen zur Erstellung eines technischen Systems. Im Rahmen des Engineering Prozess gibt es jeweils

Teilprozesse, die bewirken, dass existierende Informationen (P_Info Level x) verarbeitet, erweitert und verfeinert wird, so dass neue Informationen (P_Info Level x+1) entstehen.

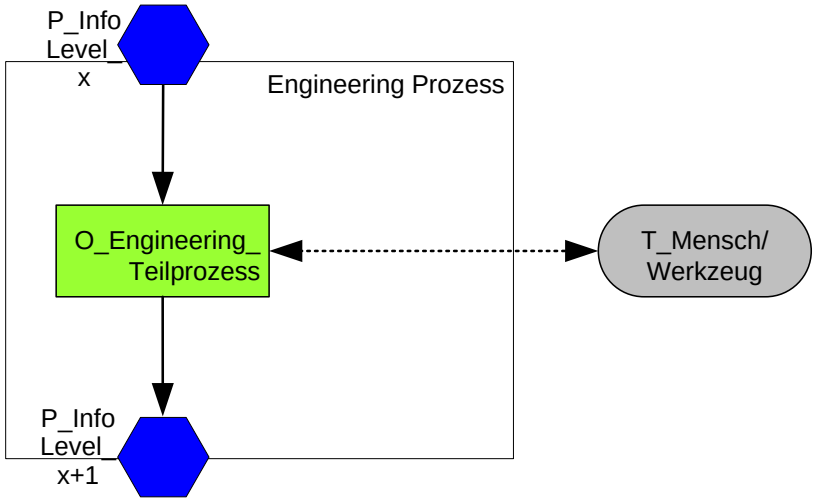


**Abbildung 45 – Engineering–Prozess[8]**

Bezieht man sich auf das BMW-Prinzip nach Schnieder [Sch99], so stellen Werkzeuge und Menschen die Struktur des soziotechnischen Systems dar. Beschreibungsmittel definieren die Regeln für die Produkte des Systems (Systemmodelle bzw. reales System) und Methoden definieren die Regeln für die Abläufe im System (Prozesse).

### 3.5.1 Einordnung in den Anlagenlebenszyklus

In der ISO 15926-1 [ISO04b] wird basierend auf [PI 94] ein Anlagenlebenszyklus definiert, der in Abbildung 46 dargestellt ist.

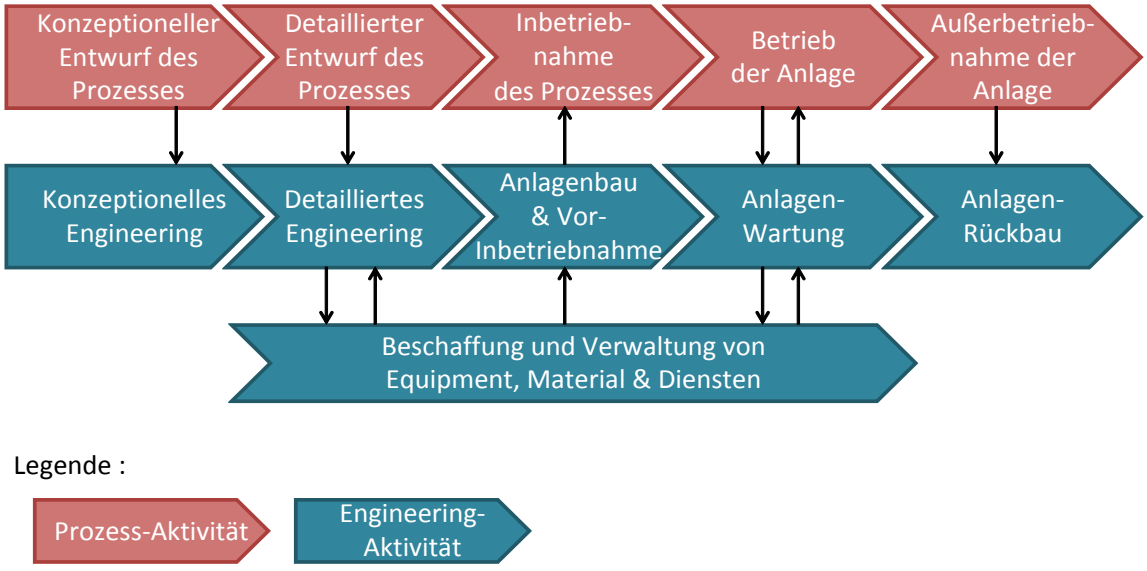


**Abbildung 46 – Anlagenlebenszyklus nach [ISO04b, S. 2f.]**

[8] Abbildung 45 nutzt entsprechend [UGF09] die sechseckige Form zur Darstellung von Informationen.

Bei der Darstellung des Lebenszyklus eines Produktionssystems wird unterschieden zwischen Aktivitäten bezüglich des Produktionsprozesses (Prozess-Aktivität) und Aktivitäten bezüglich des Produktionssystems (Engineering-Aktivität). Prozess-Aktivitäten werden üblicherweise von Spezialisten für den entsprechenden Herstellungsprozess ausgeführt. Engineering-Aktivitäten können von Systemherstellern und anderen Dienstleistern ausgeführt werden. Die Darstellung der ISO 15926-1 hier ist verkürzt und es soll nur auf die wesentlichen Engineering-Aktivitäten eingegangen werden.

Entsprechend den Definitionen des PI STEP Konsortiums ist der konzeptionelle Entwurf des Produktionsprozesses der Ausgangspunkt für das Engineering und den detaillierten Entwurf des Produktionsprozesses. Basierend auf diesem Entwurf erfolgt ein konzeptioneller Engineering-Entwurf (auch Basic Engineering genannt), in dem grundlegende Engineering-Konzepte festgelegt werden und ein Grobentwurf des Produktionssystems erfolgt. Der detaillierte Entwurf des Produktionsprozesses definiert die Spezifikationen, Anforderungen, anzuwendende Standards, Entwurfs- und Betriebsphilosophien, welche für die weiteren Aktivitäten benötigt werden. Im detaillierten Engineering werden detaillierte konstruktive, mechanische, elektrotechnische, mess- und steuerungstechnische Entwürfe erstellt, welche z.B. Spezifikationen, Modelle, Zeichnungen und Materiallisten umfassen. Beschaffung und Verwaltung von Ausrüstung und Material beinhaltet Aktivitäten, wie Anfragen und Bestellungen platzieren, Warenlieferungen verwalten und prüfen, Bestände zu kontrollieren. In der Aktivität „Anlagenbau und Vor-Inbetriebnahme“, wird die Anlage errichtet, so wie sie im detaillierten Engineering spezifiziert wurde. Dabei wird auch verifiziert, ob die Anforderungen erfüllt werden. Anschließend kann der Produktionsprozess in Betrieb genommen werden und das Produktionssystem an den Betreiber übergeben werden. Nach erfolgreicher Übergabe erfolgt der Betrieb des Produktionssystems. Parallel zum Betrieb des Produktionssystems erfolgt die Wartung und Instandhaltung des Produktionssystems. Das Produktionssystem wird betrieben bis zur Außerbetriebsetzung des Systems. Das heißt es erfolgt eine Außerbetriebnahme des Produktionsprozesses (welche prinzipiell wieder eine Inbetriebnahme des Produktionsprozesses erlauben würde) und das Produktionssystem wird rückgebaut. Beim Rückbau können Elemente des Produktionssystems für eine weitere Verwendung entnommen werden [PI 94, S. 7f.].

Alle in dieser Arbeit diskutieren Entwicklungs- und Engineering-Prozesse können in die hier beschriebene Sequenz der Engineering-Aktivitäten entsprechend Abbildung 46 eingeordnet werden.

### 3.5.2 Grundsätzliches Vorgehensmodell

Vorgehensmodelle sind Prozessbeschreibungen (siehe 2.3.1.4.1) und beschreiben Abläufe im soziotechnischen System (siehe Abbildung 44), die zur Erstellung des technischen Systems führen. D.h. Vorgehensmodelle sind funktionale Modelle des soziotechnischen Systems.

Im Rahmen der Planung und Umsetzung von automatisierten Systemen werden in verschiedenen Phasen des Engineerings verschiedene Beschreibungen der Anlage bzw. des Automatisierungssystems mit unterschiedlichem Abstraktionsgrad verwendet.

Die unterschiedlichen Abstraktionsgrade werden durch Darstellung unterschiedlicher Sichtweisen mit verschiedenem Detaillierungsgrad erreicht.

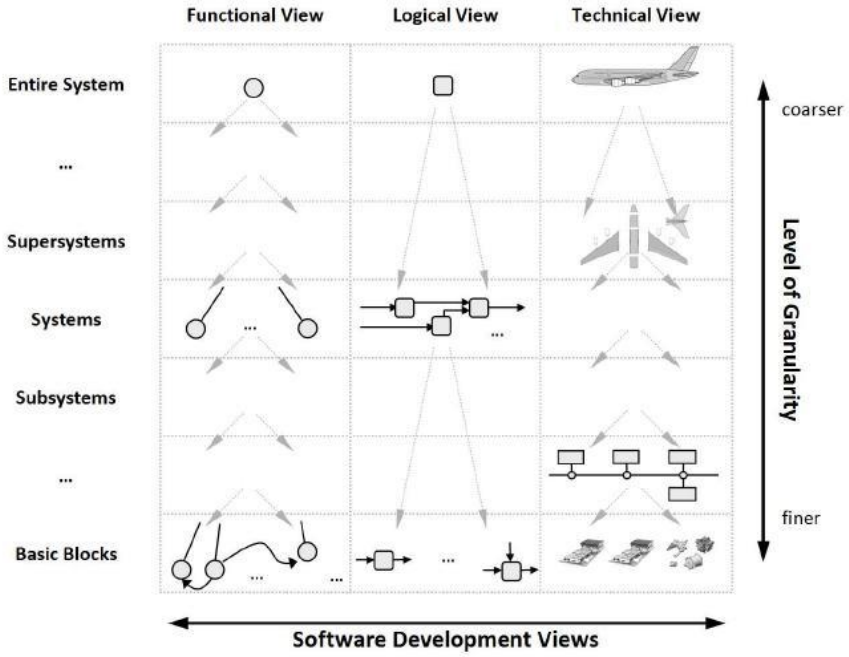


**Abbildung 47 – Abstraktionsgrade [RST09, S. 6f.]**

Abbildung 47 illustriert, dass unterschiedliche Grade der Abstraktion für unterschiedliche Sichten auf das System existieren. Sowohl funktionelle Modelle (Funktional View), als auch Strukturmodelle (Logical View als Strukturmodell der Software, Technical View als Strukturmodell der Hardware) können auf verschiedenen Abstraktionsebenen verwendet werden.

Im Laufe des Entwurfs eines Produktionssystems werden ausgehend von einem grobkörnigen Modell des Systems durch wiederholte Verfeinerung verschiedene Modelle des Systems entwickelt, die das System mit feinerem Detail darstellen.

### 3.5.3 Verfeinerung (Refinement)

Im Verlauf des Entwurfs erfolgt eine Verfeinerung des Systemmodells. Broy beschreibt den Entwicklungsprozess basierend auf drei Konzepten der ‘Verfeinerung’ [BS01]:

- behavioral refinement
- interface refinement
- conditional refinement

Diese Verfeinerungen werden in jedem Entwurfsschritt ausgeführt. Dabei entstehen aus den existierenden abstrakteren Modellen detailliertere Modelle (oft basierend auf anderen Modellsprachen). Zum Beispiel werden existierende Modellelemente durch neue konkretere Modellelemente ersetzt (z.B. Funktionen durch Komponenten, Komponenten durch Gerätetypen, Gerätetypen durch Produkttypen) und existierende Elementbeziehungen werden konkretisiert.

Diese Verfeinerungen betreffen nicht nur die Systemelemente sondern auch die Verbindungen zwischen den Systemelementen. In funktionalen Modellen wird beschrieben, welche Objekte zwischen den Prozessschritten fließen. In Bezug auf automatisierungstechnische Elemente werden dabei oft nur Bezeichner (z.B. für Messstellen) definiert. In Komponentenmodell und Topologie-Modell wird diese Information um zusätzliche Attribute, zum Beispiel Datentyp, Maßeinheit, Wertebereich und ähnlichem erweitert.

Da es sich bei Verfeinerung um wichtige Design-Entscheidungen handelt, ist es erstrebenswert diese Entscheidungen zu dokumentieren, so dass sie später nachvollzogen werden können. In einigen Entwurfsverfahren (z.B. UML) werden diese Zusammenhänge durch Beziehungsverhältnisse dokumentiert (Komponente(n) realisiert Funktion, Gerätetyp realisiert Komponente, Produkttyp erbt von Gerätetyp). D.h. es werden Beziehungen zwischen Modellelementen verschiedenartiger Systemmodelle dokumentiert um Designentscheidungen später nachvollziehen zu können und um die Äquivalenz der verschiedenartigen Modelle darzustellen. (Siehe auch 2.3.3).

### 3.5.4 CAEX-Engineering-Prozess

CAEX [IEC08] wurde als Datenformat zum Austausch von Daten zwischen verschiedenen Engineering-Werkzeugen im Engineering-Prozess entwickelt. Der Engineering-Prozess mit CAEX ist in [Dra10, S. 52f.] folgenderweise beschrieben:

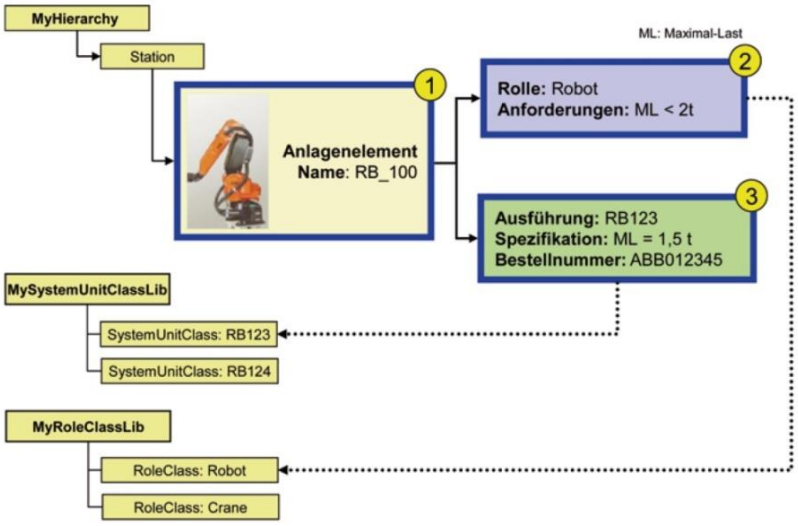


**Abbildung 48 – Entwurfsprozess laut [Dra10, S. 53f.]**

Um ein System zu entwickeln, wird in einem ersten Schritt (1) ein Systemmodell mit abstrakten Elementen (<InternalElement>) erstellt. Diesen abstrakten Elementen wird im zweiten Schritt (2) jeweils eine Rolle zugewiesen. Diese Zuweisung erfolgt mittels <RoleRequirements/RefBaseRoleClassPath>-Verweis auf eine Rollendefinition, welche eine Funktion bzw. die Anforderungen an das Systemelement beschreibt. Dieses abstrakte Modell kann auch als funktionales Modell betrachtet werden. Anschließend wird im dritten Schritt (3) die Ausführung dieser abstrakten Elemente mit je einem konkreten Betriebsmittel definiert, dies erfolgt indem mittels dem Attribut <@refBaseSystemUnitPath> jedem Element jeweils eine Referenz auf eine Typ-Definition für eine <SystemUnitClass> hinzugefügt wird, welche die Implementierung beschreibt [IEC08, S. 50, Dra10, S. 53f.].

Dieser auch in Abbildung 49 dargestellte Entwurfsprozess geht davon aus, dass der funktionale Entwurf abgeschlossen ist und dass die Elemente des funktionalen Modells bereits der zukünftigen Systemstruktur entsprechen (d.h. pro Funktion ein Systemelement). Wenn mehrere Funktionen durch ein Systemelement (z.B. eine Fräse, die auch zum Bohren eingesetzt wird) oder eine Funktion durch mehrere Systemelemente realisiert wird, dann kann dieses 1:1 Mapping nicht genutzt werden.

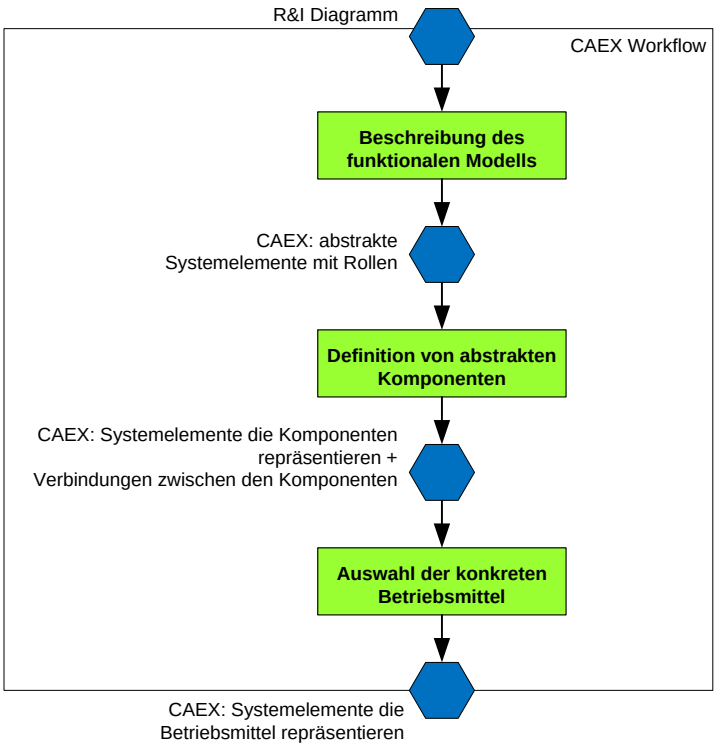


**Abbildung 49 – CAEX Entwurfsprozess[9]**

Eine schrittweise Verfeinerung ist möglich durch Ableitung und Detaillierung von Klassen innerhalb der Klassenbibliotheken (für Rollen-Klassen und Systemelement-Klassen) bzw. durch Detaillierung des eigentlichen Modells.

In CAEX ist es möglich mehrere Systemmodelle in einem CAEX-Dokument nebeneinander zu beschreiben (siehe auch 3.3.1). Damit wäre es möglich, sowohl Systemmodelle aus unterschiedlichen Phasen des Lebenszyklus abzubilden (z.B. funktionales Modell, Komponentenmodell, Netzwerkmodell) als auch verschiedene Systemmodelle auf gleichem Entwurfsstadium (SW-Modell, HW-Modell, mechanisches Modell) darzustellen. Dies ist aber im Entwurfsprozess von [Dra10, S. 52f.] nicht vorgesehen, sondern es wird davon ausgegangen, dass es ein Modell gibt, welches schrittweise verfeinert wird und welches alle Informationen zu dem System enthält.

### 3.5.5 Entwicklungsprozess mit SysML

Weilkiens beschreibt in [Wei07, S. 26f.] einen Entwicklungsprozess für Systeme, der aus zwei Teilprozessen besteht: Analyse und Umsetzung der Anwendungsfälle.

[9] Abbildung 49 nutzt entsprechend [UGF09] die sechseckige Form zur Darstellung von Informationen.

Die Analyse wird als Prozess dargestellt in dem die Anforderungen an das System ermittelt werden und in dem ein funktionales Modell sowie eine Beschreibung der Randbedingungen (engl.: *system context*) erstellt werden. Das funktionale Modell besteht dabei aus einer Beschreibung der Anwendungsfälle mit entsprechenden System-Prozessen (modelliert z.B. als Aktivitätsdiagramme).

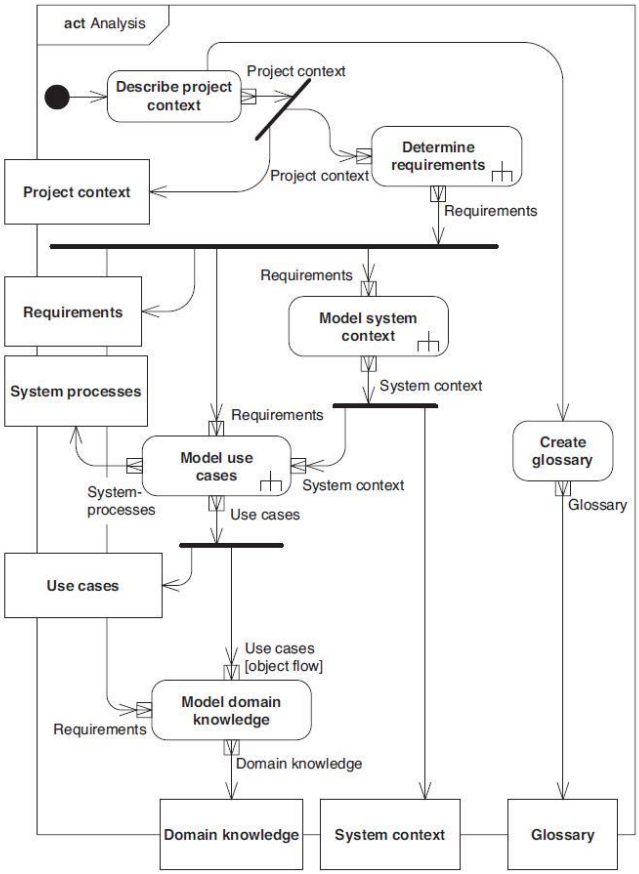


**Abbildung 50 – Analyse [Wei07, S. 25f.]**

Ausgehend von den Ergebnissen der Analyse erfolgt die Entwicklung des Systemmodells. Erst werden die Interaktion des Systems mit der Umwelt modelliert und entsprechende System-Schnittstellen entwickelt, danach die Systemstruktur und das Systemverhalten.

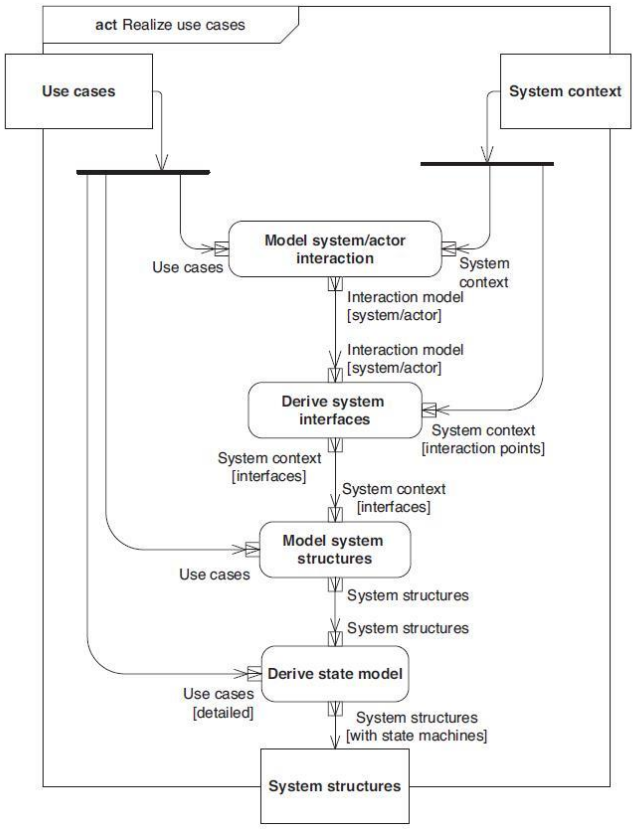


**Abbildung 51 – Umsetzung von funktionalen Modellen [Wei07, S. 26f.]**

Den kompletten Entwicklungsprozess kann man verallgemeinern mit:

- Basierend auf den Anforderungen an das System wird ein funktionales Modell des Systems (in Form von Use-cases bzw. als Prozessbeschreibung) erstellt
- Es wird eine Grobstruktur des Systems definiert
    - Schnittstellen des Systems mit der Umgebung
    - Elemente des Systems
- Die Feinstruktur des Systems wird definiert
    - Elemente und Schnittstellen werden verfeinert
- Systemverhalten wird definiert
    - Verhalten der Elemente

Interessant an dem dargestellten Entwurfsprozess ist, dass das funktionale Modell das Ergebnis der Analyse der Anforderungen ist und dass dieses Modell Ausgangspunkt für den Entwurf des Systems ist. In der Darstellung fehlt die Information, dass die einzelnen Schritte iterativ erfolgen und dass die Entwurfsergebnisse in Bezug auf Korrektheit und in Bezug auf die Erfüllung von Anforderungen geprüft werden müssen.

## 3.6 Zusammenfassung

Bei der Darstellung des aktuellen Stands der Wissenschaft und Technik wurde auf Systemmodelle und Merkmalmodelle fokussiert. CAEX als besonders offene und breit anwendbare Beschreibungssprache für Systeme wurde besonders detailliert dargestellt. Da das Ziel dieser Arbeit eine Unterstützung des Engineerings von Systemen ist, wurde weiterhin auf übliche Engineering-Prozesse eingegangen.

Das Engineering von Systemen ist in verschiedene Phasen gegliedert, in denen verschiedene Werkzeuge und unterschiedliche Modelle verwendet werden. Obwohl mit CAEX ein Format existiert, das den Datenaustausch zwischen verschiedenen Engineering-Werkzeugen ermöglicht, gibt es keine durchgehende Kette von Informationen von frühen Phasen des Engineerings bis hin zum Betrieb des Systems. Wesentliche Informationen (z.B. über die Ziele beim Entwurf des Systems) gehen dabei verloren. Obwohl standardisierte Daten-Modelle (z.B. basierend auf IEC 61360) existieren, werden diese Datenmodelle nicht genutzt um eine gemeinsame Datenbasis in den verschiedenen Phasen des Engineerings zu gestalten.

Generell erfolgt beim Entwurf von Produktionssystemen ausgehend von den Anforderungen die Entwicklung von verschiedenen Modellen des Systems. Im ersten Schritt wird typischerweise ein funktionales Modell des Systems entwickelt, das die Basis für die Entwicklung eines Strukturmodell des Systems bietet. Das Strukturmodell des Systems wird schrittweise verfeinert, wobei die Schnittstellen des Systems zu seiner Umgebung und die Schnittstellen der Systemelemente definiert werden. Nach der Definition der Struktur eines Systems wird das Verhalten des Systems bzw. seiner Systemelemente definiert.

Im Laufe des Anlagenlebenszyklus werden verschiedene Engineering-Werkzeuge genutzt, um diese Schritte zu vollziehen (siehe Abbildung 52[10]). Dabei entstehen verschiedene Systemmodelle. Die Darstellung zeigt, dass verschiedene Werkzeuge genau den unterschiedlichen Lebensabschnittsphasen entsprechen. Dies ist jedoch nur eine schematische, vereinfachte Darstellung. In Realität können einzelne Werkzeuge in verschiedenen Lebenszyklusphasen eingesetzt werden und mehrere verschiedene Werkzeuge können in einer Lebenszyklusphase eingesetzt werden.

[10] In der Abbildung wurde „Konzeptionelles Engineering“ auf „Anforderungsanalyse“ und „Konzept Design“ aufgeteilt.

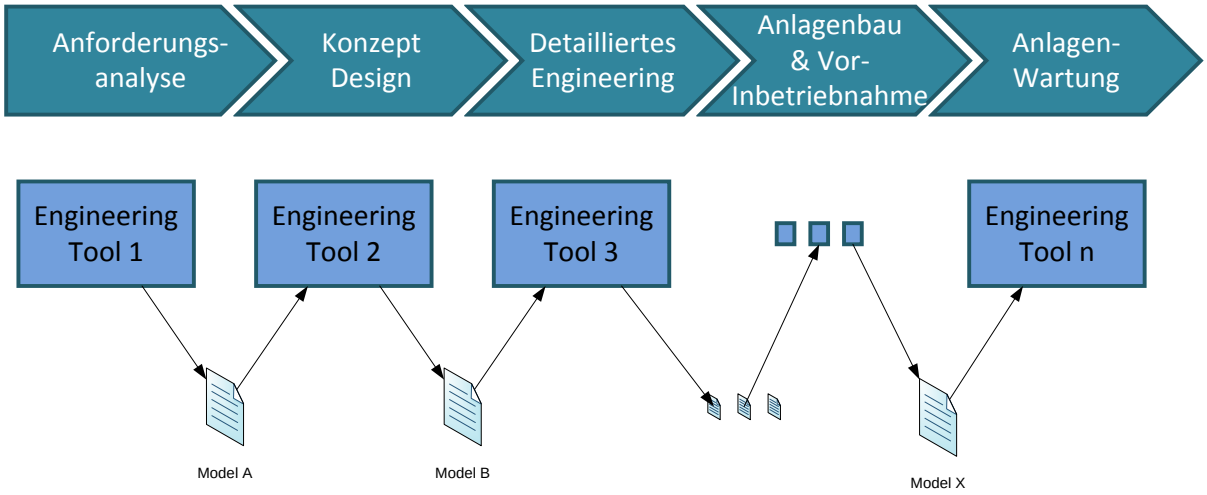


**Abbildung 52 – Genereller Workflow im Engineering**

Um die Beziehung zwischen den verschiedenartigen Modellen zu verstehen, müssen sie dokumentiert werden. Diese Dokumentation sollte nicht nur die Modelle an sich beinhalten, und nicht nur die Beziehung zwischen den Modellen, sondern auch die Beziehung zwischen den Elementen der verschiedenartigen Modelle.

Ausgehend von diesem aktuellen Stand der Wissenschaft und Technik wird in Kapitel 4 ein Konzept zur Integration von Systemmodell und Merkmalmodell zur Nutzung im System-Engineering dargestellt. Kapitel 5 schlägt eine Weiterentwicklung von CAEX vor, welche die Umsetzung des in Kapitel 4 entwickelten Konzepts unterstützt. Kapitel 6 zeigt die Anwendung des Konzepts.

# 4 Integration von Systemmodell und Merkmalmodell

## 4.1 Ziel der Integration

Nachdem immer mehr und ausführlichere Merkmalbeschreibungen für industrielle Betriebsmittel existieren (siehe 3.3), gibt es Bestrebungen diese ausführlichen Information für das Engineering von Produktionssystemen zu nutzen (z.B. eCl@ss CAX Gruppe, IEC TC65 WG16 Digitale Fabrik). Es soll eine Methode vorgestellt werden, die Entscheidungen im Engineering-Prozess unterstützt und die es ermöglicht Entwurfsergebnisse zu verifizieren. Damit Merkmalmodelle für das System Engineering genutzt werden können, ist es notwendig die Beziehung zwischen Systemmodellen und Merkmalmodell zu klären. Ausgehend von dieser Beziehung wird dargestellt, wie Merkmale für die Funktionsbeschreibung und für den Strukturentwurf genutzt werden.

Das Ziel dieser Arbeit ist es einen Entwurfsprozess zu unterstützen, in dem auf verschiedenen Stufen verschiedene Modelle parallel verwendet werden können. Diese verschiedenen Modelle sollen auf einem einheitlichen Datenmodell basieren und die relevante Information in die verschiedenen Modelle überführbar sein. Das einheitliche Datenmodell wird hier definiert.

Ausgangspunkt des Entwurfsprozesses ist ein Funktionsmodell des Systems, welches zu einem Komponentenmodell umgesetzt wird. Das Komponentenmodell wird schrittweise verfeinert (zu einem Topologie-Modell) und aus dem Funktionsmodell des Systems wird ein Softwaremodell für die Steuerung des Systems abgeleitet. Software- und Hardware-Modell der automatisierungstechnischen Komponenten werden zu einem Deployment-Modell kombiniert.

## 4.2 FAVA: V-Modell für Automatisierungssysteme

Das FAVA Vorgehensmodell wurde im Rahmen eines gemeinsamen Projektes der OvG Universität Magdeburg sowie der HSU Hamburg und der TU München entwickelt. ‚FAVA' steht für Funktionaler Anwendungsentwurf für verteilte Automatisierungssysteme.

Ziel des Projektes war die Entwicklung einer systematischen Methodik zum Entwurf von verteilten Automatisierungssystemen. Ein wesentliches Ergebnis des Projektes ist die Definition eines Vorgehensmodells zum Entwurf von verteilten Automatisierungssystemen (siehe Abbildung 53). Das FAVA-Vorgehensmodell orientiert sich am V-Modell bzw. am V-Modell XT, wobei ein Fokus auf den Entwicklungszweig des V-Modells gelegt wurde. Die entwickelte Methodik basierte darauf Merkmale zur Definition von nicht-funktionalen

Anforderungen (z.B. zeitliche Anforderungen) zu verwenden und deren Erfüllung anhand von Merkmalen der Systemelemente (z.B. des Kommunikationssystems) zu verifizieren. Dazu wurde eine Notation basierend auf SysML entwickelt [Fra14][Fra14]. Weiterer Bestandteil der Methodik ist die Auswahl von Lösungsmustern anhand von Merkmalen [HDE+11, HHD+12, FEH+12, FHE+12a, FEH+13, EFH+12, FVF+15].

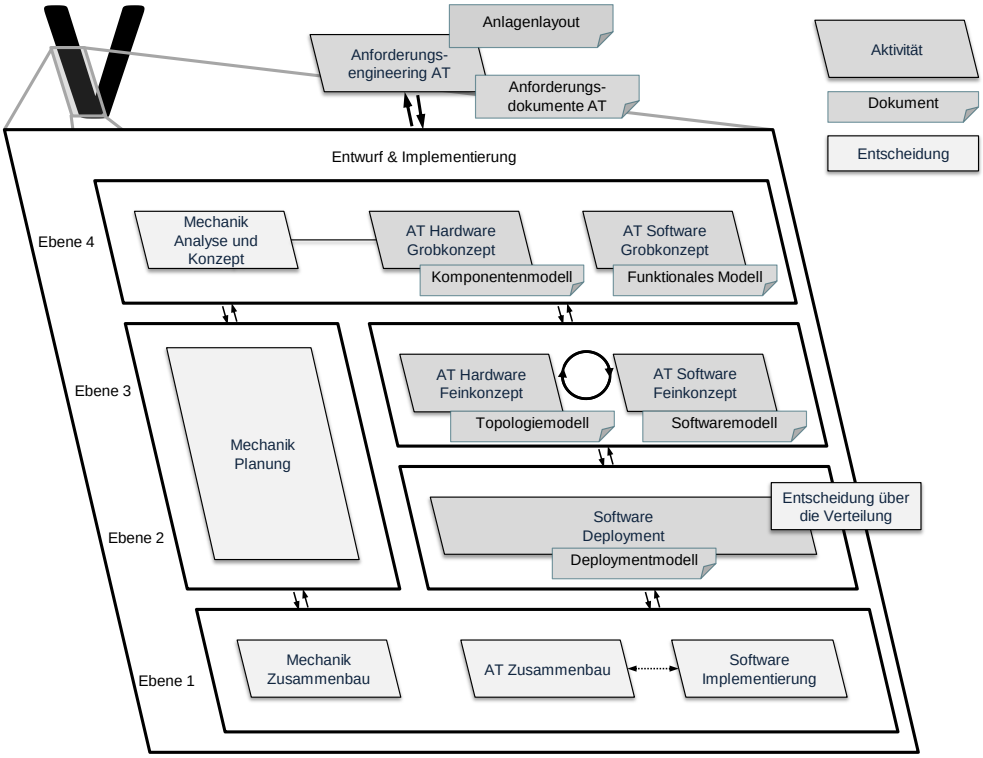


**Abbildung 53 – FAVA Vorgehensmodell [FHE+12a]**

Das für die Automatisierungstechnik angepasste Vorgehensmodell (siehe Abbildung 53) gliedert sich in 3 Säulen: Entwicklung der Mechanik, Entwicklung der automatisierungstechnischen Hardware (AT Hardware) und Entwicklung der automatisierungstechnischen Software (AT Software). Wie in der Abbildung links oben angedeutet, fokussiert das FAVA-Vorgehensmodell auf den Entwicklungszweig des V-Modells und dort speziell auf die Säulen für AT Hardware und AT Software. Ausgangspunkt ist ein Anforderungsengineering (bzw. –analyse) für die Anlage, bei dem Anforderungsdokumente für die Automatisierungstechnik(AT) erstellt werden. Diese Anforderungsanalyse geht von funktionalen bzw. strukturellen Modellen der Anlage insgesamt aus (z.B. das Anlagenlayout). Basierend auf den Anforderungsdokumenten AT erfolgt ein Entwurf des Grobkonzepts für die automatisierungstechnische Hardware und Software. Dabei entsteht ein Komponentenmodell für die Hardware und ein funktionales Modell der AT-Software. Diese Modelle werden weiter verfeinert zu einem detaillierten Hardwaremodell (z.B. Topologiemodell, Geräteliste, PLT-Stellenplan) und einem detaillierten

Softwaremodell. Ein Topologiemodell stellt die Kommunikationsstruktur eines verteilten Systems dar. Ein PLT-Stellenplan zeigt welche AT-Geräte wie verbunden sind. Die Verfeinerung der Modelle kann in mehreren Schritten erfolgen (z.B. durch Iteration). Nachdem detaillierte Modelle für Hardware und Software erstellt wurden, muss in einem verteilten System die Software (im FAVA Vorgehensmodell repräsentiert durch Funktionsblöcke) auf die Hardware (intelligente Knoten) verteilt werden. Dieser Schritt wird im FAVA Vorgehensmodell als Deployment bezeichnet, das Resultat dieses Schritts ist das Deploymentmodell. Die anschließende Implementierung (also Beschaffung und Installation der AT Hardware und Implementierung der Software) ist nicht Bestandteil der FAVA Vorgehensweise. Auf jeder Ebene des Vorgehensmodells ist jedoch vorgesehen, dass entsprechende Test- und Prüfspezifikationen erstellt werden [FEH+12, FHE+12a, FEH+13].

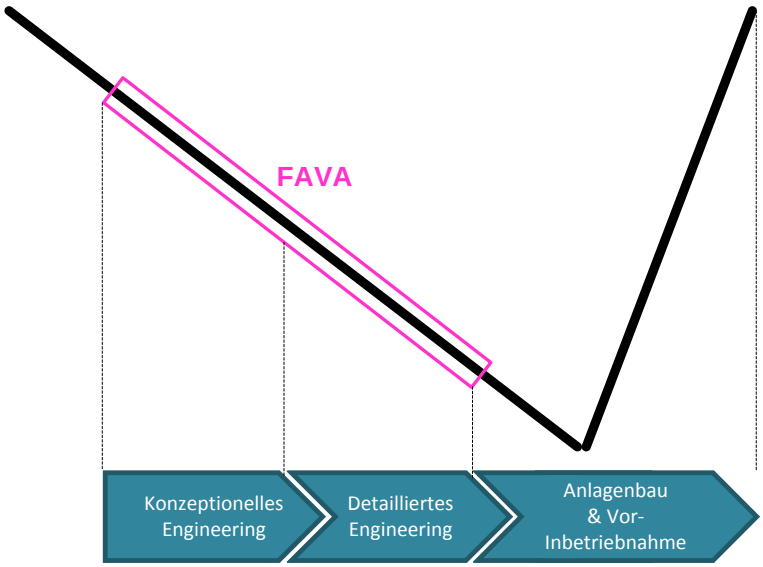


**Abbildung 54 – Einordnung von FAVA in den Anlagenlebenszyklus**

Abbildung 54 stellt dar, welchen Bereich der Entwicklung im Rahmen des Anlagenlebenszyklus nach IEC 15926 abdeckt (siehe 3.5.1). Prinzipiell wurden nur die Phasen „Konzeptionelles Engineering“ (Basic Engineering) und „Detailliertes Engineering“ (Detailed Engineering) betrachtet. Die entwickelten Methoden zur Verifikation des entwickelten Systems in Bezug auf die Anforderungen wurden als unterstützende Maßnahme in der Entwicklung betrachtet und nicht als Bestandteil des Test- und Integrationszweiges des V-Modells.

Da das FAVA Vorgehensmodell speziell für verteilte Systeme entwickelt wurde, sind Maßnahmen und Aktivitäten speziell für diese Art von Systemen vorgesehen, z.B. indem Kommunikationsanforderungen speziell berücksichtigt werden [HHD+12] oder indem eine Verteilung der Software-Elemente auf Knoten berücksichtigt werden. Man kann das

Vorgehensmodell aber auch generell für die Entwicklung von Automatisierungssystemen anwenden (siehe Abbildung 55). Das Grundprinzip ist eine schrittweise Verfeinerung der Systemmodelle basierend auf den Ergebnissen des vorhergehenden Entwurfsschritts.

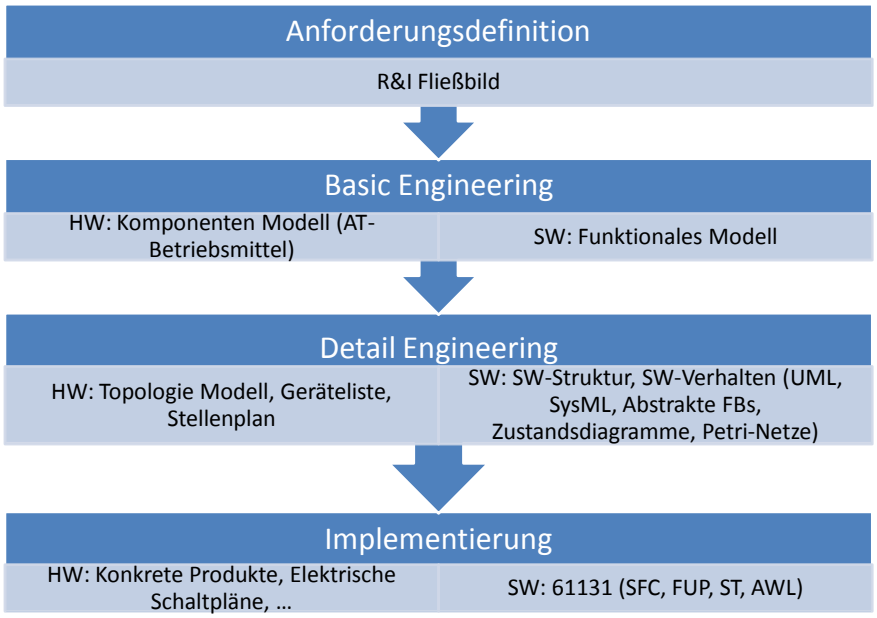


**Abbildung 55 – Entwurfsschritte mit Zwischenergebnissen (basierend auf FEH+12)**

Im FAVA-Entwicklungsprozess erfolgt die Dokumentation der Beziehungen zwischen den verschiedenen Modellen durch die Darstellung der verschiedenartigen Modelle als Ebenen des Systemmodells und durch Anordnung der Modellelemente zueinander. Kontaktpunkte werden genutzt um äquivalente Elemente in verschiedenen Modellen darzustellen [FHE+12b].

### 4.2.1 Im FAVA Vorgehensmodell verwendete Modelle der Anlage

In dem Engineering-Prozess nach [FEH+12] werden verschiedene Modelle verwendet. Abbildung 56 stellt den Informationsfluss zwischen den verschiedenen Modellen als Abhängigkeit dar.

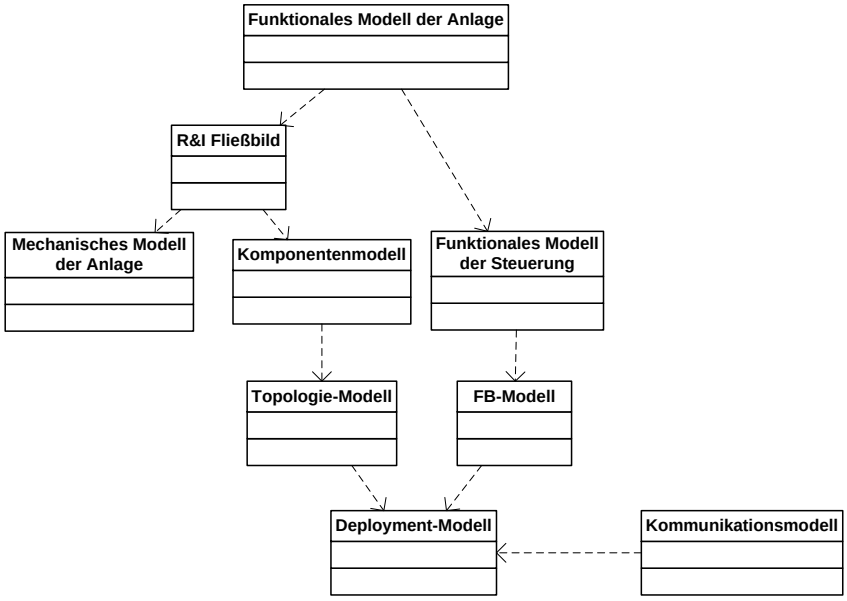


**Abbildung 56 – Modelle im Engineering-Prozess**

Funktionales Modell der Anlage, und die daraus abgeleiteten Modelle, sowie das Modell der Kommunikation sind Systemmodelle. Das funktionale Modell der Anlage beschreibt den angestrebten Produktionsprozess. Daraus wird das R&I-Fließbild abgeleitet, das prinzipiell die Komponenten des Produktionssystems und die Messstellen sowie Wirkstellen beschreibt. Die Elemente des Produktionssystems werden im mechanischen Modell der Anlage verfeinert dargestellt, basierend auf diesem Modell werden die mechanischen Bestandteile des Produktionssystems entwickelt. – Dieser Entwicklungsstrang ist außerhalb des Betrachtungsraums von FAVA. Mess- und Wirkstellen des R&I-Fließbild dienen als Ausgangspunkt für die Entwicklung der AT-Hardware. Es wird dabei ein Komponentenmodell für die AT-Hardware entwickelt, welches zu einem Topologie-Modell verfeinert wird. Parallel dazu wird, basierend auf dem funktionalen Modell der Anlage, die AT-Software entwickelt. Ausgehend von einem funktionalen Modell der Software wird ein Funktionsblock-Modell (FB-Modell) der Software entwickelt. Im nächsten Schritt werden basierend auf den Regeln des Kommunikationsmodells (siehe [HDE+11]) die Software-Elemente (FBs) auf die Hardware-Elemente verteilt, das Ergebnis ist ein Deploymentmodell. Die Bestandteile dieser Modelle werden durch Merkmale charakterisiert [FEH+12].

Es kann beobachtet werden, dass im Engineering-Prozess die Komplexität der Modelle mit jedem Schritt zunimmt. Diese Zunahme an Komplexität beruht sowohl auf dem Refinement der Modelle (Zunahme an Details, siehe 3.5.3) als auch auf der Einbeziehung zusätzlicher Anforderungen. Das führt dazu, dass man basierend auf der Komplexität des Modells in einer Entwurfsstufe bereits Vorhersagen über die Komplexität der nächsten Stufe machen

kann. Ein Ansatz zur Betrachtung von Komplexitäten in Engineering-Modellen wurde in [HSD13] dargestellt.

Moderne Entwicklungsprozesse dokumentieren die Entwicklung von weniger detaillierten Modellen zu stärker detaillierten Modellen, um Entwurfsentscheidungen nachzuvollziehen bzw. um das Entwurfsergebnis im Vergleich mit den Modellen vergleichen zu können [Ber11, KP11, SBD+10]. Dazu ist es notwendig nicht nur die Beziehungen zwischen den Modellelementen eines Modells darstellen zu können, sondern auch Beziehungen zwischen Modellelementen verschiedener Modelle (z.B. „is-a“, „inherits_from“, „is_abstraction_of“, „is_implementation_of“). Diese Beziehungen zwischen Modellelementen verschiedener Modelle, entstehen durch Design-Entscheidungen im Entwurfsprozess, z.B. wenn entschieden wird, wie eine Funktion durch eine Komponente realisiert wird, oder wenn entschieden wird, durch welches Betriebsmittel eine Komponente realisiert wird. Die Verfolgung dieser Designentscheidungen wird im Allgemeinen mit dem Begriff Rückverfolgbarkeit (Traceablility) bezeichnet (für mehr Informationen siehe auch [ISO09c, Bal11]).

Werden solche Beziehungen zwischen Elementen verschiedener Modelle nicht explizit dokumentiert, kann es dazu führen, dass diese Beziehungen später aufwändig rekonstruiert werden müssen. Ein Ansatz zur Rekonstruktion solcher Beziehungen wird z.B. in [HMD10] und [HED+12] präsentiert.

### 4.2.2 Nutzung von Merkmalen in einem Entwicklungsschritt

Entsprechend 4.2 besteht der Entwicklungsprozess aus mehreren aufeinanderfolgenden Entwurfsschritten bei denen jeweils ein Refinement des Modells erfolgt.

In Abbildung 57 wird ein solcher Entwurfsschritt dargestellt. Dabei wird ausgehend von Anforderungen bzw. von den Ergebnissen des vorhergehenden Entwurfsschrittes eine Lösung entwickelt. Diese Lösung wird dargestellt durch ein neues oder weiterentwickeltes Systemmodell.

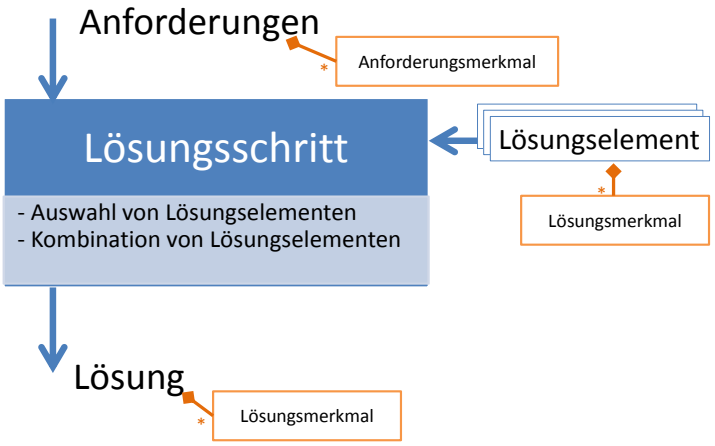


**Abbildung 57 – Entwurfsschritt**

Die Eingangs-Anforderungen können explizit formuliert sein, z.B.:

- Kosten dürfen eine definierte Grenze nicht überschreiten.
- Auf ein spezifisches Ereignis muss innerhalb einer definierten Zeit reagiert werden.

Anforderungen können auch implizit definiert sein, z.B.:

- Damit die funktionalen Anforderungen erfüllt werden können, müssen interagierende Elemente des Systems kompatibel sein (z.B. Modul-SW muss auf Knoten ausgeführt werden können).

Sowohl Anforderungen als auch zur Verfügung stehende Lösungselemente (z.B. aus einer Bibliothek) sind durch Merkmale charakterisiert. D.h. die Merkmale der Anforderungen können genutzt werden, um das am besten passende Lösungselement auszusuchen. Die ausgesuchten Lösungselemente werden dann zu einer Gesamtlösung zusammengesetzt, welche wieder durch Merkmale charakterisiert ist. Die Merkmale der Gesamtlösung entsprechen zum großen Teil den Merkmalen der Lösungselemente, es können durch die Synthese der Lösungselemente auch neue Merkmale entstehen (siehe auch 2.3.5). Für eine ausführlichere Beschreibung siehe [HD13].

Bei einer Feldstudie wurde festgestellt, dass die Verwendung von Merkmalen zur automatischen Verifikation des Entwurfs signifikant zur Verbesserung der Qualität des Entwurfsergebnisses beiträgt. Abbildung 58 stellt den Vergleich für Unterstützung durch FAVA-Notation, zusätzliche Unterstützung durch Merkmale und zusätzliche Unterstützung durch Merkmale und Lösungsmustern dar. Aus der Abbildung ist erkennbar, dass die

Verwendung von Merkmalen zu einer deutlichen Verbesserung des Entwurfsergebnisses führt. Aufgrund der niedrigen Probandenzahl kann man diese Ergebnisse nur qualitativ betrachten [VDF13, Fra14].

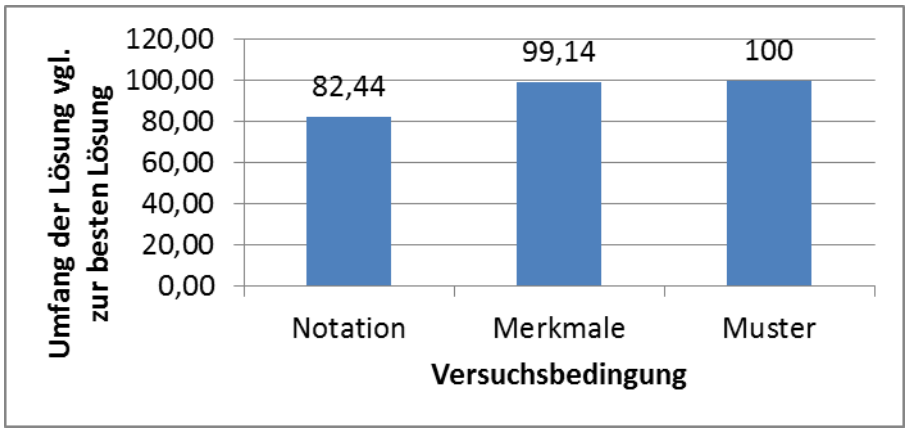


**Abbildung 58 – Vergleich für zunehmende Entwicklungsunterstützung [VDF13]**

Bei der Beurteilung dieses Ergebnisses muss berücksichtigt werden, dass Unterstützung durch Merkmale nur für wenige Zusammenhänge (Speichernutzung durch Funktionsblöcke, Datentypen bei funktionalen Verbindungen, zeitliche Anforderungen für Funktionen) implementiert wurde. Diese Implementierung erfolgte spezifisch nur für die genannten Zusammenhänge und basierte auf dem empirischen Wissen der Projektbeteiligten. Würde man diese Unterstützung basierend auf einer generellen Methodik und für eine breite Palette von Merkmalen bereitstellen, wären wesentliche Effekte zu erwarten.

### 4.2.3 Zusammenfassung

Mit dem Projekt FAVA konnte erfolgreich demonstriert werden, dass Merkmale genutzt werden können, um Engineering-Entscheidungen während der Entwurfsaktivität zu unterstützen und zu verifizieren. Wie in Abbildung 59 dargestellt bezogen sich die Merkmalsdefinitionen jeweils auf einen konkreten Modelltyp.

Der Entwicklungsprozess sieht eine schrittweise Verfeinerung des Systemmodells vor, diese Vorgehensweise erlaubt es eine Verifikation jeweils an den verschiedenen Modellen durchzuführen. Eine weitere wesentliche Neuerung in der FAVA-Methode war die Herstellung von Bezügen zwischen Elementen verschiedener Modelle (in Abbildung 59 durch roten Pfeil dargestellt) und die Verwendung von Merkmalen zur Beschreibung von Anforderungen und zur Verifikation von Entwurfsergebnissen.

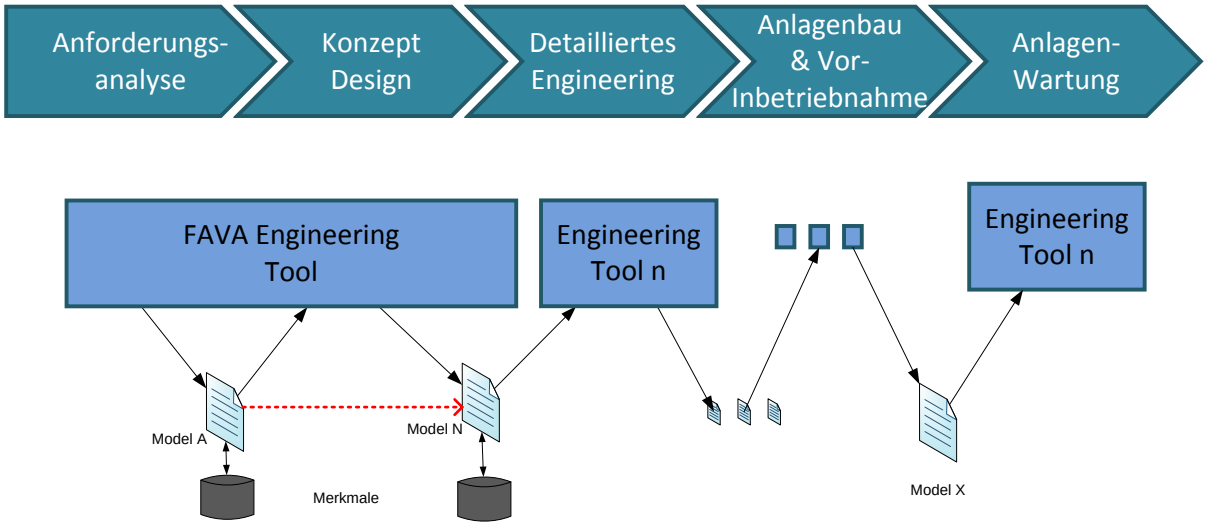


**Abbildung 59 – FAVA Methodik**

FAVA nutzt eine spezielle Notation basierend auf SysML. Dabei stellt sich die Frage, inwieweit eine weitere Notation akzeptiert wird, die nicht von Beteiligten im Entwicklungsprozess getragen wird.

## 4.3 Beziehung zwischen Merkmalmodell und Systemmodellen

Die grundsätzliche Beziehung zwischen Systemmodell und Merkmalmodell wurde bereits in Abbildung 4 dargestellt. Wie in Abbildung 56 zu sehen ist, werden viele verschiedene Modelle im Entwurfsprozess verwendet. Jedes dieser Modelle ist ein Systemmodell, welches mittels Merkmalmodell charakterisiert werden kann (siehe Abbildung 60).

Das Merkmalmodell steht dabei in einem kausalen Zusammenhang mit diesen anderen Modellen. Aufgrund seiner Struktur ist es als allgemeines Datenmodell geeignet, Informationen aus verschiedensten Engineering-Modellen auf eine semantisch eindeutige und maschineninterpretierbare Art abzubilden. Eine formalisierte Einführung des Merkmalmodells als Metamodell für Engineering-Modelle erfolgte in [Mer11].

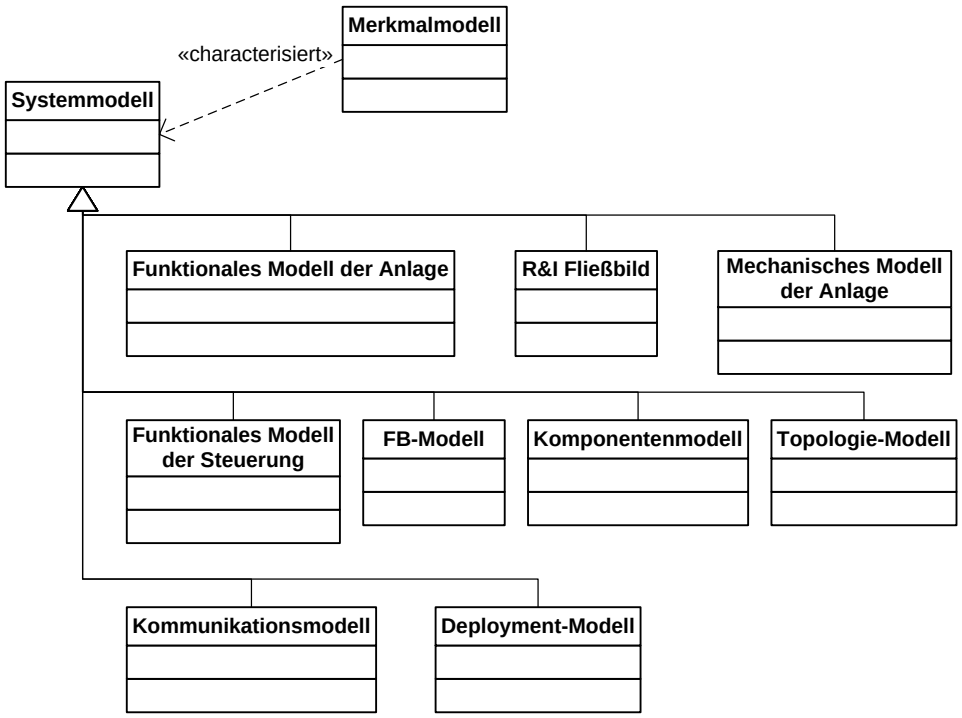


**Abbildung 60 – Beziehung zwischen Systemmodellen und Merkmalmodell**

Die Modellelemente der verschiedenen Engineering-Modelle können jeweils als Merkmalträger repräsentiert werden. Das heißt, die Daten von Engineering-Modellen können mit Merkmalen dargestellt werden. Damit wird es möglich die Daten der Engineering-Modelle wie in 4.2 beschrieben zu bearbeiten und auszuwerten.

In [HD13] wurde vorgeschlagen, das Merkmalmodell so zu erweitern, dass die Beziehung zwischen einem Merkmal und dem entsprechenden Engineering-Modell expliziter dokumentiert werden kann.

## 4.4 Nutzung des Merkmalmodells für die Funktionsbeschreibung

In diesem Abschnitt wird ein Vorschlag zur Nutzung von Merkmalen für die Beschreibung von Anlagenfunktionen eingeführt. Dabei wird die Funktion des Systems als Merkmalträger aufgefasst. Zur vollständigen Beschreibung einer Funktion (siehe auch Definition der Funktion in 2.3.1.3) ist es notwendig, die Eingangsgrößen und Ausgangsgrößen der Funktion zu beschreiben. Deswegen werden auch Material, Energie und Information als Merkmalträger dargestellt.

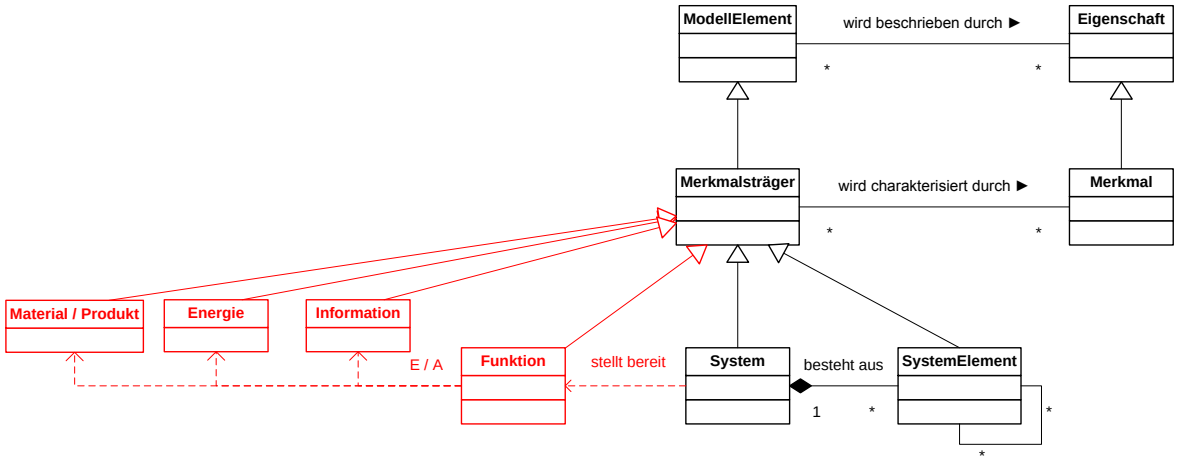


**Abbildung 61 – Funktion als Merkmalträger**

Im Laufe des Anlagenlebenszyklus wird die Beschreibung der Anlagenfunktionen in unterschiedlichen Rollen verwendet. In frühen Phasen des Lebenszyklus (Basic und Detail Engineering) dienen Funktionsbeschreibungen zur Darstellung des Entwurfsziels. In der Betriebsphase eines Systems werden Funktionsbeschreibungen zur Darstellung der Fähigkeiten eines Systems genutzt.

Da die Funktionsbeschreibungen in mehreren Phasen des Anlagenlebenszyklus von verschiedenen beteiligten Gewerken benötigt werden, ist es sinnvoll die Informationen in semantisch eindeutiger Form aufzubereiten.

Merkmale können z.B. zur Darstellung nicht-funktionaler Anforderungen verwendet werden. Beispielsweise können Merkmale zur Beschreibung der Zuverlässigkeit definiert werden. Opgenoorth et al. definieren in [ORG+11] und [WFR+10] folgende Attribute für Anlagenressourcen:

- Mittlere Lebensdauer *MTTF*, Datentyp double,
- Mittlere Reparaturdauer *MTTR*, Datentyp double,
- Verfügbarkeit *A*, Datentyp double,

und folgende Attribute für Prozesse (Funktionen):

- Geforderte Verfügbarkeit $A_{soll}$, Datentyp double
- Geforderte mittlere Lebensdauer $MTTF_{soll}$, Datentyp double

Die Definition des IEC Merkmals „mean operating time between failures“ (mit der ID IEC_CDD#ABB016#002) entspricht der von Opgenoorth et al. verwendeten Definition für MTTF. MTTF kann als inhärentes Merkmal einer Systemkomponente bzw. eines Systems betrachtet werden, das hauptsächlich von der Herstellung des Systems (z.B. verwendetes Material, Konstruktionsprinzip) beeinflusst wird.

Ausgehend von einer allgemeinen Anforderungsbeschreibung, können basierend auf den Anforderungen zu Anzahl von Werkstücken und zur Nutzungszeit die Anforderungen in Bezug auf die maximale Bearbeitungszeit für einzelne Funktionen (maximale Prozessdauer) abgeleitet werden.

Weiterhin können aus den Eigenschaften der Werkstücke die Merkmale der Funktion abgeleitet werden, z.B. können aus Gewicht und Dimensionen des Werkstücks Anforderungen für Transportfunktionen abgeleitet werden, oder aus der Kombination der Materialeigenschaften des Werkstücks mit den Dimensionen der Bearbeitungsfunktion (Größe des Bohrlochs, Dicke des abzutragenden Materials) können Anforderungen in Bezug auf notwendige Kräfte abgeleitet werden.

Ausgehend von einem Merkmalmodell der Funktionen kann das Merkmalmodell für das System abgeleitet werden. Jäger et al. zeigen, dass Attribute der Prozessbeschreibung als Anforderungen an die Systemkomponenten aufgefasst werden können, welche bei der Realisierung des Prozesses eingesetzt werden [JCS+12]. Laut [JCS+12] erfolgt ein Vergleich der Prozessattribute mit den Ressource-Attributen basierend auf Regeln (siehe auch [SFT+11] ).

Entsprechend IEC 61360 werden Funktionen als ‚Itemclass' beschrieben, wobei jeder Klasse von Funktionen ein spezifischer Satz von Merkmalen zugewiesen wird. Es wird vorgeschlagen, die in 3.4.10.2 beschriebenen OLOP als Merkmalbeschreibungen von Funktionen aufzufassen, da in einer OLOP die Anforderungen zur Erfüllung der Funktion beschrieben werden und nicht die Merkmale der Implementierung.

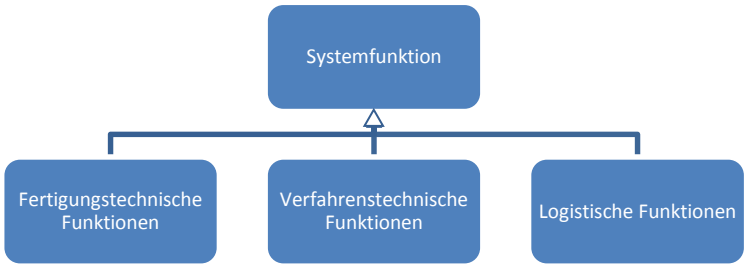


**Abbildung 62 – Klassifizierung von Systemfunktionen**

Die Stammklasse „Systemfunktion" wird durch einen allgemeingültigen Satz von Merkmalen beschrieben. Systemfunktionen der Fertigungstechnik, der Prozesstechnik und der Logistik werden von dieser Stammklasse abgeleitet und können mit zusätzlichen spezifischen Merkmalen beschrieben werden (siehe Abbildung 63).

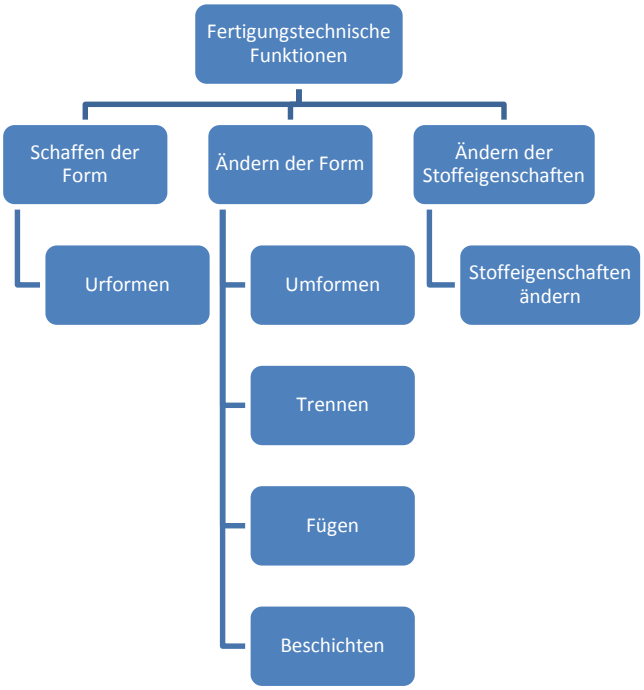


**Abbildung 63 – Klassen fertigungstechnischer Funktionen nach [Heh11]**

Allgemeine Merkmale der Funktionen (Merkmale der Stammklasse „Systemfunktion“) sind:

- Zuverlässigkeit
  (entsprechend der betriebswirtschaftlichen Bedeutung)
- Verfügbarkeit

Entsprechend der Klassifizierung von Funktionen (siehe 2.3.1.3) können weitere Merkmale von Funktionen abgeleitet werden wie:

- Zuordnung (Anlage, AT, ..)
- Eingangsgröße / Ausgangsgröße
  - Typ (Material, Energie, Information)
- Quantität
- Typ (Elementarfunktion, Teilfunktion, Gesamtfunktion)

Will man die Ausführung von Funktionen beschreiben, so muss man die entsprechenden Prozesse beschreiben (siehe 2.3.1.4.1). Merkmale von Prozessen sind:

- Ausführungsdauer
  beschreibt die Zeit für die Ausführung für eine Produkt-Menge (z.B. Zeit pro Stück, Zeit pro Volumeneinheit)
- Durchsatz
  beschreibt die mögliche bearbeitete Produkt-Menge (Output) pro Zeiteinheit

## 4.5 Nutzung des Merkmalmodells für den Strukturentwurf

### 4.5.1 Merkmale von Elementbeziehungen

Es wird eine Erweiterung des Modells vorgeschlagen, um auch Beziehungen zwischen Elementen charakterisieren zu können. Da die Eigenschaften eines Systems sowohl durch die Eigenschaften der Systemelement definiert werden als auch durch die Relation der Systemelemente, ist es notwendig sowohl die Eigenschaften der Elemente zu berücksichtigen als auch die Eigenschaften der Relation.

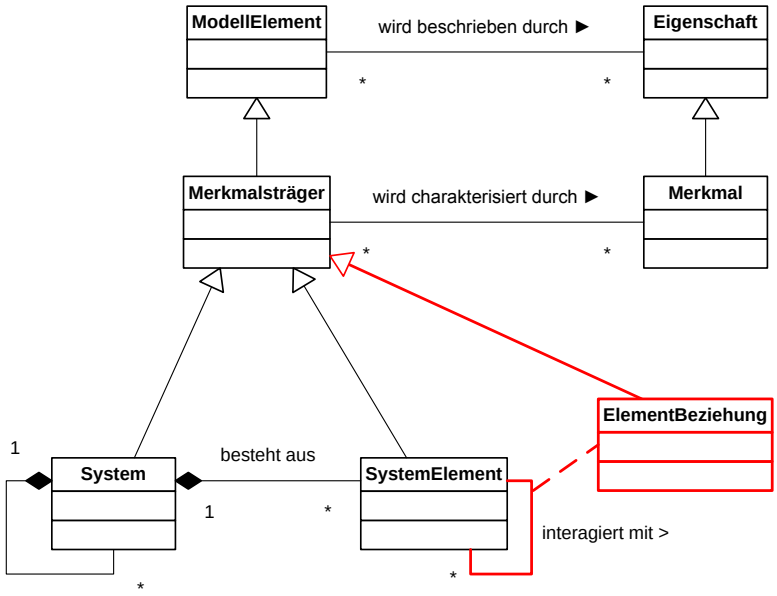


**Abbildung 64 – Einordnung der Elementbeziehung**

Dazu muss die Elementbeziehung als separater Merkmalträger erfasst werden (siehe Abbildung 64). In der Abbildung wird die Beziehung „interagiert mit“ zwischen den SystemElementen durch eine Beziehungsklasse „ElementBeziehung“ beschrieben. Diese Beziehungsklasse wird wiederum als Merkmalträger aufgefasst, dem nun Merkmale zugeordnet werden können.

Ein Beispiel für solch eine Elementbeziehung ist die Beziehung zwischen Funktionsblock(FB) und Hardware auf der der FB ausgeführt wird (Knoten). Um einen FB auf einem Knoten ausführen zu können, müssen diverse Interoperabilitätskriterien erfüllt werden (z.B. benötigter Speicher eines FBs/ bereitgestellter Speicher in einem Knoten). Interoperabilität ist also ein Merkmal für die Beziehung zwischen den Elementen.

Beispiele für Elementbeziehungen:

- Interaktion zwischen Knoten eines Automatisierungssystems
- Interaktion zwischen SW-Komponenten

### 4.5.2 Binnenzusammenhang

Merkmale eines Merkmalträgers können miteinander in Beziehung stehen. Z.B. wird das Gewicht eines Gerätes stark von dem Material des Gehäuses beeinflusst. Dieser Zusammenhang soll durch eine zusätzliche Eigenschaft des Merkmals „Beziehung zu anderen Merkmalen“ dargestellt werden.

Mit dieser Beziehung wird es möglich, zu dokumentieren ob ein Merkmal mit anderen Merkmalen des Merkmalträgers in Beziehung steht. Dabei werden nur allgemeine Abhängigkeiten dargestellt. Arithmetische Beziehungen können mit der Eigenschaft ‘Formel’ dargestellt werden.

Die Eigenschaft „Beziehung zu anderem Merkmal“ kann dabei genutzt werden, wenn die möglichen Auswirkungen einer Entwurfsentscheidung evaluiert werden soll. Beispielsweise kann bei der Wahl des Gehäusematerials das resultierende Gewicht des Gerätes berücksichtigt werden, das evtl. wieder Restriktionen unterliegt (z.B. durch maximales Versandgewicht oder maximales Gewicht am Einbauort). Die Eigenschaft ermöglicht also innere Zusammenhänge darzustellen. Deswegen wird diese Eigenschaft „Binnenzusammenhamg“ (intra relation) genannt. (siehe Abbildung 65).

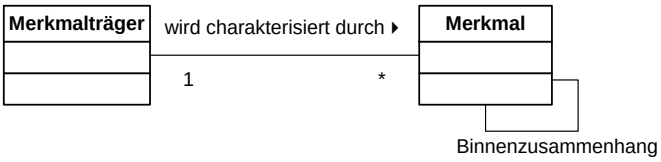


**Abbildung 65 – Binnenzusammenhang**

*Ein ‘Binnenzusammenhang’ beschreibt eine Verhältnis-Beziehung zwischen zwei Merkmalen des gleichen Merkmalträgers.*

### 4.5.3 Außenzusammenhang

Wenn eine Beziehung zwischen Systemelementen hergestellt wird, werden die Merkmale dieser Elemente auch zueinander in Beziehung gesetzt. – Diese Beziehung zwischen Merkmalen verschiedener Merkmalträger wird in dieser Arbeit mit ‚Außenzusammenhang‘ bezeichnet.

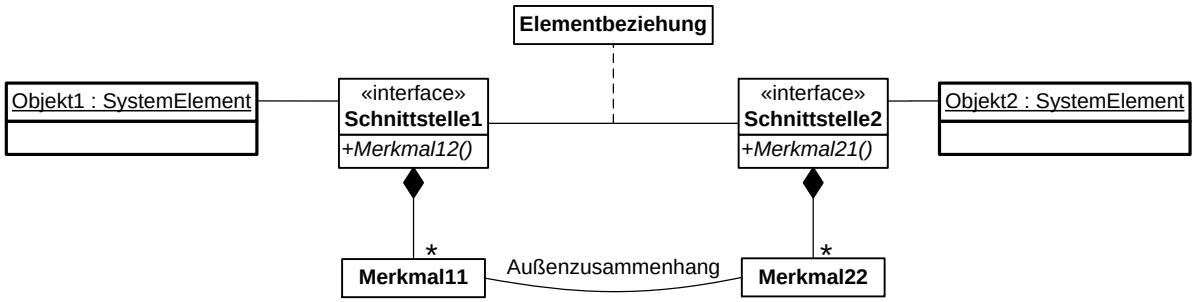


**Abbildung 66 – Außenzusammenhang**

Außenzusammenhänge existieren (im Gegensatz zu Binnenzusammenhängen) nicht statisch, sondern werden erst durch die Beziehung der Systemelemente initiiert. Wird eine Beziehung der Systemelemente hergestellt (über die entsprechende Schnittstelle), dann werden nicht automatisch alle Merkmale der Schnittstellen zueinander in Beziehung gesetzt, sondern es werden je nach Anwendungsfall nur bestimmte Merkmale zueinander in Beziehung gesetzt. (Beispiel Anschluss-Innendurchmesser / Rohr-Außendurchmesser)

*Ein ‚Außenzusammenhang' ist eine Verhältnis-Beziehung zwischen Merkmalen unterschiedlicher Merkmalträger.*

Außenzusammenhänge können nicht statisch erfasst werden, da sie von den Elementbeziehungen abhängen. Das heißt, Außenzusammenhänge werden dynamisch angelegt, wenn Elementbeziehungen hergestellt werden. Die verwendeten Außenzusammenhänge hängen von der Art der Elementbeziehung ab. Dies macht es notwendig Typen von Außenzusammenhängen zu unterscheiden. In Abhängigkeit vom Typ der Elementbeziehung die zwischen zwei Systemelementen hergestellt wird, können verschiedene Außenzusammenhänge angelegt werden. Z.B. wenn eine elektrische Verbindung hergestellt wird, werden die elektrischen Leistungsbereiche der Systemelemente einbezogen; wenn eine Schraubverbindung hergestellt wird, werden geometrische Merkmale der Schnittstellen einbezogen.

### 4.5.4 Regeln für Elementbeziehungen

Nachdem das Konzept „Außenzusammenhang" zur Beschreibung der Beziehung zwischen Merkmalen unterschiedlicher Merkmalträger definiert ist, muss ein entsprechendes Datenmodell gefunden werden. Die in 4.5.3 dargestellte Abhängigkeit eines Außenzusammenhanges von der jeweiligen Elementbeziehung kann modelliert werden, indem ein Außenzusammenhang als Merkmal der jeweiligen Elementbeziehung aufgefasst

wird (siehe 4.5.1). Das Beziehungsmerkmal beschreibt den Außenzusammenhang. Dieser Ansatz ist in Abbildung 67 dargestellt.

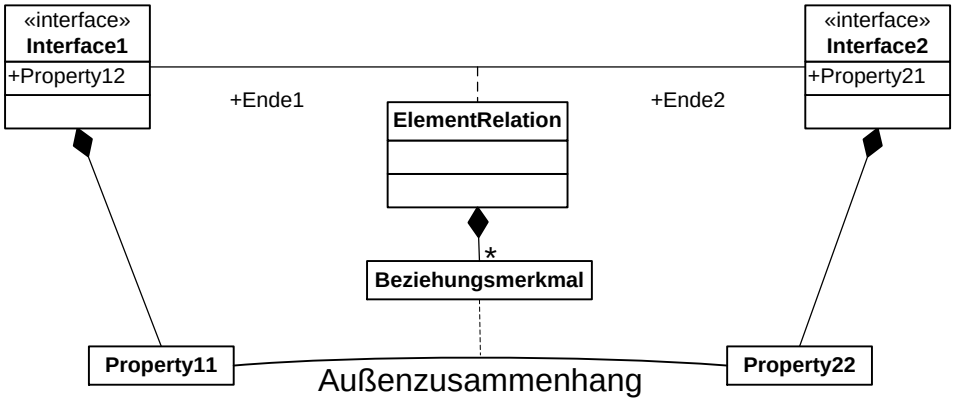


**Abbildung 67 – Beziehungsmerkmal**

UML beschreibt Elementbeziehungen mit gleichartigen Enden (ungerichtete Beziehung) und verschiedenartigen Enden (gerichtete Beziehungen). Jedem Ende der Beziehung ist eine Rolle zugewiesen, die durch das damit verbundene Systemelement ausgefüllt wird [OMG11b] (siehe auch 2.3.2.1). Diese Rollen sind in Abbildung 67 als „Ende1“ und „Ende2“ dargestellt.

Es gibt unterschiedliche Klassen (und Unterklassen) von Elementbeziehungen, z.B.:

- Mechanische Verbindungen
    - Formverbindungen
    - Schraubverbindungen
- Software-Interaktion
    - Funktionsaufruf
    - Methodenzugriff für Objekte

Je nach Typ der Elementbeziehung können unterschiedliche Außenzusammenhänge definiert und jeweils durch ein entsprechendes Beziehungsmerkmal repräsentiert werden. Da ein solches Beziehungsmerkmal die Beziehung zwischen zwei Merkmalen der verbundenen Elemente repräsentiert, wobei es für die Erstellung der Elementbeziehung wichtig ist, dass die Merkmale der verbundenen Merkmale miteinander vereinbar sind. Solch eine Beziehung wird als „Kompatibilität“ bezeichnet [BWW+10, S. 63f.]

Das heißt ein Link (d.h. die Instanz der Elementbeziehung) hat entsprechend des Typs der Elementbeziehung spezifische Merkmale. Beispiele sind:

- Für eine Schraubverbindung
    - Beziehungsmerkmal: Gewindekompatibilität

- Für eine Software-Interaktion
    - Beziehungsmerkmal: Datenkompatibiliät

Die Beschreibung des Außenzusammenhangs kann in einer Regel zusammengefasst werden. In Abhängigkeit von der jeweils einem Systemelement zugewiesenen Rolle werden definierte Merkmalausprägung des Systemelements genutzt um die Ausprägung der Elementbeziehung zu bestimmen. Zum Beispiel für eine Schraubverbindung (siehe Abbildung 68) von metrischen ISO-Gewinde-Elementen gilt[11]:

Gewindekompatibilität = (Außendurchmesser der Rolle ‚Innengewinde' ==
Innendurchmesser der Rolle ‚Außengewinde')

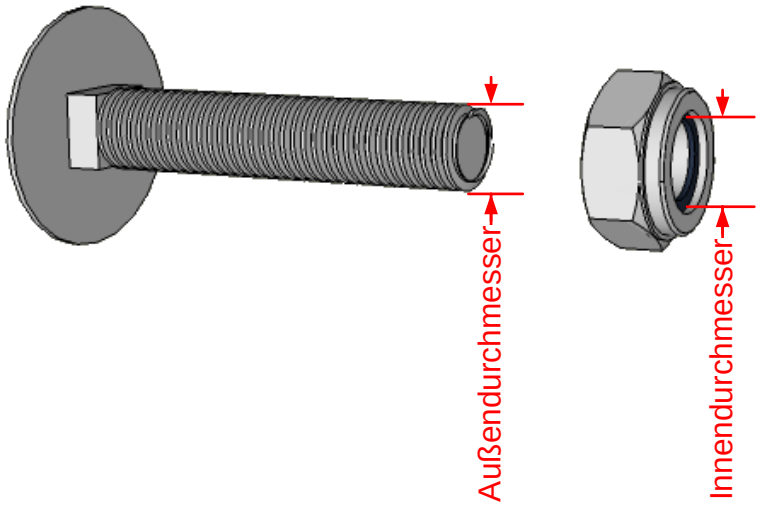


**Abbildung 68 – Beispiel Gewindekompatibilität (Darstellung basierend auf [Jan08])**

Die Regel zur Berechnung der Kompatibilität für den Außenzusammenhang wird in dem Attribut „Formel" des Beziehungsmerkmals definiert. Da die beteiligten Systemelemente bei Definition der Formel nicht bekannt sind, werden Referenzen auf die Rollen der Beziehung verwendet, zum Beispiel für die Formel des Beziehungsmerkmals PropertyR1 in Abbildung 67 werden "Ende1.Property11' und 'Ende2.Property22' verwendet.

[11] Die Darstellung hier ist vereinfacht. Die Kompatibilität von Schraubverbindungen ist komplexer (siehe 6.2.1).

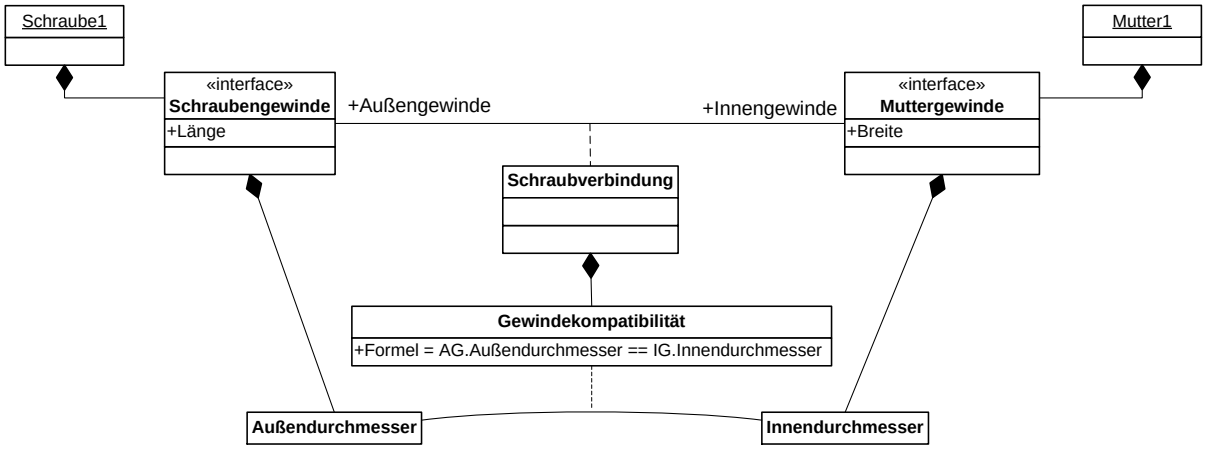


**Abbildung 69 – Beispiel Gewindekompatibiliät als Beziehungsmerkmal**

Basierend auf diesem Ansatz können Bibliotheken von Elementbeziehungen erstellt werden, für die definierte Merkmale und definierte Merkmalbeziehungen (Außenzusammenhänge) zu den Merkmalen der verbundenen Systemelemente (jeweils repräsentiert durch ‚Rolle') beschrieben werden. Die Art der Elementbeziehungen, die in der Bibliothek gespeichert werden müssen, hängt von den im Engineering-Werkzeug verwendeten Systemelementen und den zu erwartenden Anwendungsfällen ab.

Wird in dem Entwurf eines Produktionssystems zwischen zwei Systemelementen eine entsprechende Elementbeziehung hergestellt, so kann mittels der in der Bibliothek hinterlegten Vorlage den jeweiligen Systemelementen die entsprechenden Rollen zugewiesen werden und die jeweilige Äusprägungen der Beziehungsmerkmale bestimmt werden. Beschreibt das Beziehungsmerkmal eine Kompatibilität, so kann das Merkmal genutzt werden, um die Elementbeziehung zu verifizieren.

## 4.6 Zusammenfassung

In diesem Kapitel wurde die Integration von Systemmodellen und Merkmalmodell diskutiert. Die Nutzung der Merkmal-Informationen im Engineering-Prozess wurde dargestellt. Dabei werden Komponenten des Systems bezüglich ihrer Merkmale nicht nur in Hinblick auf das Vorhandensein eines Merkmals betrachtet, sondern Zusammenhänge (Verhältnisbeziehungen) zwischen Merkmalen eines oder unterschiedlicher Merkmalträger können berücksichtigt werden. Eine Methode zur Nutzung dieser Zusammenhänge im Strukturentwurf wurde entwickelt.

# 5 CAEX++

## 5.1 Zielstellung

In diesem Abschnitt wird ein Vorschlag für die Weiterentwicklung von CAEX vorgestellt: CAEX++. Es ist möglich diese Weiterentwicklung für CAEX auch auf AutomationML anzuwenden. Der Nachweis für diese Möglichkeit würde den Rahmen dieser Arbeit überschreiten, deswegen wird hier darauf verzichtet.

Ziel der Weiterentwicklung ist es, im Rahmen von CAEX-Dokumenten eine Entwicklung von Systemen entsprechend dem FAVA-Workflow zu unterstützen und zu dokumentieren. Die Erweiterung von CAEX ermöglicht, dass vorhandene Informationen für die Unterstützung von Engineering-Entscheidungen genutzt werden können und dass eine Dokumentation des Entwurfsprozesses erfolgt. Dabei werden mit „vorhandenen Informationen“ sowohl Modelle aus vorhergehenden Phasen der Entwicklung (z.B. Funktionsmodell als Basis für Strukturmodelle) als auch Informationen aus Merkmalsbibliotheken (für Funktionen und Anlagenkomponenten) gemeint.

CAEX wird als Grundlage dieser Umsetzung gewählt, weil es auf einem Strukturmodell basierend, die Fähigkeit hat, diverse Systemmodelle aus unterschiedlichen Phasen der Entwicklung abzubilden, und weil es ein allgemein anerkannter Standard ist, der in verschiedenen Industrien eingesetzt wird. Da CAEX eine XML-basierte Sprache ist, ist es leicht erweiterbar und erlaubt eine gute Präsentation der Erweiterungsvorschläge. Prinzipiell können diese Vorschläge auch auf andere Strukturbeschreibungssprachen angewendet werden, zum Beispiel auf SysML (siehe [FHE+12a, Fra14]).

## 5.2 Genereller Ansatz

Der Vorschlag CAEX++ bietet gegenüber CAEX folgende Erweiterungen:

- Es ist möglich, verschiedenartige Modell-Dokumente zu referenzieren (nicht nur CAEX-Dokumente bzw. Collada- und PLCopen-Dokumente).
- Verschiedene Modelle für ein System (z.B. aus verschiedenen Phasen der Entwicklung) existieren parallel in einem CAEX-Dokument und die Beziehungen zwischen den Modellen können dargestellt werden (siehe 3.2.4 und 4.2).
- Folgende Modellelemente können mit Merkmalbeschreibungen dargestellt werden:
    - Funktionen (siehe 4.4),
    - Komponenten,

    - Betriebsmittel (siehe 3.4),
    - Schnittstellen (siehe 3.4.10.1).
- Verhältnis-Beziehungen zwischen Modellelementen verschiedener Modelle können dargestellt werden (siehe 3.2.4), z.B. für
    - Funktionen → Komponenten,
    - Komponenten → Betriebsmittel.
- Verhältnis-Beziehungen zwischen Modellelementen eines Modells können dargestellt werden (z.B. Abhängigkeiten) (siehe 2.3.1.2.1 und 3.2.4).
- Diese Verhältnis-Beziehungen können mit weiteren Informationen (z.B. Merkmalen) beschrieben werden (siehe 4.5.1).
- Zusammenhänge zwischen Merkmalen eines Modellelements können beschrieben werden (Binnenzusammenhang) (siehe 4.5.2).
- Die Regeln für funktionale Verbindungen können dokumentiert werden (Außenzusammenhang) (siehe 4.5.3).

**Hinweis zu Konventionen in UML-Klassendiagrammen in diesem Kapitel:** In den UML-Klassendiagrammen werden unveränderte CAEX-Elemente schwarz dargestellt, geänderte CAEX-Elemente werden orange dargestellt, neue CAEX++-Elemente werden rot dargestellt.

## 5.3 CAEXFile

Entsprechend CAEX kann ein CAEXFile verschiedene Elemente (<ExternalReference>, <InstanceHierarchy> etc.) enthalten.

Mit CAEX++ gibt es die Einschränkung, dass alle <InstanceHierarchy>-Elemente, die in einem CAEXFile existieren, Modelle des gleichen technischen Systems bzw. dessen Komponenten darstellen. Dabei ist es erlaubt, dass die Instanzhierarchien unterschiedliche Ausschnitte, Aspekte oder Entwurfsschritte des Systems darstellen.

Die Verwendung von CAEXObject.ID ist obligatorisch, der Wert soll eine UUID sein (im Gegensatz zur Definition bei CAEX v2.15, genauso wie bei AutomationML und CAEX v3.0 definiert; siehe auch Tabelle 7).

## 5.4 <ExternalReference>-Element

Mit CAEX++ wird das existierende <ExternalReference>-Element als URI redefiniert, die sowohl auf verschiedenartige externe Dokumente verweisen kann (z.B. CAEX, Collada, PLCopen) als auch auf Datenbanken (z.B. IEC CDD (siehe 3.4.4)). Dafür werden neue Attribute eingeführt:

- Format (Aussage über Format bzw. Typ des referenzierten Dokuments) und

- UriRef (Referenz auf das Dokument).

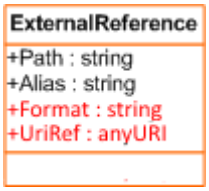


**Abbildung 70 – Erweiterung des <ExternalReference>-Elements**

Mit diesen Änderungen kann auf externe Dokumente und Datenbanken verwiesen werden, die einen eigenen Lebenszyklus unabhängig vom aktuell bearbeiteten Dokument haben. Es erfolgt eine klare Trennung von Referenzen auf Dokumenten und Modellelementen (siehe Referenzen auf PLCopen-XML- und Collada-Dokumente in Tabelle 7).

## 5.5 <InstanceHierarchy>-Element

Ein <InstanceHierarchy>-Element repräsentiert ein in sich abgeschlossenes Modell des betrachteten Systems. Da ein <CAEXFile>-Element unendlich viele <InstanceHierarchy>-Elemente enthalten kann, können sehr viele verschiedene Modelle aus unterschiedlichen Phasen der Entwicklung in einem CAEX-Dokument repräsentiert werden. Deshalb hat ein <InstanceHierarchy>-Element zusätzliche Attribute, um sie in den Fabrik-Lebenszyklus einzuordnen:

- Zeitstempel
- Lebenszyklusphase
- Modelltyp (z.B. funktionales Modell, Komponentenmodell, Equipment, SW, Deployment)
- <ModelRelation>-Element zur Darstellung der Beziehung zu anderen Instanzhierarchien

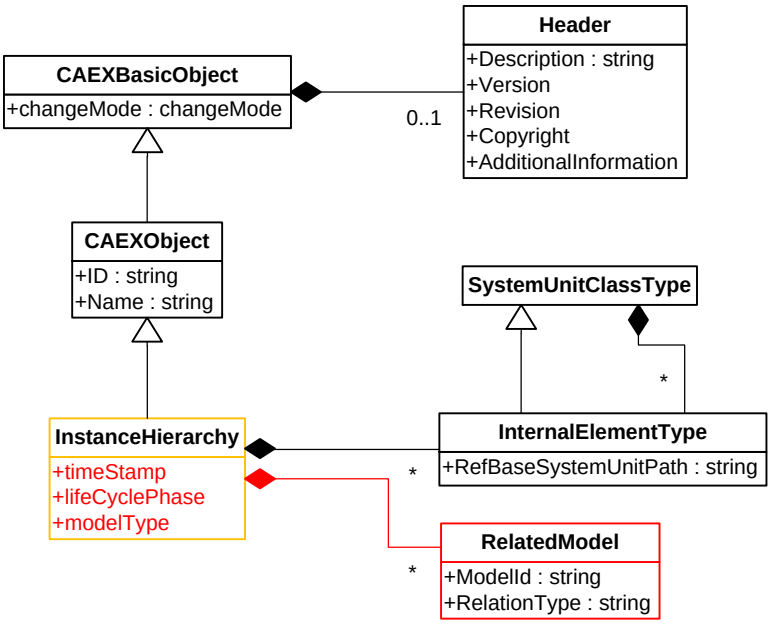


**Abbildung 71 – Erweiterung von <InstanceHierarchy> für Lebenszyklus-Information**

Der existierende Header und der Zeitstempel müssen ausgefüllt werden (d.h. die Elemente werden obligatorisch verwendet, obwohl sie im Schema optional deklariert sind).

Damit ist es möglich zeitliche Beziehungen als auch entwicklungstechnische Beziehungen von mehreren Modellen in einem CAEX-Dokument zu dokumentieren. Es ist möglich den Ablauf der Entwicklung nachzuvollziehen und die Unterschiede in den verschiedenen Modellen (z.B. im Detaillierungsgrad) zu verstehen.

## 5.6 Darstellung funktionaler Modelle mit CAEX++

Mit CAEX++ können reine funktionale Modelle dargestellt werden. Dabei wird der in [SF10], [JCS+12] und [Dra10, S. 195] dargestellte Ansatz genutzt. Ein funktionales Modell wird durch eine Systemhierarchie dargestellt, in der die <InternalElement>-Elemente keine Referenzen auf <SystemUnitClass> verfügen, sondern nur Referenzen auf <RoleClass>-Elemente. Die Attribute der <RoleClass>-Elemente referenzieren entsprechende Merkmalbeschreibungen der Funktionen (beispielsweise die OLOP für die Erfassung von Druck). Der Objektfluss zwischen den Prozesselementen wird durch <InternalLink>-Elemente dargestellt. Das heißt, die Prozesselemente haben entsprechende Interfaces und anhand der Interfaces kann definiert werden, welche Objekte (Energie, Materie, Information) zwischen den Prozesselementen übertragen werden.

```
<InstanceHierarchy ID="36f23100-e812-495e-8c3f-fe8103c992db" Name="Process Model"
        TimeStamp="2014-11-18T13:59:36.7396424+01:00">
    <InternalElement ID="c4c4fa8c-7e24-491b-a1c7-c5312f8eb3fc" Name="Fertigungsprozess Auto" >
        <InternalElement ID="fd4db6f4-3ecf-4bdd-9a5a-010279f090e4" Name="Fahrgestell"
                RefBaseSystemUnitPath="FormalizedProcessDescriptionElementClassLib/Product">
            <Attribute Name="Overall dimensions and weight">
                <Attribute Name="Length" Unit="mm" AttributeDataType="REAL_TYPE">
                    <Value>56</Value>
                    <RefSemantic CorrespondingAttributePath="iec@ABA640"/>
                </Attribute>
            </Attribute>
            <ExternalInterface Name="InOut" RefBaseClassPath="AutomationMLInterfaceClassLib/PPRConnector"/>
        </InternalElement>
        <InternalElement ID="896fc966-36d8-4d50-93e8-62ea792eda06" Name="Fahrgestell bereitstellen">
            <ExternalInterface Name="InOut" RefBaseClassPath="AutomationMLInterfaceClassLib/PPRConnector"/>
            <RoleRequirements RefBaseRoleClassPath="FormalizedProcessDescriptionElementRoleLib/ProcessOperator">
                <Attribute Name="ExecutionTime" Unit="sec" AttributeDataType="REAL_TYPE">
                    <Value>34</Value>
                </Attribute>
            </RoleRequirements>
        </InternalElement>
…
        <InternalLink ID="903be590-e288-486a-b3e6-aa05886c0a05" Name="f1"
            RefPartnerSideA="{fdef15ea-5c7f-4e87-babb-5e9d96ee563a}:InOut"
            RefPartnerSideB="{896fc966-36d8-4d50-93e8-62ea792eda06}:InOut" />
        <InternalLink ID="24b2b76b-1fe9-419f-ba75-5ee4373fd300" Name="f2"
            RefPartnerSideA="{896fc966-36d8-4d50-93e8-62ea792eda06}:InOut"
            RefPartnerSideB="{fd4db6f4-3ecf-4bdd-9a5a-010279f090e4}:InOut" />
…
        </InternalElement>
    </InstanceHierarchy>
```

**Abbildung 72 – Darstellung eines funktionalen Modells**

Verhältnis-Beziehungen (z.B. verwendete Ressourcen) können im nächsten Schritt mit <ElementRelation>-Elementen dargestellt werden (siehe 5.9.1).

Durch die Möglichkeit funktionale Modelle parallel zu den davon abgeleiteten Strukturmodellen darzustellen wird es möglich, die zu erfüllende Funktion eines Strukturelements vollständig zu dokumentieren.

## 5.7 Klassen-Attribute und Instanz-Attribute

Wird ein <InternalElement>-Element angelegt und mittels <@refBaseSystemUnitPath>-Attribut auf ein <SystemUnitClass>-Element bzw. mittels <@refRoleBase> auf ein <RoleClass>-Element verwiesen, so bedeutet dies, dass das Element alle Attribute der Klasse erbt (Klassen-Attribute). Diese Vererbung wird wie beim CAEX 2.15 als Vererbung durch Referenz definiert. D.h. wenn die Klassendefinition geändert wird, bedeutet dies auch, dass die abgeleiteten Instanzen von dieser Änderung betroffen sind. Bei dieser Vererbung können auch Werte der Klassen-Attribute vererbt werden (typischerweise werden in Klassen die Default-Werte für Attribute definiert). Wird im <InternalElement>-Element ein entsprechendes Attribut (Instanz-Attribut) angelegt, so erfolgt dabei eine Wertzuweisung für dieses Attribut. Dabei wird von dem Instanz-Attribut die Identität des Klassen-Attribut übernommen. D.h. das Attribut einer Objekt-Instanz hat den gleichen Identifier wie das

Attribut der Klasse (z.B. <SystemUnitClass> und <RoleClass>), kann aber einen von der Klassendefinition abweichenden Wert annehmen.

Dies ist als Beispiel in Abbildung 73 dargestellt. Das Attribute „Length“ (fett dargestellter Bezeichner) hat einen in der Instanz definierten Wert, das Attribute „Width“ hat einen in der Klasse definierte Wert, die anderen Attribute haben keinen definierten Wert.

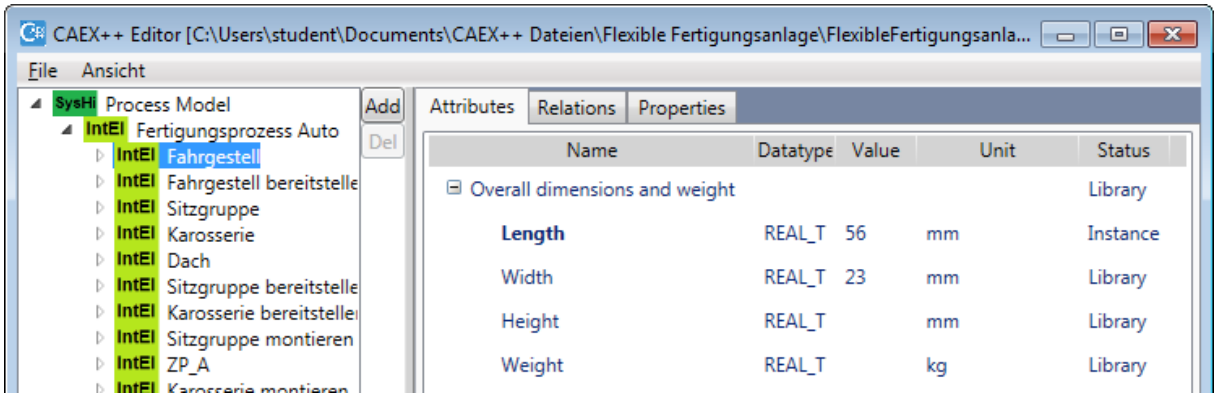


**Abbildung 73 – Klassen- und Instanz-Attribute**

Vorteil dieses Vererbungsmechanismus ist, dass es möglich ist Bibliotheken von ‚Typicals‘ anzulegen, welche die typischen Attribute und Parameter für den Einsatz bestimmter Gerätetypen in einem Unternehmen oder in einer Anlage darstellen. Diese typischen Attribute mit ihren Werten brauchen dann nur einmal in der Bibliothek definiert werden und gelten für alle abgeleiteten Instanzen sofern sie nicht in der Instanz mit abweichenden Werten belegt werden.

## 5.8 Berechnung von <Attribute>-Werten

Um automatische Berechnungen von Attribute-Werten entsprechend 4.5.2 und 4.5.3 durchführen zu können, muss die Berechnungsvorlage für diese Werte hinterlegt werden. Im Projekt FAVA ist dies für spezielle Merkmale durch ein spezifisches Modell erfolgt (siehe 4.2.2), dies genügt aber nicht für eine allgemeine Lösung. Hier wird vorgeschlagen, die Berechnungsvorlage (die Formel) im Attribute-Typ zu hinterlegen.

Deswegen wird für das <AttributeType>-Element ein zusätzliches optionales Attribut „Formel“ eingeführt, das eine Berechnungsvorlage als String enthält (siehe Abbildung 74). Der String enthält einen mathematischen Ausdruck in MathML (content markup) [W3C10][W3C10].

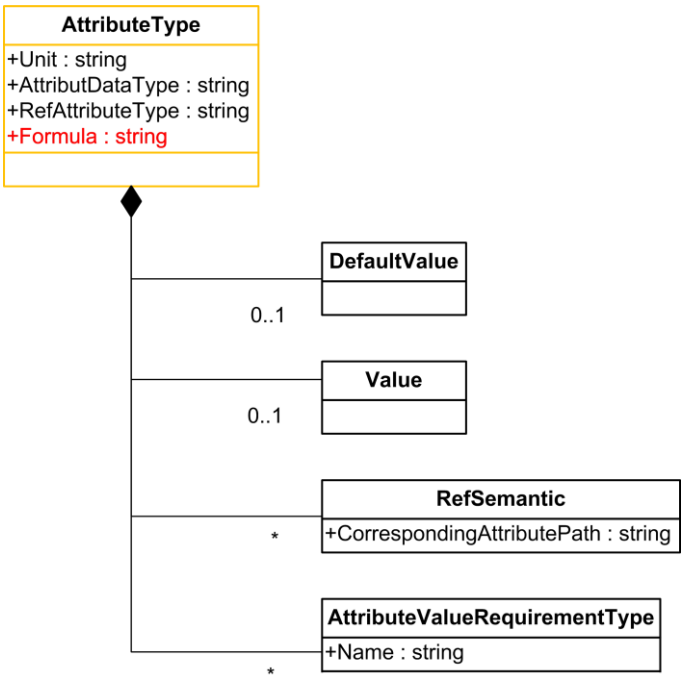


**Abbildung 74 – Erweiterung des <AttributeType>-Elements**

Von dem MathML-Ausdruck können Werte von CAEX-Attributen mittels ID oder Namen des <Attribute>-Elements referenziert werden:

**Tabelle 12 – MathML-Syntax zum Referenzieren von CAEX-Attributen**

| Beschreibung | Syntax | Beispiel |
|---|---|---|
| Binnenzusammenhang | | |
| Referenz auf den Wert eines Attributs am gleichen Merkmalträger | {AttributeID}<br>{AttributeName} | <ci>AttrGUID1</ci><br><ci>Gewicht</ci> |
| Außenzusammenhang | | |
| Referenz auf den Wert eines Attributs an einem anderen Merkmalträger | {Pfad zu Merkmalträger}:{AttributeID}<br>{Pfad zu Merkmalträger}:{AttributeName} | <ci>ElementGUID1:AttrGUID1</ci><br><ci>ElementGUID1:Gewicht</ci> |
| Referenz auf den Attributwert eines Teilnehmers an einer ElementRelation | {sideToken}:{AttributeID}<br>{sideToken}:{AttributeName}<br><br>(sideToken sind „SideA“, „SideB“) | <ci>SideA:AttrGUID1</ci><br><ci>SideA:GewindeArt</ci> |
| Referenz auf den Wert eines Attributs an einem Merkmalträger von einem referenzierten CAEXFile | {alias}@{Pfad zu Merkmalträger}:{AttributeID}<br>{alias}@{Pfad zu Merkmalträger}:{AttributeName} | <ci>Datei01@ElementGUID1:AttrGUID1</ci><br><ci>Datei01@ElementGUID1:GewindeArt</ci> |

Diese Referenzen müssen vom CAEX++-Werkzeug aufgelöst werden, bevor der MathML-Ausdruck validiert werden kann.

Mit diesem Ansatz ist es möglich, Zusammenhänge zwischen Attribut-Werten (siehe 4.5.2 und 4.5.3) so zu dokumentieren, dass Entwurfswerkzeuge diese Zusammenhänge nachvollziehen und darstellen können. Die Folgen von Entwurfsentscheidungen (z.B. Wahl

von Gehäusematerial) können direkt im Entwurfsprozess dargestellt und berücksichtigt werden (z.B. durch stärkere Befestigungen).

## 5.9 Beziehungen zwischen Elementen

Da man mit CAEX++ mittels verschiedener <InstanzHierarchy>-Elementen nun verschiedene Modelle des Systems aus verschiedenen Phasen der Entwicklung und mit unterschiedlicher Granularität darstellen kann (siehe 5.5), genügt es nicht mehr nur Äquivalenz zwischen Modellelementen darzustellen. Wenn beispielsweise in einem funktionalen Modell eine Funktion modelliert wurde, die von mehreren Systemelementen realisiert wird, ist die Darstellung einer Äquivalenzbeziehung entsprechend dem Mirror-Konzept (siehe 3.3.4) falsch, es ist notwendig Teiläquivalenz-, Realisierungs- oder anderen Abhängigkeitsbeziehungen darzustellen. In diesem Abschnitt werden verschiedene Lösungselemente zur Darstellung von Elementbeziehungen zwischen Elementen verschiedener Modelle vorgeschlagen.

### 5.9.1 Äquivalenz zwischen Klassen

Werden in einem CAEX-File mehrere Klassenbibliotheken (<RoleClassLib> und <SystemUnitClassLib>) per <ExternalReference> aus unterschiedlichen Quellen importiert, besteht die Möglichkeit, dass es in verschiedenen Bibliotheken äquivalente Klassen gibt, die jeweils verschiedene Aspekte einer gleichen Klasse darstellen (z.B. Funktionen, Merkmale, elektrische Anschlussbilder, pneumatische Anschlussbilder eines Gerätetyps). Deswegen wird für Klassen (<RoleClass>, <SystemUnitClass>, <InterfacesClass>) die Möglichkeit eingeführt, über <RefSemanticClass> auf äquivalente Klassen mit zusätzlichen Informationen zu verweisen.

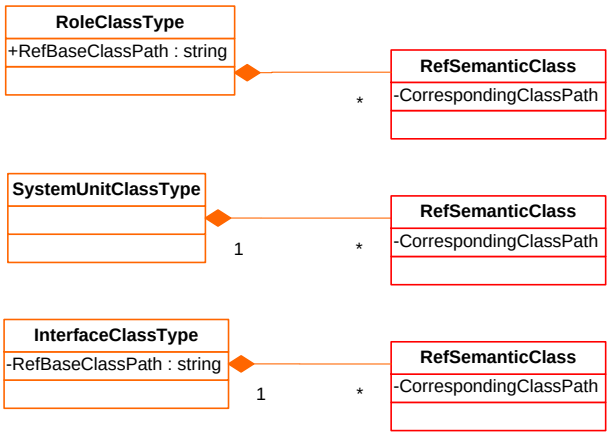


**Abbildung 75 – RefSemanticClass**

Basierend auf dieser Referenz ist es möglich, von einer Bibliothek aus auf Klassen in anderen Bibliotheken zu referenzieren, die weitere Informationen zu den verwendeten Elementen (z.B. Betriebsmitteln) und deren Attribute liefern.

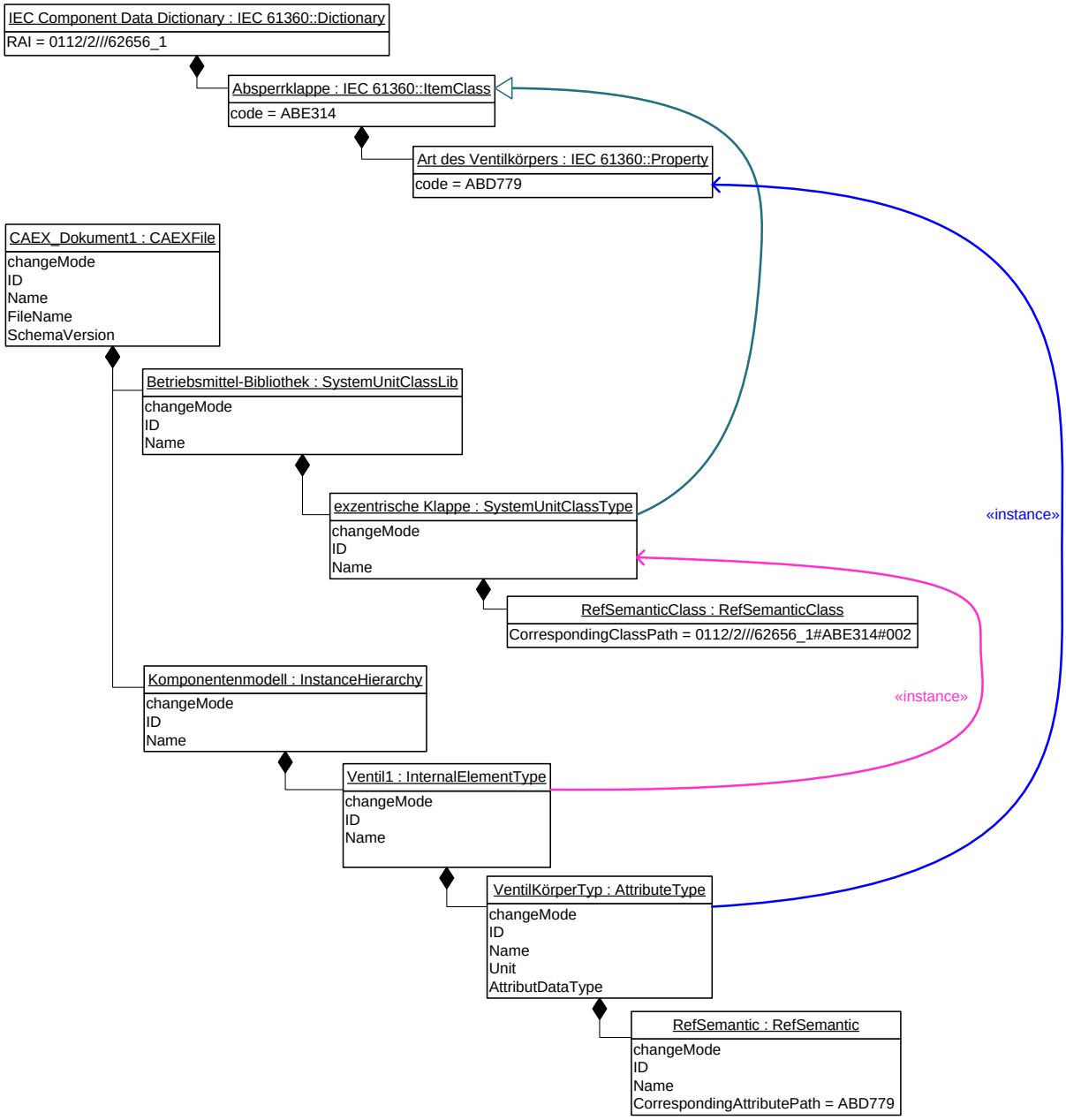


**Abbildung 76 – Äquivalenz zwischen Klassen**

In Abbildung 76 wird als Beispiel dargestellt, wie Property-Definitionen im IEC Common Data Dictionary (CDD) referenziert werden können. Das Systemelement Ventil1 des Komponentenmodells ist durch die Referenz auf die SystemUnit-Klasse in der Klassenbibliothek als exzentrische Sperrklappe spezifiziert (lila Instanziierungsbeziehung). Für diese SystemUnit-Klasse wurde über das Attribut <@CorrespondingClassPath> eine Referenz auf Definition der Sperrklappe im IEC CDD definiert (orange Vererbungsbeziehung).

Im CDD wurde ein Property „Art des Ventilkörpers“ definiert, auf den nun von dem Attribut „VentilkörperTyp“ des Systemelements über ein <SemanticRef>-Element verwiesen werden kann (blaue Instanziierungsbeziehung).

Damit können die in der CDD definierten Geräteklassen und ihre Property eingebunden werden.

### 5.9.2 Beziehungen zwischen Systemelementen

Neben dem Mirror-Konzept wird ein weiteres Element <ElementRelation> eingeführt, welches verschiedene Ausprägungen (z.B. Teiläquivalenz, Generalisierung, Realisierung) haben kann. Die Teilnehmer an der Beziehung werden durch die Attribute <@RefPartnerSideA> und <@RefPartnerSideB> dargestellt. Die Richtung der Beziehung durch das Attribut <@Direction> mit dem Wertevorrat {‚to_A‘; ‚to_B‘; ‚none‘} dargestellt. Damit ist es möglich sowohl gerichtete als auch ungerichtete Beziehungen darzustellen.

Elementbeziehungen werden als eigenständige Strukturelemente in einem <ElementRelationCollection>-Element verwaltet.

Die Art der Ausprägung der Elementbeziehung wird über entsprechende <SupportedRoleClass>-Elemente definiert. Die Attribute des <SupportedRoleClass>-Elements können für die Beschreibung der Elementbeziehung genutzt werden, z.B. kann für eine Teiläquivalenz ein Abdeckungsgrad angegeben werden.

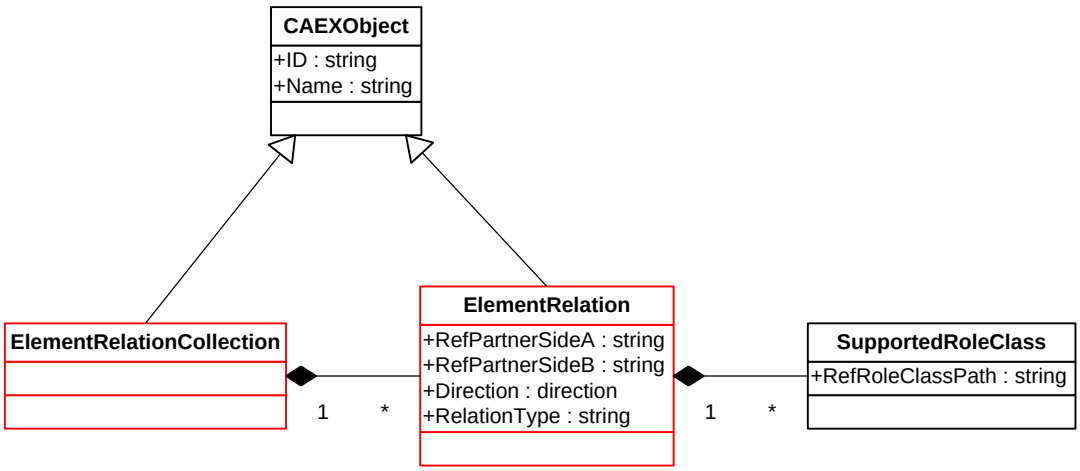


**Abbildung 77 – <ElementRelation>-Element**

Mittels der <ElementRelation>-Elemente können sowohl Beziehungen zwischen Elementen innerhalb einer Systemhierarchie als auch zwischen Elementen unterschiedlicher Systemhierarchien beschrieben werden. Sollen Beziehungen zwischen Elementen in einer Systemhierarchie (d.h. in einem Modell) beschrieben werden, so wird diese Beziehung der

<ElementRelationCollection> der betreffenden Systemhierarchie zugeordnet. Soll eine Beziehung zwischen Elementen unterschiedlicher Modelle dargestellt werden, so wird diese Beziehung der <ElementRelationCollection> der CAEX-Datei zugeordnet (siehe Abbildung 78).

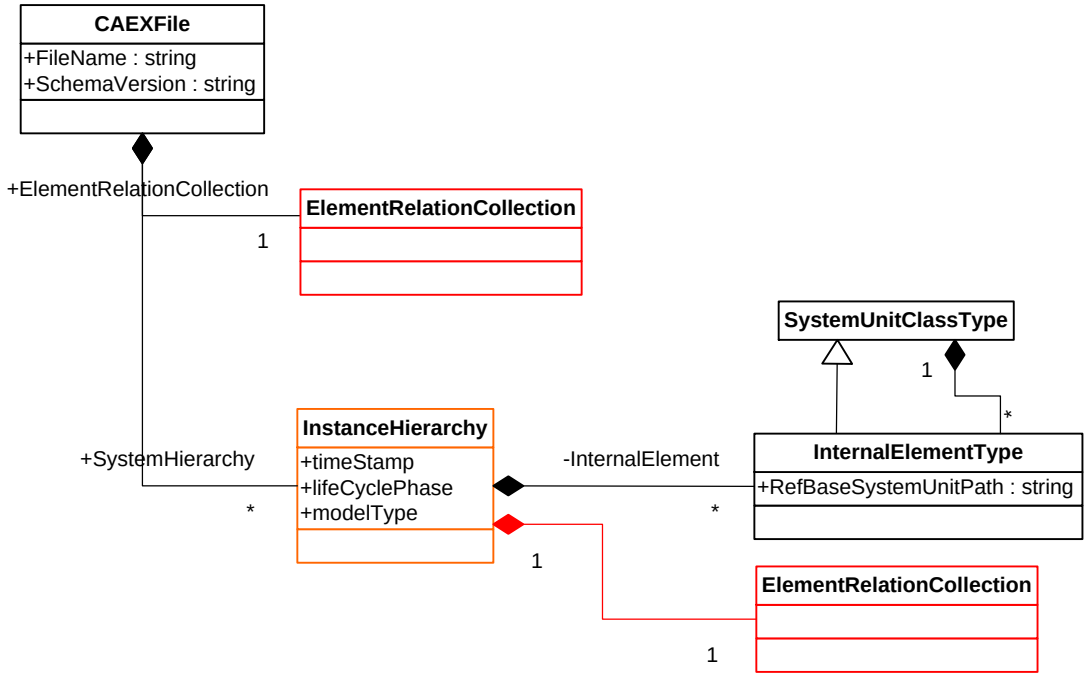


**Abbildung 78 – ElementRelationCollection**

Abbildung 79 zeigt zwei Systemhierarchien, die jeweils das funktionale Modell einer Anlage und das Komponentenmodell einer Anlage darstellen. Die Komponente „Drill-Machine“ realisiert die Funktion „Drill mounting holes“ des Funktionsmodells. Diese Beziehung ist mit einer Elementbeziehung „implements“ dargestellt.

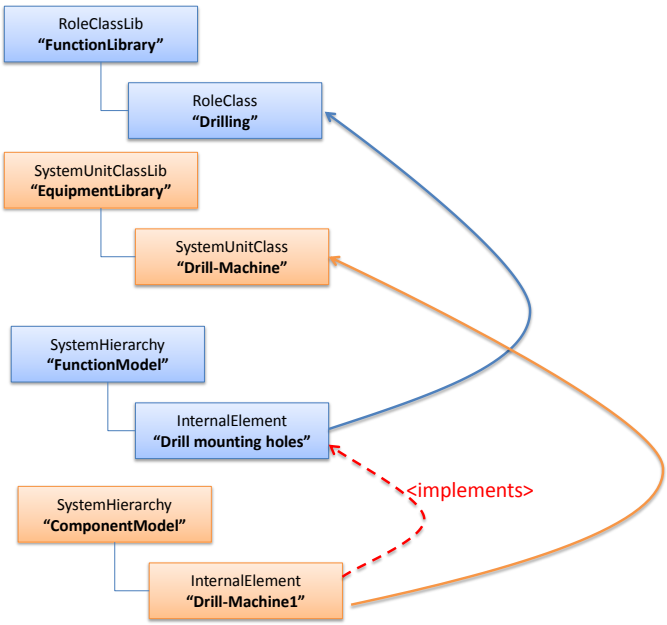


**Abbildung 79 – Elementbeziehung <implements> zwischen Komponente und Funktion**

Durch die „implements"-Referenz wird der Kontext der durch die Komponente bereitzustellenden Funktion expliziter dokumentiert, als wenn direkt von der Komponente auf das <RoleClass>-Element in der RoleClass-Bibliothek verwiesen werden würde.

Mit dem <ElementRelation>-Element können nun unterschiedliche Elementbeziehungen sowohl zwischen Elementen eines Modells als auch Elementen verschiedener Modelle dargestellt werden.

### 5.9.3 Erweiterung des <InternalLink>-Elements

Um den in 4.5.3 beschriebenen Außenzusammenhang unterstützen zu können, ist es notwendig <InternalLink>-Elemente zu erweitern.

Prinzipiell gibt es verschiedenen Möglichkeiten diese Erweiterung umzusetzen: Nutzung von <RoleClass>-Elementen (d.h. Rollen für Elementbeziehungen) oder Nutzung einer neuen Beziehungsklasse (welche in CAEX noch nicht existiert).

<RoleClass>-Elemente könnten für die Beschreibung von Außenzusammenhänge genutzt werden, indem ein <InternalLink>-Element um eine Referenz mittels <SupportedRoleClass>-Element erweitert würde (so wie das <ElementRelation>-Element, siehe 5.9.2). Das referenzierte <RoleClass>-Element würde dann den Typ des Links (der Assoziation) definieren und würde die Eigenschaften der Verbindung mittels Attributen beschreiben. Das Problem bei diesem Ansatz ist, dass <RoleClass>-Elemente nicht für Strukturbeschreibungen

vorgesehen sind, sondern für eine Funktionsbeschreibung. Funktionale Verbindungen müssen jedoch als Strukturelemente betrachtet werden.

Deswegen wird hier der Ansatz gewählt, dass eine neue Klasse <LinkClassType> eingeführt wird, die jeweils einen Link-Typ repräsentiert. Ein <InternalLink>-Element referenziert, wie in CAEX vorgesehen, die beteiligten Interfaces, mit der vorgeschlagenen Erweiterung wird zusätzlich die Link-Klasse mittels dem Attribut „RefBaseClassPath“ referenziert. Außerdem wird das Element <InternalLink> um Attribute erweitert, welche die Link-Instanz beschreiben (siehe Abbildung 80).

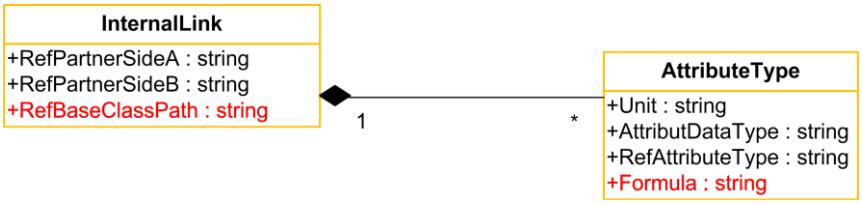


**Abbildung 80 – Erweiterung des <InternalLink>-Elements**

Die referenzierte Link-Klasse (siehe <LinkClassType> in Abbildung 81) beschreibt die Eigenschaften des Verbindungstyps mittels Attributen. Basierend auf der in 5.7 dargestellten Erweiterung für Attribute ist es möglich Algorithmen zu beschreiben, die für eine automatische Validierung der Verbindung genutzt werden können.

Die Link-Klasse beschreibt ebenfalls mit welchen Interface-Typen diese Verbindung hergestellt werden kann. D.h. bei Herstellung einer funktionalen Verbindung, kann anhand der beteiligten Interfaces der Typ der Verbindung identifiziert werden. Die im Typ hinterlegten Informationen (z.B. Regeln) können dann genutzt werden um die Korrektheit der Verbindung zu validieren.

Die verschiedenen vordefinierten Link-Klassen werden in einer neuen Bibliothek <LinkClassLib> verwaltet. Mit dem Element <LinkFamilyType> ist es möglich Hierarchien für Typen von funktionellen Verbindungen darzustellen. Diese Hierarchien können frei für die Organisition der darstellung verwendet werden.

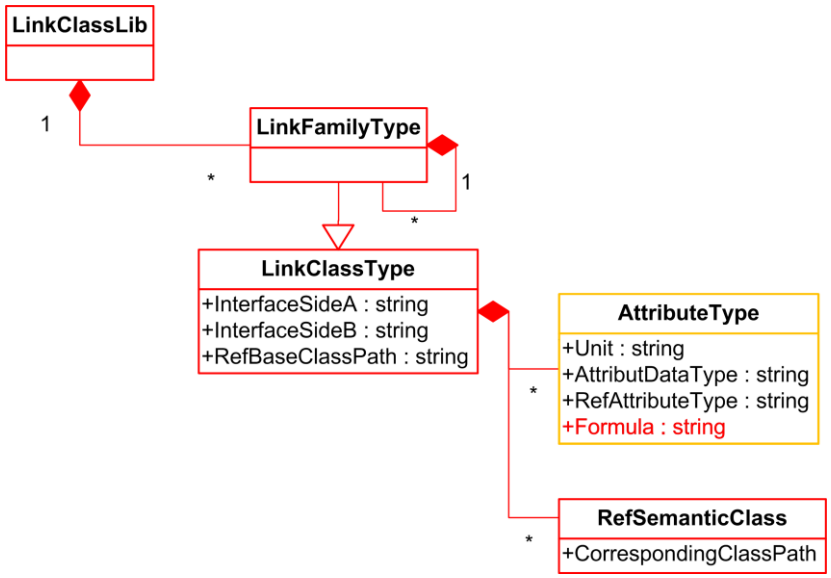


**Abbildung 81 – <LinkClassType>-Bibliothek**

Die Vererbung von einer Link-Klasse zu einer anderen Link-Klasse wird mittels dem Attribut „RefBaseClassPath“ dargestellt. Damit wird es möglich, ein Refinement von Verbindungen darzustellen (siehe 3.5.3).

### 5.9.4 Übersicht

Tabelle 13 zeigt wie die in 2.3.1.2.1und 2.3.3 eingeführten Elementbeziehungen mit CAEX++ dargestellt werden können.

**Tabelle 13 – Unterstützung von Elementbeziehungen in CAEX++**

| Beziehung | Unterstützung | Kommentar |
|---|---|---|
| Funktionale Verbindungen | | |
| Binary Association | <InternalLink> | Typisierung durch LinkClass |
| n-ary Association | <InternalElement> mit spezifischer Role und <InternalLink> | |
| Bestandsbeziehungen | | |
| Aggregation | XML-Hierarchie | Keine Unterscheidung zwischen Shared Aggregation und composite aggregation. |
| Komposition | XML-Hierarchie | |
| Verhältnisbeziehungen | | |
| Realisierung | <SupportedRoleClass> reference und ElementRelation | Referenz nur auf eine Bibliothek und auf eine andere SystemHierarchy |
| Äquivalenz | Mirror-Konzept und RefSemanticClass | Äquivalenz zwischen Instanzen, Äquivalenz zwischen Klassen kann durch RefSemanticClass dargestellt werden |
| TeilÄquivalenz | ElementRelation<br>- Gerichtet<br>- Beschreibung der Abdeckung / Abdeckungsgrad (Attribute) | |

| Beziehung | Unterstützung | Kommentar |
|---|---|---|
| Generalisierung | ElementRelation<br>- Gerichtet | |
| Abhängigkeit | ElementRelation<br>- Gerichtet<br>- Beschreibung der Abhängigkeit | |
| Komplementärbeziehung | ElementRelation | |

## 5.10 Beziehung zwischen CAEX-Systemmodell und Merkmalmodell

CAEX dient zum Austausch von Informationen zwischen verschiedenen Werkzeugen, die für den Entwurf von automatisierten Systemen genutzt werden. Der aktuelle Ansatz von CAEX beruht darauf, dass die benötigte Information inklusive der Definition von Equipment-Klassen in dem CAEX-Dokument enthalten ist (siehe Abbildung 52). Dieser Ansatz ignoriert die vorhandenen Ansätze zum Austausch von Informationen basierend auf Merkmalbeschreibungen (siehe 3.4.10 und 3.4.11). In diesem Abschnitt wird die Integration von Merkmalbeschreibungen in CAEX dargestellt. Durch diesen Ansatz ist es möglich, dass die beteiligten Werkzeuge auf einen einheitliche und möglichst vollständige Definition von Equipment zurückgreifen, während der Informationsaustausch zwischen den Werkzeugen (basierend auf dem Key-Value-Ansatz entsprechend 3.4.10.2) sehr effizient gestaltet werden kann.

### 5.10.1 Beziehung zwischen <SystemUnitClass> und IEC 61360-ItemClass

Eine DLOP-Definition entsprechend IEC 61987-10 (siehe 3.4.10) entspricht der Definition einer <SystemUnitFamilyType> in einer <SystemUnitClassLib>. Ein Block of Properties entspricht der Definition eines Moduls bzw. der Definition einer <InterfaceFamilyType> in einer <InterfaceClassLib> (siehe Abbildung 82).

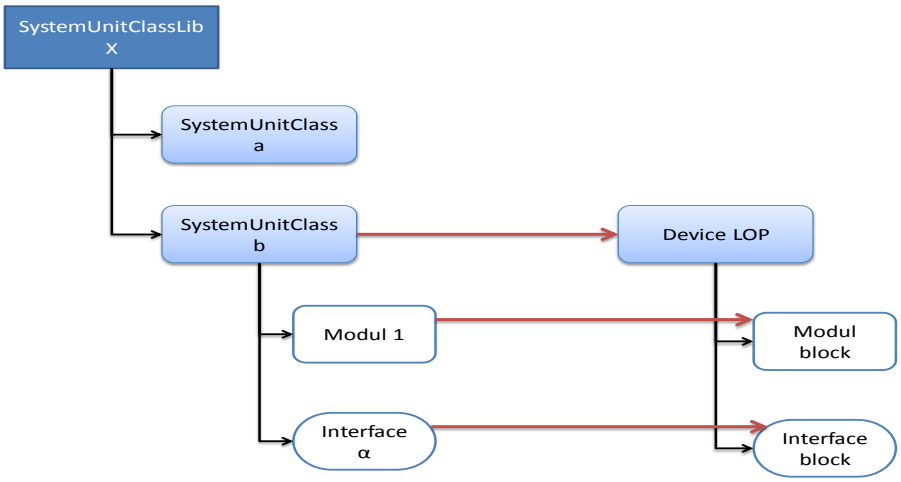


**Abbildung 82 – Verhältnis zwischen Merkmalmodell und CAEX-Modell**

Allerdings stellt die ItemClass-Definition (also auch LOP und Block of properties) wesentlich mehr Informationen bereit als eine <SystemUnitClass>-Definition (siehe Tabelle 14).

**Tabelle 14 – Vergleich von CAEX-SystemUnitClass und IEC 61360-ItemClass**

| SystemUnitClass – Eigenschaften | IEC 61360 ItemClass-Eigenschaften[12] |
|---|---|
| ID | |
| Name | |
| Header | |
| RefSemanticClass | Code (Identifier) |
| | Versionsnummer |
| | Revisionsnummer |
| | Bevorzugte Benennung(en) |
| | Coded name |
| | Synonymous name |
| | Definition |
| | Note |
| | Remark |
| | Drawing reference |
| | Source document of class definition |
| | Classifying data element type |
| | Applicable data element type |
| | Superclass |
| | Subclass |

Um diesen großen Vorrat an Informationen nutzen zu können, wird vorgeschlagen, dass mittels <RefSemanticClass>-Element von einer Klassendefinition in einer System-Elementen-Bibliothek (<SystemUnitLib>) oder in einer Interface-Bibliothek (<InterfaceLib>) auf eine entsprechende ItemClass-Definition in einem Dictionary (CDD oder OTD) verwiesen wird.

Da die ID einer Klasse innerhalb eines Dictionary eindeutig durch Code und Versionsnummer definiert wird (siehe 3.4.2), entspricht die <RefSemanticClass> dieser Kombination (siehe Tabelle 14, 4. Zeile). Die entsprechende Reference muss sich entsprechend aus der Id der Bibliothek und dem Klassen-Identifikatoren zusammensetzen. Vorgeschlagen wird hier, mittels <ExternalDocumentRef> ein Alias für das Dictionary zu definieren und die Reference auf die Klasse entsprechend aus diesem Alias und dem Identifikatoren für die Klasse zusammenzusetzen (siehe Abbildung 83).

```
    <ExternalReference Path="C:/Users/student/Documents/CAEX++ Dateien/Schraube/eClassOntology.owl"
        Alias="eClass" Format="owl">
        <Description>eClass Dictionary</Description>
    </ExternalReference>
…
        <SystemUnitClass ID="2cc37057-93f7-4668-8251-913cd50f3c4f" Name="Mutter">
…
            <EquivalentClass CorrespondingClassPath="eClass@0173-1#02-AAB013#010"/>
        </SystemUnitClass>
```

**Abbildung 83 – Referenz auf eine Klassendefinition in einem eCl@ss-Dictionary**

Tabelle 15 stellt die entsprechende Zuordnung von IEC 61987-Elementen und CAEX-Elementen dar.

[12] Eigenschaften zur Dokumentenverwaltung und Verwaltung der Übersetzung werden nicht gelistet.

**Tabelle 15 – Zuordnung von IEC 61987-Element und CAEX-Element**

| IEC 61987-Element | CAEX-Element |
|---|---|
| OLOP | RoleClassType |
| DLOP | SystemUnitClassType |
| PropertyBlock (ModuleDefinition) | SystemUnitClassType |
| PropertyBlock (InterfaceDefinition) | InterfaceClassType |
| PropertyBlock (Gruppierung von Properties zu einem Thema) | AttributeType mit untergeordneten Attributen |
| Property | Attribute |

### 5.10.2 Beziehung zwischen CAEX-Attribut und IEC 61360-Property

Ein <Attribute>-Element kann die oben für <RefSemanticClass> beschriebenen Methode nutzen um mittels <RefSemantic> auf eine Property-Definition verweisen. Die Referenz wird aufgebaut als „alias@PropertyID“. ‚Alias‘ entspricht dabei einem <ExternalReference>-Element, welches auf ein Dictionary verweist. ‚PropertyID‘ entspricht dabei der ID eines Properties in diesem Dictionary. Diese Art von Referenz funktioniert sowohl für Dictionary basierend auf IEC 61360 als auch für Ontologien nach ISO 22745.

Tabelle 16 stellt einen Vergleich zwischen <Attribute>-Element und einer Property-Definition nach IEC 61360 dar. Es ist deutlich zu erkennen, dass ein Property einen deutlich größeren Informationsvorrat beinhaltet.

**Tabelle 16 – Vergleich von CAEX-Attribut und IEC 61360-Property**

| CAEX-AttributType – Eigenschaften | IEC 61360 Property-Eigenschaften[13] |
|---|---|
| ID | |
| Name | |
| Header | |
| RefSemantic | Code (Identifier) |
| | Versionsnummer |
| | Revisionsnummer |
| | Bevorzugte Benennung(en) |
| | Sprache |
| | Text |
| | Synonyme Bezeichner |
| | Kurzname (Abkürzung ) |
| | Buchstaben-Symbol (Formelzeichen) |
| | Synonymes Buchstaben-Symbol |
| | Definition |
| | Sprache |
| | Text |
| | Note (zur Definition) |
| | Bemerkung (zur Definition) |
| Formel | Formel |
| | Abbildung |
| | Quellendokument für die Datenelement-Typ-Definition |

[13] Eigenschaften zur Dokumentenverwaltung und Verwaltung der Übersetzung werden nicht gelistet.

| CAEX-AttributType – Eigenschaften | IEC 61360 Property-Eigenschaften[13] |
|---|---|
| AttributDataType | Datentyp |
| | Werteformat |
| Unit | Maßeinheit |
| AttributeValueRequirementType | Werteliste (Wertebereich / Wertevorrat ) |
| DefaultValue | |
| Value | Wert |
| | Werte-Kodierung |
| | Wert-Bedeutung |
| | Quellendokument für die Werteliste |
| | Referenzierte Klassen-Identifier |
| | Abhängigkeits-Datenelementtyp |
| | Datenelement-Typ-Klasse |

## 5.11 Zusammenfassung

In diesem Kapitel wurden die Weiterentwicklung von CAEX zu CAEX++ vorgestellt.

Die wesentlichen Änderungen betreffen die Dokumentation von Entwurfsentscheidungen, z.B. dadurch, dass Beziehungen zwischen Modellen eines Systems und die Beziehungen zwischen den Elementen verschiedener Modelle (z.B. die Ableitung von einem funktionalen Modell zum Strukturmodell) dokumentiert werden können. Durch Definition von Referenzen auf externe Dokumente und von Referenzen zwischen Klassen wird die Wiederverwendung existierender Beschreibungen und Bibliotheken (z.B. IEC CDD und eCl@ss) gefördert. Die Unterstützung von Property-Beziehungen (Binnenzusammenhänge und Außenzusammenhänge) wurde ebenfalls definiert, so dass es möglich ist, die Beziehung von Properties zu beschreiben und auszuwerten.

Insgesamt ermöglicht der beschriebene Ansatz einen Entwicklungsprozess, der auf einer schrittweisen Verfeinerung der Systemmodelle beruht. Zusammenhänge zwischen den Systemmodellen bzw. zwischen Elementen der verschiedenen Modelle können durch Referenzen dokumentiert werden (in Abbildung 84 durch rote Pfeile dargestellt).

Engineering-Daten können durchgängig von frühen Phasen des Engineering bis hin zum Betrieb der Anlage genutzt werden, da sie in einer gemeinsamen Datenbasis (dem CAEXfile) mit einer gemeinsamen semantischen Basis (den Standardmerkmalen) basierend auf einer gemeinsamen semantischen Definition gepflegt werden (in Abbildung 84 durch grüne Pfeile dargestellt). Diese gemeinsame semantische Basis wird für alle Modelle durch Referenzen auf entsprechende Merkmalsdefinitionen bereitgestellt.

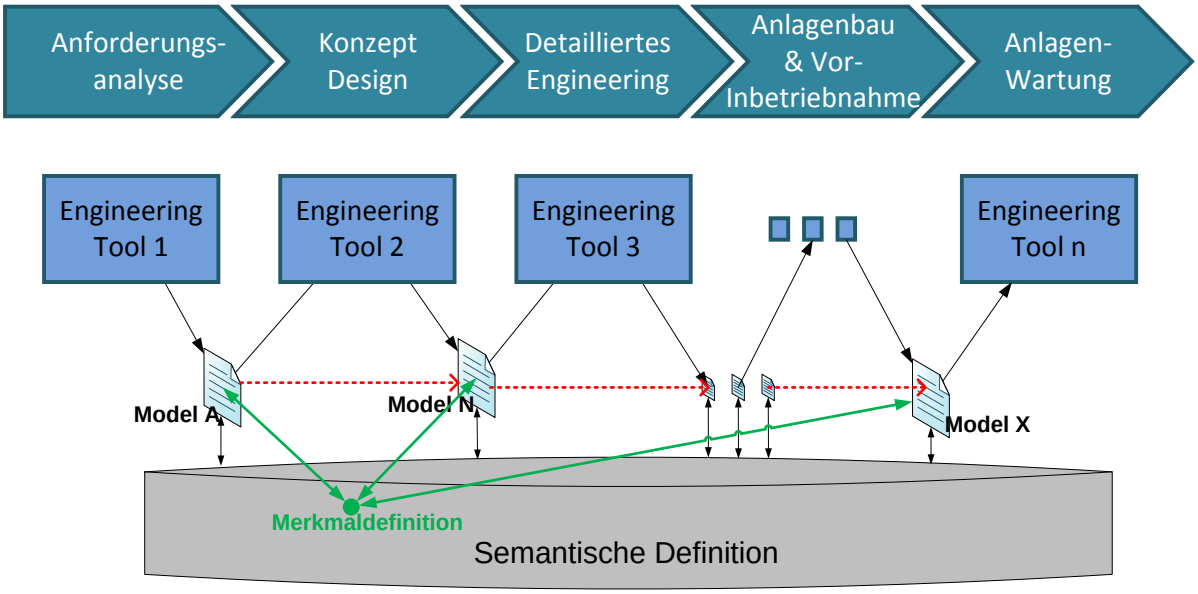


**Abbildung 84 – CAEX++ Methodik**

CAEX++ ist rückwärts-kompatibel zu CAEX. Da es Daten als obligatorisch definiert, die in CAEX nur optional definiert werden, kann nicht aus jedem CAEX-Dokument ein CAEX++-Dokument generiert werden. Es ist aber möglich ein CAEX++-Dokument in ein CAEX-Dokument zu konvertieren. Dabei erfolgt jedoch ein Informationsverlust. Die nur in CAEX++ definierten Informationen gehen bei der Konvertierung verloren.

# 6 Anwendung beim Entwurf von Systemen

## 6.1 CAEX++ Editor

Um die in Kapitel 5 beschriebenen Erweiterungen von CAEX und die Verwendung von CAEX++ im System-Engineering zu testen wurde ein Editor basierend auf CAEX++ entwickelt.

Mit diesem Editor ist es möglich verschiedene Dictionaries (z.B. basierend auf IEC CDD und eCl@ss) darzustellen, den Bezug zu Bibliotheken herzustellen, verschiedene Systemmodelle zu erstellen, aus den Systemmodellen Diagramme für formalisierte Prozessbeschreibungen und Komponentendiagramme zu erzeugen und Validierungen für funktionale Verbindungen (d.h. für Außenzusammenhänge) durchzuführen.

Der Editor wurde genutzt, um die im Folgenden dargestellten Beispiele zu erstellen und die in dieser Arbeit beschriebenen Vorgehensweisen zu validieren.

## 6.2 Einfache Verbindungen

In diesem Abschnitt werden zwei einfache Beispiele für Außenzusammenhänge dargestellt:

- Schraubverbindung und
- Signalverbindung.

Es werden die relevanten Merkmale der Elemente dargestellt und die Merkmale der Elementbeziehung. Diese sehr einfachen Beispiele sollen das Vorgehen nach CAEX++-Methodik detailliert illustrieren.

### 6.2.1 Schraubverbindung

In diesem Abschnitt wird eine Schraubverbindung als einfaches Beispiel einer funktionalen Verbindung diskutiert. Diese Darstellung ist sehr detailliert und dient nur zu Illustration des Prinzips. In realen Anwendungen wird man die Schraubverbindungen eines technischen Systems nur in Ausnahmefällen so detailliert darstellen.

Das System Schraube-Mutter hat die Funktion Kräfte auf die Umgebung auszuüben (d.h. Eingangsgröße: Drehwinkel der Mutter zur Schraube, Ausgangsgrößen: jeweils eine Kraft an Mutter und Schraube). Die Kräfte werden typischerweise über die Auflageflächen der Schraube bzw. der Mutter ausgeübt, wobei es von der Art der Anwendung abhängt ob Zug- oder Druckkräfte ausgeübt werden. Diese Kräfte können durch die lokale Anordnung von Schraube und Mutter zueinander beeinflusst werden. – Schraube und Mutter haben damit

also jeweils 2 Schnittstellen: die Auflageflächen (jeweils als Schnittstelle zur Umgebung) und das Gewinde (jeweils als Schnittstelle zum anderen Systemelement).

Soll eine Schraubverbindung zwischen zwei Elementen hergestellt werden, so müssen diverse Eigenschaften der Schnittstellen beider Elemente einander entsprechen. Wesentliche Voraussetzung zur Herstellung einer Schraubverbindung ist, dass ein Innengewinde (Muttergewinde) und ein Außengewinde (Bolzengewinde) aufeinander treffen, welche in wesentlichen Bestimmungsgrößen (Merkmalen) zusammenpassen. Wesentliche Merkmale eines Gewindes sind (siehe auch Abbildung 85):

- Außendurchmesser (Nenndurchmesser) d bzw. D,
- Flankendurchmesser $d_2$ bzw. $D_2$
- Kerndurchmesser $d_3$ bzw. $D_1$,
- Gewindesteigung P,
- Flankenwinkel α,
- Teilflankenwinkel $\alpha_1$ und $\alpha_2$,
- Radius am Gewindegrund (Rundung) R,
- Gewindetiefe $h_3$,
- Flankenüberdeckung (Gewindetragtiefe) $H_1$ [WKT07].

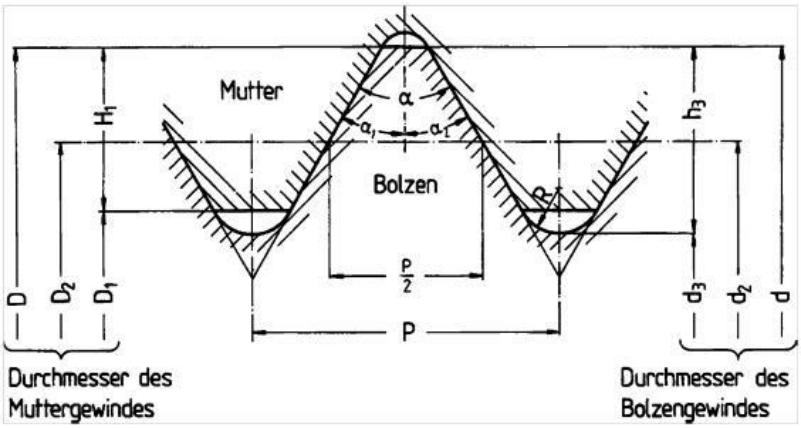


**Abbildung 85 – Bestimmungsgrößen eines Gewindes nach DIN 13 Teil 19 [WKT07, S. 6f.]**

Das heißt vor dem Herstellen einer Schraubverbindung müssen die Merkmale der Mutter den Merkmalen des Bolzens gegenübergestellt werden (siehe Tabelle 17).

**Tabelle 17 – Merkmale von Elementen einer Schraubverbindung**

| Merkmalbezeichnung | Bolzen | Mutter |
|---|---|---|
| Außendurchmesser (Nenndurchmesser) | d | D |
| Flankendurchmesser | $d_2$ | $D_2$ |
| Kerndurchmesser | $d_3$ | $D_1$ |
| Gewindesteigung | P | P |
| Flankenwinkel | α | α |
| Teilflankenwinkel | $\alpha_1$, $\alpha_2$ | $\alpha_1$, $\alpha_2$ |
| Radius am Gewindegrund (Rundung) | R | |

| | | |
|---|---|---|
| Gewindetiefe | $h_3$ | |
| Flankenüberdeckung (Gewindetragtiefe) | $H_1$ | $H_1$ |

Die Flankenüberdeckung ist nicht eine Eigenschaft eines der verbundenen Elemente, sondern eine Eigenschaft, welche durch die Paarung der Elemente entsteht. Man kann die Flankenüberdeckung also als Eigenschaft der Verbindung betrachten ($H_1 = D – D_1$).

Würden Mutter und Bolzen mit gleichen Maßen gefertigt werden, wäre es nicht möglich die Verbindung herzustellen, weil das Muttergewinde nicht genug Raum hat auf dem Bolzengewinde zu gleiten. Es muss ein Spiel hergestellt werden, indem ein Grundabmaß *„es“* bei den Maßen des Bolzengewindes beachtet wird (siehe Abbildung 86).

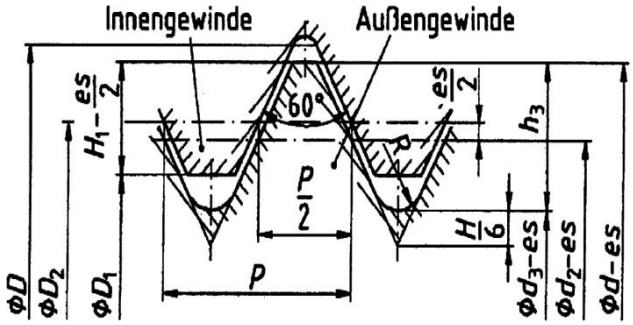


**Abbildung 86 – Bestimmungsgrößen eines Gewindes nach DIN 13 Teil 19 [WKT07, S. 10f.]**

D.h. für das Herstellen der Schraubverbindung müssen folgende Regeln beachtet werden:

**Tabelle 18 – Regeln für Schraubverbindung**

| Geprüfte Merkmale der verbundenen Systemelemente | Bolzen | Regel | Mutter |
|---|---|---|---|
| Nenndurchmesser | d | D-d = es | D |
| Flankendurchmesser | $d_2$ | $D_2 – d_2$ = es | $D_2$ |
| Kerndurchmesser | $d_3$ | $D_1 - d_3$ = es | $D_1$ |
| Gewindesteigung | P | P = P | P |
| Flankenwinkel | α | α = α | α |
| Teilflankenwinkel | $\alpha_1$ | $\alpha_1 = \alpha_1$ | $\alpha_1$ |
| | $\alpha_2$ | $\alpha_2 = \alpha_2$ | $\alpha_2$ |
| Radius am Gewindegrund (Rundung) | R | | |
| Gewindetiefe | $h_3$ | | |
| Gewinderichtung | | | |

Die Flankenüberdeckung wird durch das Grundabmaß reduziert: $H_1 = D – D_1$– es/2.

Für Schrauben und Muttern sind diese Merkmale in entsprechenden Normen standardisiert und zusammengefasst. Industrielle Merkmalbeschreibungen verweisen auf diese Normen und nutzen nicht alle oben beschriebenen Eigenschaften der Schraubverbindung. Zum Beispiel definiert die DIN ISO 724 für Nenndurchmesser im Bereich von 1 mm bis 300 mm

jeweils die erlaubten Werte für die Gewindesteigung und die zugeordneten Werte für Flankendurchmesser und Kerndurchmesser [DIN10, S. 6].

Siehe Anhang A für Beispiele für Merkmalbeschreibungen von Schrauben und Muttern (siehe A.1 und A.2). Basierend auf diesen Merkmalbeschreibungen können SystemUnit-Klassen in CAEX++ definiert werden (siehe A.3), wobei die entsprechenden Attribute der Darstellung der relevanten Schnittstelle zugeordnet wurden und zur vollständigen Beschreibung der Kompatibilität die Beschreibung des Interfaces „Aussengewinde“ um ein Attribut „GewindeArt“ (siehe A.3.1).

**Tabelle 19 – Regeln für Schraubverbindung**

| Bezeichung | Geprüfte Merkmale der verbundenen Systemelemente | Bolzen | Regel | Mutter |
|---|---|---|---|---|
| Durchmesser-Kompatibilität | Nenndurchmesser | d | D-d = es | D |
| RichtungsCompatibility | Gewinderichtung | $GR_b$ | $GR_b$ == $GR_m$ | $GR_m$ |
| LaengenCompatibility | Gewindelänge | $GL_b$ | $GL_b$ >= $GR_m$ | $GL_m$ |
| GewindeArtCompatibility | GewindeArt | $GA_b$ | $GA_b$ == $GA_m$ | $GA_m$ |

Für die entsprechend Schraubverbindung kann nun eine Verbindungsklasse (siehe 5.9.2) mit entsprechenden Attribute-Elementen definiert werden. Die Attribute-Elemente stellen die Beziehungsmerkmale entsprechend 4.5.4 dar (siehe Abbildung 87).

```
    <LinkClassLib Name="MechanischeVerbindungen">
        <LinkClass Name="SchraubVerbindung" ID="0ED13EC6-3FF3-4323-A6AF-0E61EFC2494E"
            InterfaceSideA="Außengewinde" InterfaceSideB="InnenGewinde">
            <Attribute Name="DurchmesserCompatibility" ID="3B470525-61EB-400A-83FB-4D4B17837BFA"
                AttributeDataType="BOOL" >
                <Formula xmlns:mml="http://www.w3.org/1998/Math/MathML">
                    <mml:math>
                        <mml:mi>DurchmesserCompatibility</mml:mi>
                        <mml:mo>=</mml:mo>
                        <mml:mo>(</mml:mo>
                        <mml:mi>SideA.Gewindenenndurchmesser</mml:mi>
                        <mml:mo>==</mml:mo>
                        <mml:mi>SideB.Gewindenenndurchmesser</mml:mi>
                        <mml:mo>)</mml:mo>
                    </mml:math>
                </Formula>
            </Attribute>
            …
        </LinkClass>
…
    </LinkClassLib>
```

**Abbildung 87 – Darstellung einer Verbindungsklasse in CAEX++**

Die Verbindung wird beschrieben und die Regeln der Verbindungsklasse werden genutzt um die Verbindung zu evaluieren. Entsprechend dem Vorschlag in 5.7 werden die Instanz-Attribute nicht explizit definiert, sondern von der Klassendefinition über Referenz geerbt (siehe Abbildung 88).

```
<InstanceHierarchy Name="Mechanisches Modell" TimeStamp="2014-10-29T08:19:00">
    <InternalElement ID="c92893a6-6b29-49f5-9409-d8e0542ec5a9" Name="System Schraube+Mutter">
        <InternalElement ID="1320cec9-d937-4c40-b791-454a67a5aaf3" Name="Mutter1234"
            RefBaseSystemUnitPath="MechanischeElemente/M8_Mutter_D_439_04_BM_8_S">
            <ExternalInterface Name="InnenGewinde" RefBaseClassPath="MechanischeSchnittstellenLib/InnenGewinde"/>
        </InternalElement>
        <InternalElement ID="20cd2b27-09d1-4ec0-92f2-4fe6c9b1c6eb" Name="Schraube6789"
            RefBaseSystemUnitPath="MechanischeElemente/M8_Schraube_D_933_8.8_U_M_8x16_S">
            <ExternalInterface Name="AußenGewinde"
                RefBaseClassPath="MechanischeSchnittstellenLib/AußenGewinde"/>
        </InternalElement>
        <InternalLink Name="SchraubVerbindung4567"
            RefPartnerSideA="{1320cec9-d937-4c40-b791-454a67a5aaf3}:InnenGewinde"
            RefPartnerSideB="{20cd2b27-09d1-4ec0-92f2-4fe6c9b1c6eb}:AußenGewinde"
            RefBaseClassPath="MechanischeVerbindungen/SchraubVerbindung">
            <Attribute Name="DurchmesserCompatibility"><Value>True</Value></Attribute>
            <Attribute Name="RichtungsCompatibility"><Value>True</Value></Attribute>
            <Attribute Name="LaengenCompatibility"><Value>True</Value></Attribute>
            <Attribute Name="GewindeArtCompatibility"><Value>True</Value></Attribute>
        </InternalLink>
    </InternalElement>
</InstanceHierarchy>
```

**Abbildung 88 – Darstellung einer Verbindung nach positiver Evaluierung**

Anhand des einfachen Beispiels kann die Definition von Attributen basierend auf Merkmalbeschreibungen, die Vererbung von Attributen zwischen Klassendefinition und Klasseninstanz, die Definition einer Beziehungsklasse und die Validierung der Verbindung basierend auf den Regeln der Beziehungsklasse demonstriert werden

### 6.2.2 Signal-Verbindungen

Signal-Verbindungen werden auf verschiedenen Stufen des in 4.2 dargestellten Entwicklungsprozesses verwendet. Im Verlauf des Entwicklungsprozesses wächst die Zahl der relevanten Merkmale eines Signals. Wie in 2.3.2.6 dargestellt, sind die Merkmale für ein Signal der zeitliche Verlauf, der Signalträger, der Wertevorrat und der Wertebereich (Grenzen der erlaubten Werte). Außer diesen Standardmerkmalen für ein Signal gibt es noch spezifische Merkmale, zum Beispiel Merkmale die spezifisch für den Signalträger sind. Nicht jedes der genannten Merkmale ist in jeder Phase des Entwicklungsprozesses definiert, es erfolgt eine schrittweise Verfeinerung des Signal-Modells (siehe Tabelle 20).

**Tabelle 20 – Relevante Merkmale der Signale für unterschiedliche Entwurfsstadien**

| Modellierungsebene | Signal-Merkmal |
|---|---|
| Funktionales Modell | Bezeichnung<br>semantische Bedeutung |
| Komponentenmodell | Signalträger<br>Wertevorrat |
| Topologiemodell / SW-Modell | Wertebereich<br><spezifische Merkmale> |

Die Signal-Merkmale auf Ebene des Topologiemodells sind stark abhängig von der Art der Implementierung. Wird das Signal innerhalb einer Software übertragen, werden die

Signalmerkmale durch das entsprechende SW-System bestimmt. Zum Beispiel bei einer Implementierung mit einer Programmiersprache wird durch die Sprachdefinition bestimmt, wie Daten zwischen verschiedenen Komponenten des Systems übertragen werden.

Bei einer Implementierung mit einer physikalischen Datenübertragung werden wiederum andere Merkmale verwendet. Zum Beispiel definiert das IEC CDD für ein Ausgangssignal eines Drucksensoren 10 verschiedene Umsetzungsvarianten (analoger Stromausgang, analoger Spannungsausgang, Frequenzausgang, Binärausgang IEC 60947-5-6 (NAMUR), Binärstromausgang, Binärausgang (Kontakt), Binärausgang (elektronisch), herstellerspezifischer Ausgang, pneumatischer/hydraulischer Ausgang, Binärausgang (pneumatisch/hydraulisch)), welche jeweils durch einen Satz von Merkmalen beschreiben werden.

Diese Verfeinerungsstufen können in CAEX durch eine Vererbungshierarchie von Interface-Klassen abgebildet werden, bei der die Klassen jeweils durch einen entsprechenden Satz von Attributen definiert werden (siehe Abbildung 89).

```
    <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="KomponentenAusgang" ID="61AF003C-61B9-4BA1-933A-862B312308FB" >
        <Attribute Name="Signalträger" />
        <Attribute Name="Wertevorrat" />
    </InterfaceClass>
    <InterfaceClass Name="AnalogAusgang" ID="BF9D11DF-33B9-41CB-A2E0-C93EAE82575A"
        RefBaseClassPath="KomponentenAusgang">
        <Attribute Name="Signalträger" />
        <Attribute Name="Wertevorrat" ><Value>Analog</Value></Attribute>
        <Attribute Name="Wertebereich" />
    </InterfaceClass>
    <InterfaceClass Name="Analoger Stromausgang [4]" ID="0FC94D4B-EB7C-4848-A4C0-9B8B652A6AB2"
        RefBaseClassPath="AnalogEingang">
        <Attribute Name="Signalträger" ><Value>Strom</Value></Attribute>
        <Attribute Name="Wertevorrat" ><Value>Analog</Value></Attribute>
        <Attribute Name="Wertebereich" />
        <Attribute Name="Eingestellte Größe [5]">
            <RefSemantic CorrespondingAttributePath="iec@ABC122" />
            <Attribute Name="Eingestellter Bereich für Druck">
                <RefSemantic CorrespondingAttributePath="iec@ABC115" />
…
        <RefSemanticClass CorrespondingClassPath="iec@ABC089" />
    </InterfaceClass>
```

**Abbildung 89 – Refinement von Schnittstellen durch Vererbung dargestellt**

In Bezug auf die Kopplung von Systemelementen können entsprechend Abschnitt 4.5.3 Regeln definiert werden bei dem die Merkmale der gekoppelten Systemelemente verglichen werden (siehe Tabelle 21). Die Regeln können entsprechenden Merkmalen der Signal-Verbindung zugeordnet und als ‚Interoperabilitätskriterien' bewertet werden.

**Tabelle 21 – Regeln für Signalverbindungen**

| Signalmerkmal | Regel | Erläuterung zur Regel | Interoperabilitäts-Merkmal |
|---|---|---|---|
| Signalträger | == | Senke und Quelle sollten gleichen Signalträger unterstützen | Signalträger-Verträglichkeit |
| Wertevorrat | == | Senke und Quelle sollten gleichen Wertevorrat unterstützen | Wertevorrat-Verträglichkeit |

| Signalmerkmal | Regel | Erläuterung zur Regel | Interoperabilitäts-Merkmal |
|---|---|---|---|
| Wertebereich | <= | Senke muss mindestens gleichen Wertebereich wie Quelle unterstützen | Wertebereich-Verträglichkeit |
| Bedeutung | | | semantische Verträglichkeit, pragmatische Verträglichkeit |

So, wie die Signal-Beschreibung im Verlauf des Entwicklungsprozesses verfeinert wird, gelten für die Signal-Verbindungen entsprechend verfeinerte Regeln. Diese Verfeinerung kann durch eine Vererbungshierarchie der Verbindungsklassen dargestellt werden (siehe Abbildung 90).

```
<LinkClassLib Name="Signal-Verbindungen">
    <LinkClass ID="B83D62D7-A330-4B4B-B644-2AAF60F53D7F" Name="KomponentenSignalverbindung"
        InterfaceSideA="C24A524D-D3CD-4ECC-9905-A862FF1B4E7F"
        InterfaceSideB="0FC94D4B-EB7C-4848-A4C0-9B8B652A6AB2" >
        <Attribute ID="3B470525-61EB-400A-83FB-4D4B17837BFA" Name="SignalträgerCompatibility"
            AttributeDataType="BOOL"><Formula>…</Formula>
        </Attribute>
        <Attribute ID="C1A4EC75-1730-4F7B-9612-DFF14E75C466" Name="WertevorratCompatibility"
            AttributeDataType="BOOL"><Formula>…</Formula>
        </Attribute>
    </LinkClass>
    <LinkClass ID="5A499F08-B29D-40FD-BD9C-788DC0D8D9E7" Name="AnalogSignalverbindung"
        RefBaseClassPath="KomponentenSignalverbindung"
        InterfaceSideA="72131226-3406-4EDE-8E1B-90EAFA707328"
        InterfaceSideB="BF9D11DF-33B9-41CB-A2E0-C93EAE82575A" >
        <Attribute ID="C1A4EC76-1730-4F7B-9612-DFF14E75C466" Name="WertebereichCompatibility"
            AttributeDataType="BOOL">
            <Formula>
                <mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML">
                  <mml:mi>WertebereichCompatibility</mml:mi>
                  <mml:mo>=</mml:mo>
                  <mml:mo>(</mml:mo>
                  <mml:mo>(</mml:mo>
                  <mml:mi>SideA_Wertebereich_UntereGrenze</mml:mi>
                  <mml:mo>&amp;gt;</mml:mo>
                  <mml:mo>=</mml:mo>
                  <mml:mi>SideB_Wertebereich_UntereGrenze</mml:mi>
                  <mml:mo>)</mml:mo>
                  <mml:mo>&amp;amp;</mml:mo>
                  <mml:mo>&amp;amp;</mml:mo>
                  <mml:mo>(</mml:mo>
                  <mml:mi>SideA_Wertebereich_ObereGrenze</mml:mi>
                  <mml:mo>&amp;lt;</mml:mo>
                  <mml:mo>=</mml:mo>
                  <mml:mi>SideB_Wertebereich_ObereGrenze</mml:mi>
                  <mml:mo>)</mml:mo>
                  <mml:mo>)</mml:mo>
                 </mml:math>
              </Formula>
        </Attribute>
    </LinkClass>
</LinkClassLib>
```

**Abbildung 90 – Refinement von Verbindungsklassen durch Vererbung dargestellt**

In Abbildung 90 sind in der Link-Klasse „KomponentenSignalverbindung“ die Regeln für die Prüfung auf Komponentenebene verkürzt dargestellt (ohne Formel für den einfachen Vergleich). Für die Prüfung auf Ebene des Topologiemodells ist die Prüfung für untere und obere Grenze des Werbebereichs dargestellt.

## 6.3 Anwendungsbeispiel „Flexibles Fertigungssystem“

### 6.3.1 Funktionales Modell

Ausgehend von der Beschreibung eines Endproduktes wird das funktionale Modell zu Fertigung dieses Produktes erstellt. Eines der Produkte die auf der Flexiblen Fertigungsanlage hergestellt werden, ist ein Spielzeugauto. In diesem Abschnitt wird das funktionale Modell der Fahrzeugfertigung mittels Formalisierter Prozessbeschreibung repräsentiert (siehe Abbildung 91). In dem funktionalen Modell sind Fertigungsfunktionen (Montieren) und logistische Funktionen (Lager + Transport) als Operatoren („… bereitstellen“ und „… einlagern“) dargestellt.

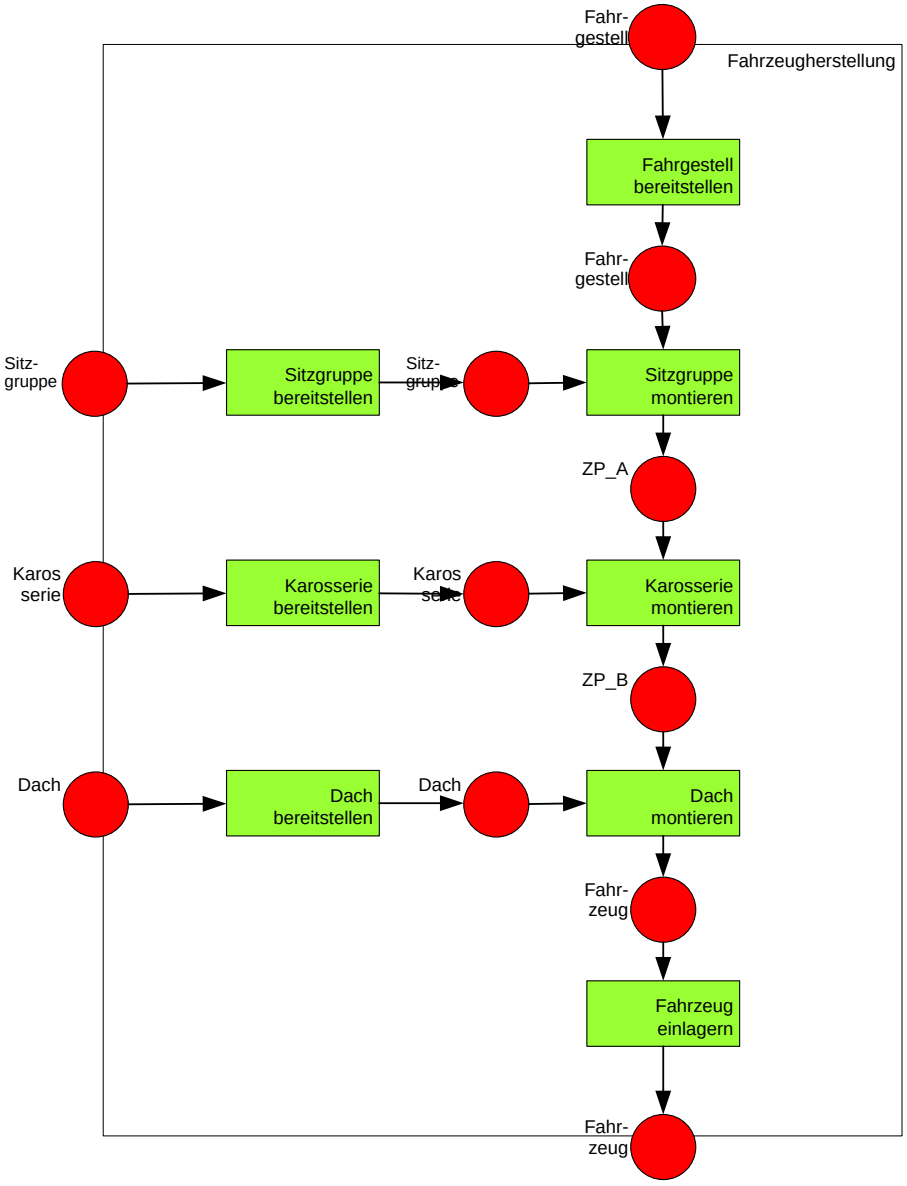


**Abbildung 91 – Modell für die Produktion eines Spielzeugautos**

Wie in 3.2.2 dargestellt, können Prozessoperatoren und Produkt-Objekte als Merkmalträger aufgefasst und mit Merkmalen beschrieben werden. Beispiele für Merkmale der Produkt-Objekte sind in Tabelle 22 dargestellt.

**Tabelle 22 – Merkmale des Produkts „Spielzeugauto“ und seiner Elemente**

| Merkmal | Fahrgestell | Sitzgruppe | Karosserie | Dach | Gesamt |
|---|---|---|---|---|---|
| Länge(mm) | 136 | 90 | 137 | 76 | 147 |
| Breite (mm) | 66 | 51 | 62 | 56 | 66 |
| Höhe (mm) | 36 | 27 | 47 | 27 | 64 |
| Gewicht (g) | 54 | 19 | 34 | 26 | 133 |

Während der konzeptionellen Entwurfsphase (in der funktionale Modelle genutzt werden), sind die Eigenschaften und Merkmale des zu entwerfenden Systems noch nicht bekannt. Aber die Eigenschaften und Merkmale des Produktes sind bekannt und können zur Definition von Anforderungen genutzt werden. Zum Beispiel besteht das Spielzeugauto aus 4 Teilen (Fahrgestell, Sitzgruppe, Karosserie und Dach).

Betrachtet man das Endprodukt „Spielzeugauto“ wiederum als System, so ergeben sich die Eigenschaften dieses Systems aus den Eigenschaften seiner Elemente und ihrer Beziehung zueinander. Leicht zu erkennen ist das am Merkmal „Gewicht“, bei dem das Merkmal Gewicht des Endprodukts aus der Summe aller Elemente gebildet wird. Bei den geometrischen Dimensionen des Systems ist es nicht so einfach. Hier wird das Merkmal des Systems nicht nur durch die Dimensionen der Elemente definiert, sondern auch durch die Position der Elemente zueinander.

$$Dim_{System} := \max(Pos_{Element} + Dim_{Element})$$

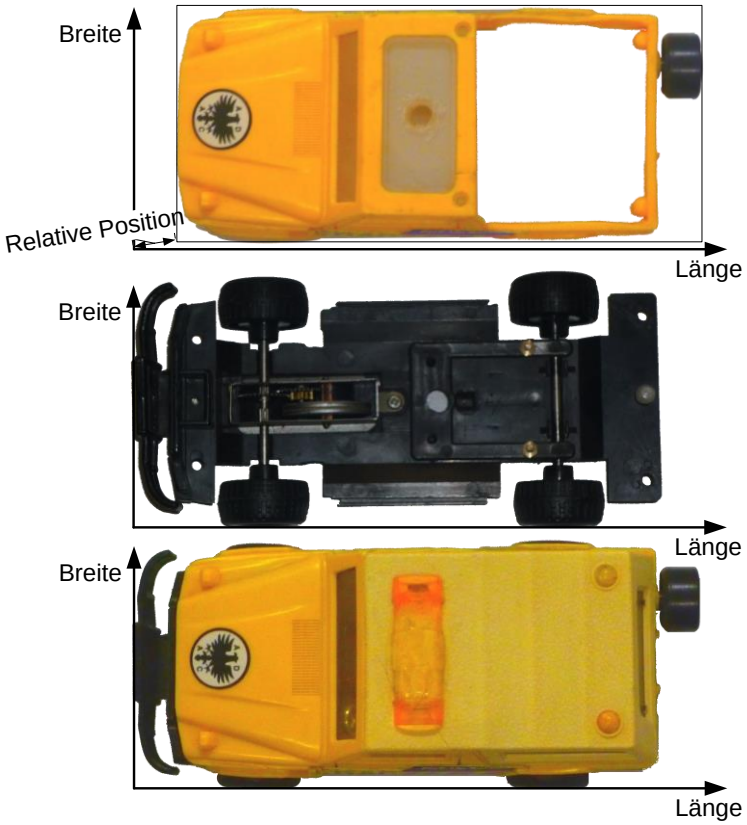


**Abbildung 92 – System „Spielzeugauto“ mit 2 Elementen**

Basierend auf den Eigenschaften des Produkts können Anforderungen an das Fertigungssystem abgeleitet werden, zum Beispiel in Bezug auf das zu transportierende Gewicht oder die geometrischen Dimensionen des Systems. Außerdem können aus der geplanten Fertigungsmenge auch zeitliche Anforderungen (z.B. maximale Zeitdauer) an die Funktion-Ausführung im System abgeleitet werden.

Für das Flexible Fertigungssystem sind weitere Fertigungsabläufe zur Produktion weiterer Produkte geplant, aber hier nicht dargestellt. Für diese Fertigungsabläufe ist zum Beispiel das Bereitstellen der Funktion „Fräsen“ notwendig.

### 6.3.2 Komponentenmodell

Das funktionale Modell des flexiblen Fertigungssystems dient als Ausgangspunkt für ein Komponentenmodell, welches auf einem hohen Abstraktionsniveau die Elemente des Fertigungssystems darstellt. Die im funktionalen Modell definierten Eigenschaften der Funktionen können als Anforderungen an die Komponenten, welche die jeweiligen Funktionen erbringen, aufgefasst werden.

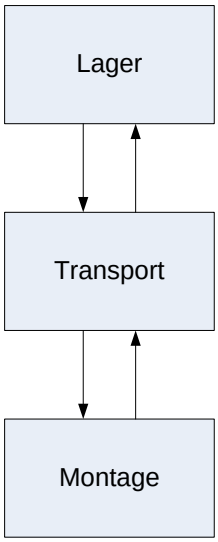


**Abbildung 93 – Einfaches Komponentenmodell für die Flexible Fertigungsanlage**

Das Komponentenmodell wird mit dem funktionalen Modell abgeglichen. Der Abgleich kann zum Beispiel dadurch erfolgen, dass in der formalisierten Prozessbeschreibung (siehe Abbildung 91) eine Zuweisung von Prozessoperatoren zu technischen Ressourcen erfolgt.

Diese Zuweisung der ausgeführten Funktionen zu technischen Ressourcen kann in CAEX++ durch <ElementRelation>-Elemente dargestellt werden (siehe Abbildung 94).

```
    <ElementRelationCollection Name="InterModelRelations">
        <ElementRelation Name="ComponentImplementsFunction1" RelationType="implements"
            RefPartnerSideA="{3189f7c1-c0c8-4705-a887-9a84dec0161b}"
            RefPartnerSideB="{b8ac05a9-b421-4627-b1c4-0cca608aa1b4}" Direction="to_SideB"/>
        <ElementRelation Name="ComponentImplementsFunction2" RelationType="implements"
            RefPartnerSideA="{3189f7c1-c0c8-4705-a887-9a84dec0161b}"
            RefPartnerSideB="{1fe53dad-91d3-43fe-97c5-d32e18f4db35}" Direction="to_SideB"/>
        <ElementRelation Name="ComponentImplementsFunction3" RelationType="implements"
            RefPartnerSideA="{3189f7c1-c0c8-4705-a887-9a84dec0161b}"
            RefPartnerSideB="{1849e997-6330-40e2-854f-8c7a9863127e}" Direction="to_SideB"/>
…
    </ElementRelationCollection>
```

**Abbildung 94 – Zuordnung von Prozessoperator zu technischer Ressource**

### 6.3.3 Topologiemodell, Stellenplan

Das Komponentenmodell wird verfeinert, so dass ein detailliertes Modell der Anlage entsteht. Abbildung 95 stellt zum Beispiel die kompositionelle Struktur der Komponente „Transport“ aus Abbildung 93 dar.

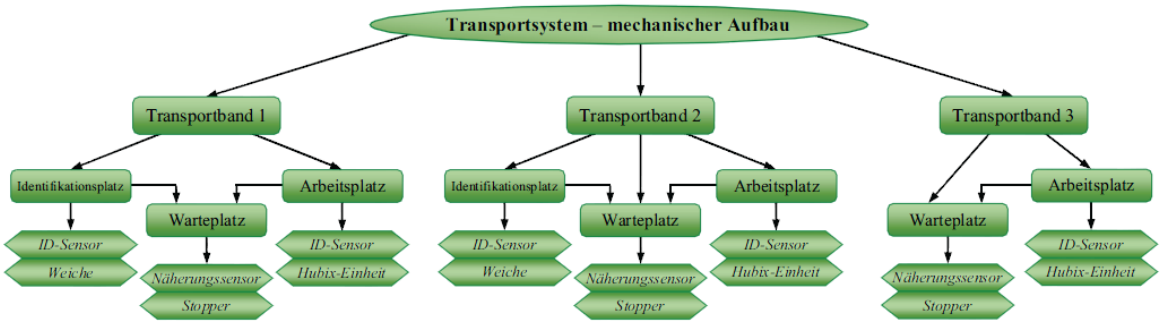


**Abbildung 95 – Detailliertes Modell des Transportsystems [Wil11, S. 17f.]**

Dieses detaillierte mechanische Modell der Anlage wird in einem CAEX++-Modell der Anlage entsprechend repräsentiert (siehe Abbildung 96). Das Modell wird durch ein <SystemHierarchy>-Element (dunkelgrün, benannt „Mechanisches System“) repräsentiert, die Elemente des Systems werden durch <InternalElement>-Elemente (hellgrün) dargestellt. Die dargestellte Hierarchie entspricht der Kompositionsstruktur der Elemente.

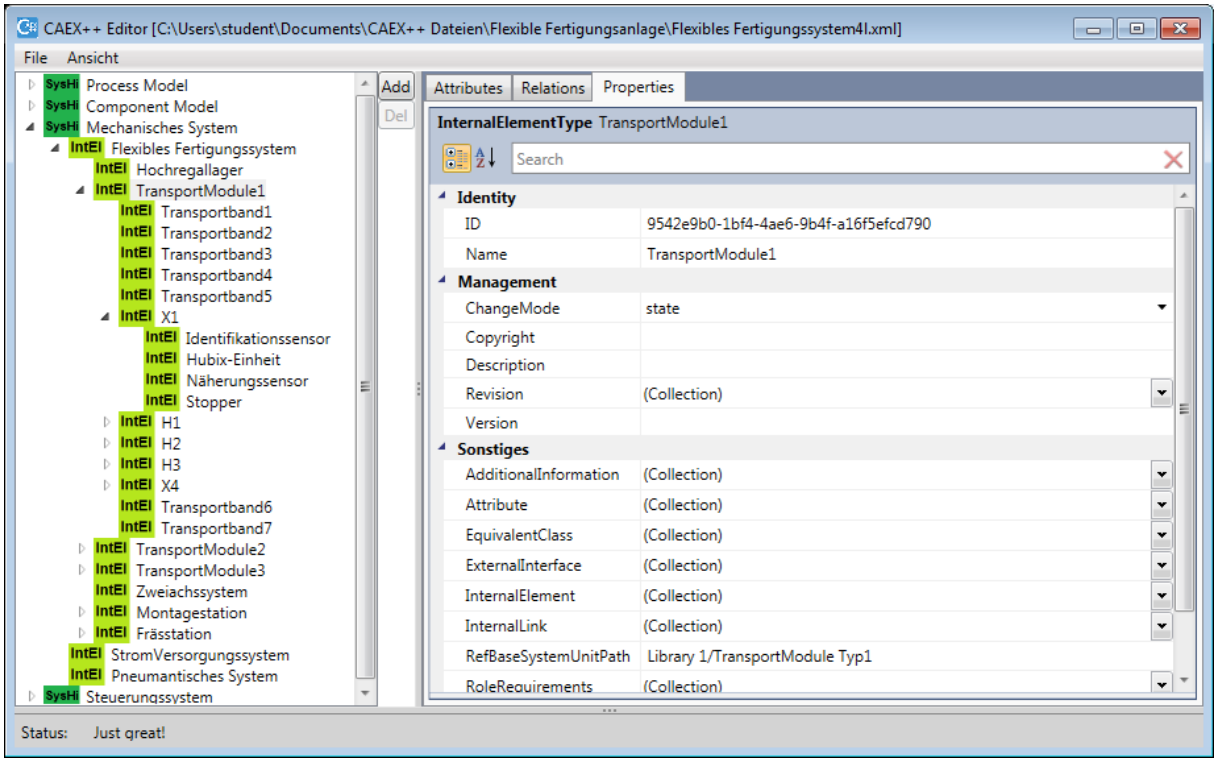


**Abbildung 96 – Detailliertes Modell des Fertigungssystems in CAEX++**

Wiederum ist es möglich, Implementierungsbeziehungen zwischen Komponentenmodell und detailliertem Modell der Anlage zu definieren (siehe Abbildung 97).

```
    <ElementRelationCollection Name="InterModelRelations">
…
        <ElementRelation Name="RessourceRealizesComponent1" RelationType="realize"
            RefPartnerSideA="{2fe44bcd-16c8-4aaa-9d44-71779bba1270}"
            RefPartnerSideB="{9542e9b0-1bf4-4ae6-9b4f-a16f5efcd790}" Direction="to_SideB"/>
        <ElementRelation Name="RessourceRealizesComponent2" RelationType="realize"
            RefPartnerSideA="{3189f7c1-c0c8-4705-a887-9a84dec0161b}"
            RefPartnerSideB="{f507c09e-3eba-44b2-9af0-7dfe3ac785f7}" Direction="to_SideB"/>
…
    </ElementRelationCollection>
```

**Abbildung 97 – <realize>-Beziehungen zwischen Elementen verschiedener Modelle**

Dadurch ist es möglich Implementierungsbeziehungen über mehrere Ebenen der Modellierung hinweg darzustellen. Es ist auch nach der detaillierten Modellierung möglich, nachzuvollziehen, welche Funktion mit einer Komponente ausgeführt werden soll.

Abbildung 97 zeigt, wie Implementierungsbeziehungen über mehrere Ebenen der Modellierung nachvollzogen werden können. Alle Modelle sind in einem <CAEXFile>-Element gespeichert, die Darstellung hier zeigt dreimal die gleiche Datei, wobei jeweils ein anderes Modell in der Projektansicht des Editors erweitert dargestellt wird. Das Prozessmodell auf der linken Seite des Bildes repräsentiert das Prozessmodell aus Abbildung 91. Das Komponentenmodell in der Mitte repräsentiert das Komponentenmodell aus Abbildung 93. Das mechanische System rechts wurde in Abbildung 96 eingeführt.

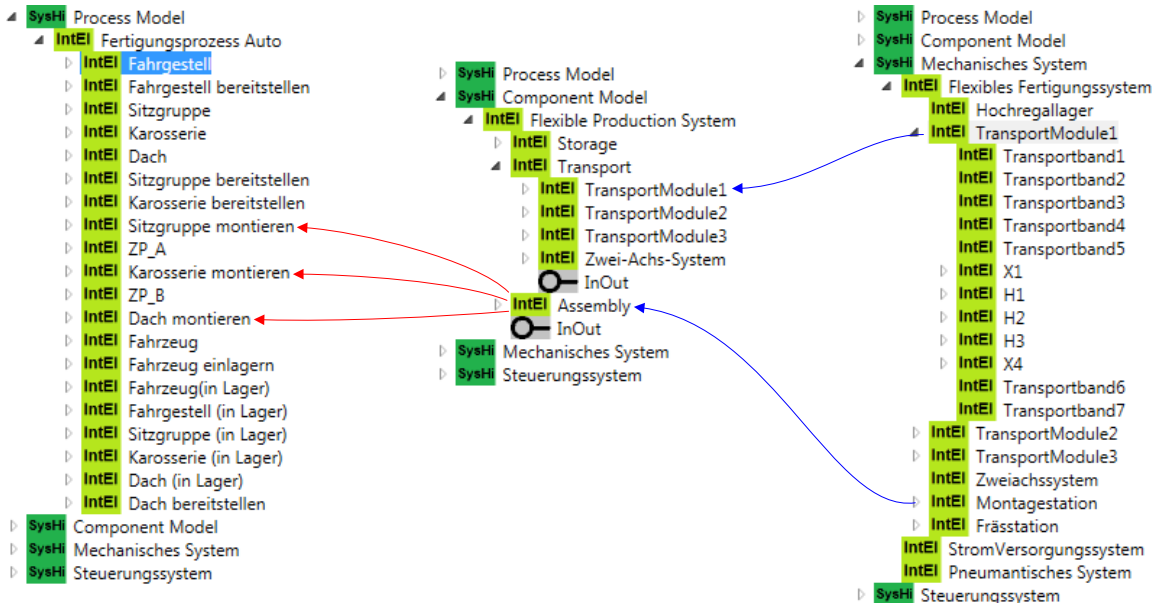


**Abbildung 98 – Verschiedene Elementbeziehungen für das Flexible Fertigungssystem**

Die dargestellten Pfeile entsprechen den Elementbeziehungen aus Abbildung 94 (rot) und aus Abbildung 97 (blau). So ist es zum Beispiel möglich, die Funktionen „Sitzgruppe montieren“, „Karosserie montieren“ und „Dach montieren“ im detaillierten Modell der Montagestation zuzuordnen und zu prüfen, ob dieses Element diese Funktionen ausführen kann. Diese Prüfung kann zum Beispiel beinhalten, dass geprüft wird, ob die Produktkomponenten durch die Montagestation gehandhabt werden können.

## 6.4 Zusammenfassung

Die Anwendung des integrierten System-Merkmalmodels wurde anhand von Anwendungsbeispielen dargestellt. Dabei wurde sowohl die Verwendung von Merkmalen zur Prüfung von Verbindungen (siehe 6.2) als auch Verwendung von Merkmalen für die Beschreibung von Funktionen demonstriert (siehe 6.3.1). Während Abschnitt 6.2.2 darstellt, wie Merkmalbeschreibungen im Verlauf der Entwicklung durch Verfeinerung des Modells wachsen, wurde im Abschnitt 6.3 auf den generellen Fluss der Informationen über verschiedene Phasen der Entwicklung fokussiert.

# 7 Zusammenfassung und Ausblick

Ziel dieser Arbeit ist es, ein System-Engineering basierend auf Merkmalen zu unterstützen. Die wesentlichen Vorteile der Nutzung von Merkmalen im Engineering ist die Eindeutigkeit in der Definition von Systemelementen und Systemen sowie die Fähigkeit relevante Informationen durchgehend im gesamten Engineering-Prozess zu nutzen.

Dazu werden in Kapitel 2 die grundlegenden Modelle beschrieben. Ein grundlegendes Merkmalmodell wird eingeführt und ein Systemmodell wird definiert, welches die Zusammenhänge zwischen Entwurfsziel (angestrebte Funktion) und den anderen Systemaspekten (Struktur und Verhalten) darstellt.

Basierend auf dieser Grundlage wird in Kapitel 3 der aktuelle Stand der Wissenschaft und Technik diskutiert. Dabei werden die typischerweise im Engineering verwendeten Systemmodelle vorgestellt und die Systembeschreibungssprache CAEX eingeführt. Die existierende Landschaft der Merkmalmodelle wird dargestellt. Dabei wird erkenntlich, dass zwar Merkmalmodelle existieren, welche sich grundsätzlich unterscheiden, dass die existierenden Ansätze aber als prinzipiell einander entsprechend betrachtet werden können. Weiterhin werden grundsätzliche Aussagen zum Engineering-Prozess gemacht und Engineering-Prozesse für CAEX und SysML dargestellt.

In Kapitel 4 erfolgt ausgehend vom angestrebten Ziel (der Verbesserung des System-Engineering) die Darstellung eines Vorschlags für einen Engineering-Prozess (basierend auf den Ergebnissen des FAVA-Projektes). Schrittweise wird der Vorschlag zur Integration von Systemmodell und Merkmalmodell verfeinert, von einer Zuordnung zwischen Systemmodellen und Merkmalmodell bis hin zur Integration von Elementbeziehung und Merkmalbeziehung (Binnenzusammenhang und Außenzusammenhang).

Basierend auf den in Kapitel 4 dargestellten Konzepten wird in Kapitel 5 eine Erweiterung von CAEX vorgeschlagen, die als CAEX++ bezeichnet wird. Existierende CAEX-Elemente werden erweitert und neue CAEX-Elemente werden eingeführt, um einen durchgängigen Informationsfluss im gesamten Engineering-Prozess zu ermöglichen und um Merkmalbeschreibungen für Funktionen und Systemelemente in die Systembeschreibung zu integrieren.

In Kapitel 6 erfolgt die Darstellung der Anwendung der Konzepte aus Kapitel 4 und von CAEX++. Anhand von einfachen Verbindungen werden prinzipielle Vorgehensweisen erklärt und am Beispiel des Flexiblen Fertigungssystems wird ein Engineering-Prozess dargestellt.

Die dargestellte Vorgehensweise für das Engineering und die Integration von Merkmalmodell und Systemmodell wurde im Rahmen des FAVA-Projektes evaluiert und ein Nutzen der Vorgehensweise wurde nachgewiesen. Die in Kapitel 4 dargestellte Integration zwischen Merkmalmodell und Systemmodell wurde im Rahmen des IEC-Arbeitskreis TC 65E WG16 „Digital Factory“ diskutiert und wurde bei der Erstellung der Norm IEC 62832 berücksichtigt [IEC14a].

Der Autor ist davon überzeugt, dass die dargestellte Integration von System- und Merkmalmodell nicht nur das Engineering von Systemen verbessert, sondern auch einen wesentlichen Beitrag zum Thema Industrie 4.0 darstellt. Basierend auf Merkmalbeschreibungen von Funktionen und Systemkomponenten wird es möglich sein Systemkomponenten automatisch zu Produktionssystemen zu orchestrieren und Systeme automatisch zu konfigurieren (siehe auch [HGH+15]). Um die in dieser Arbeit vorgeschlagene Methodik allgemein nutzen zu können sind verschiedene Maßnahmen notwendig. Um ein Engineering oder die Orchestrierung von Produktionssystemen basierend auf funktionalen Modellen zu ermöglichen, ist es notwendig, dass System-Funktionen klassifiziert und mit Merkmalen beschrieben werden. Existierende Merkmalbeschreibungen für Betriebsmittel müssen geprüft werden, ob sie für ein Engineering geeignet sind. Wichtig ist hierbei die Bereitstellung von Merkmalen zur Beschreibung der Schnittstellen (siehe 3.4.10.1). Weiterhin müssen Merkmalbeschreibungen von Beziehungen zwischen System-Elementen erstellt werden. Das heißt es müssen Bibliotheken von Verbindungs-Typen erstellt werden, welche die Prüfung von Merkmal-Zusammenhängen ermöglichen.

# Literatur

[ADK+06] Ahrens, W.; Drathen, H.; Kroll, O.; Löffelmann, G.; Zgorzelski, P.: Standardisierte Merkmale als Schlüssel für den unternehmensweiten Datenaustausch im Engineering-Umfeld.. In (Schnieder, E. Hrsg.): EKA 2006. Entwurf komplexer Automatisierungssysteme, 9. Fachtagung. Beschreibungsmittel, Methoden und Werkzeuge für Entwurf und Zuverlässigkeit von Anwendungen in Automatisierung und Verkehr. IVA, Braunschweig, 2006.

[Ahr10a] Ahrens, W.: Der Branchenstandard PROLIST und die Harmonisierung mit eCl@ss. In (Malisa, V. Hrsg.): AALE 2010 Tagungsband, 2010.

[Ahr10b] Ahrens, W.: Gegenüberstellung von VDI/VDE 3682, PROLIST, eClass. Formalisierte Prozessbeschreibung und Branchenstandards. In atp edition – Automatisierungstechnische Praxis, 2010; S. 32–45.

[Bal11] Balzert, H.: Lehrbuch der Softwaretechnik. Entwurf, Implementierung, Installation und Betrieb. Spektrum Akademischer Verlag, Heidelberg, 2011.

[Bar84] Bartsch, H.-J.: Mathematische Formeln. Fachbuchverlag, Leipzig, 1984.

[Ben12] Benson, P. R.: Vortrag: ISO 8000 Data Quality, 04.01.2012.

[Ber11] Bergmann, S.: Konzept für funkbasierte Steuerung einer 2-Achskomponenente in einer flexiblen Fertigungsanlage. Diplomarbeit, Magdeburg, 2011.

[BMQ+10] Behnen, D.; Mersch, H.; Quix, C.; Schmitz, D.; Zhang, M.; Fayzullin, K.; Brecher, C.; Epple, U.; Jarke, M.: Gemeinsamkeiten und Unterschiede in der Modellierung von prozesstechnischen und diskreten Produktionsanlagen. In (Jumar, U.; Schnieder, E.; Diedrich, C. Hrsg.): EKA 2010. Entwurf komplexer Automatisierungssysteme. ifak e.V., Magdeburg, 2010.

[BS01] Broy, M.; Stølen, K.: Specification and development of interactive systems. Focus on streams, interfaces, and refinement. Springer, New York, 2001.

[BWW+10] Birkhofer, R.; Wollschläger, M.; Winzenick, M.; Kalhoff, J.; Kleedörfer, C.; Mühlhause, M.; Niemann, J.; Schrieber, R.: Life-Cycle-Management für Produkte und Systeme der Automation. Ein Leitfaden des Arbeitskreises Systemaspekte im ZVEI Fachverband Automation. Zentralverb. Elektrotechnik- und Elektronikindustrie, Fachverb. Automation, Frankfurt, M, 2010.

[Che76] Chen, P. P.-S.: The Entity-Relationship Model - Toward a Unified View of Data. In ACM Transactions on Database Systems, 1976, vol. 1; S. 9–36.

[CRK08] Chun, B.-G.; Ratnasamy, S.; Kohler, E.: NetComplex: A Complexity Metric for Networked System Designs: NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation. USENIX Assoc., Berkeley, CA, USA, 2008; S. 393–406.

[Dav93] Davenport, T. H.: Process innovation. Reengineering work through information technology. Harvard business school, Boston (Mass.), 1993.

[DIN00] DIN: DIN-Merkmallexikon. DINsml.net. http://www.dinsml.net/portal/info.seam, Geprüft am: 28.01.2014.

[DIN07] DIN 4000-160: Sachmerkmal-Leisten - Teil 160: Verbindungselemente mit Außengewinde, DIN, 2007-02.

[DIN10] DIN ISO 724: Metrische ISO-Gewinde allgemeiner Anwendung - Grundmaße, DIN, 2010-01.

[DIN12] DIN 4000-1: Sachmerkmal-Listen – Teil 1: Begriffe und Grundsätze, DIN, September 2012.

[DIN13] DIN 4000-102: Sachmerkmal-Listen – Teil 102: Datenaustausch für Sachmerkmallisten mittels XML-Schema, DIN, Dezember 2013.

[DIN75] DIN 4000-1 Beiblatt 1: Sachmerkmal-Leisten; Grundsätze, Anwendung in Sachmerkmal-Verzeichnissen, DIN, 1975-02.

[DKE12a] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: Deutsche Online-Ausgabe des IEV - Teil 351: Leittechnik. http://www.dke.de/de/Online-Service/DKE-IEV/Seiten/IEV-Woerterbuch.aspx, Geprüft am: 17.12.2012.

[DKE12b] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: Deutsche Online-Ausgabe des IEV - Teil 101: Mathematik. http://www.dke.de/de/Online-Service/DKE-IEV/Seiten/IEV-Woerterbuch.aspx, Geprüft am: 17.12.2012.

[Dra10] Drath, R. Hrsg.: Datenaustausch in der Anlagenplanung mit AutomationML. Integration von CAEX, PLCopen XML und COLLADA. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2010.

[Dra13] Drath, R.: Vortrag: CAEX Maintenance Cycle 2013. Status - Collection of changes, ABB Forschungszentrum Ladenburg, 04.07.2013.

[DV14] DIN 40912: Kernmodelle - Beschreibung und Beispiele, DIN; VDE, 2014-11.

[ECC14a] ECCMA: ECCMA Standard Class Dictionary (ESCD). Product Search. http://www.eotd.org/prod_search/, Geprüft am: 29.12.2013.

[ECC14b] ECCMA: ECCMA - eOTD Search. http://www.eotd.org/prod_search/index.php, Geprüft am: 24.02.2014.

[eCl09] eCl@ss 7.0 – Produktdatenmanagement vom Hersteller bis zum CAD. Bericht der eCl@ss-Arbeitsgruppe CAx-Anwendungen, 2009.

[eCl13] eCl@ss e.V.: eCl@ss Wiki. http://wiki.eclass.eu/wiki/Main_Page, Geprüft am: 16.10.2013.

[eCl14] eCl@ss e.V.: Der eCl@ss e.V.. http://www.eclass.de/eclasscontent/about/index.html.de, Geprüft am: 14.12.2014.

[EFH+12] Eckert, K.; Fay, A.; Hadlich, T.; Diedrich, C.; Frank, T.; Vogel-Heuser, B.: Design Patterns for Distributed Automation Systems with Consideration of Non-Functional Requirements: ETFA 2012. IEEE Press, Piscataway, NJ, 2012.

[Epp11] Epple, U.: Merkmale als Grundlage der Interoperabilität technischer Systeme. In at - Automatisierungstechnik, 2011, vol. 59; S. 440–450.

[FB08] Frey, T.; Bossert, M.: Signal- und Systemtheorie. Mit 26 Tabellen, 64 Aufgaben mit Lösungen und 84 Beispielen. Vieweg + Teubner, Wiesbaden, 2008.

[FEH+12] Frank, T.; Eckert, K.; Hadlich, T.; Fay, A.; Diedrich, C.; Vogel-Heuser, B.: Workflow and decision support for the design of distributed automation systems: INDIN 2012. IEEE 10th International Conference on Industrial Informatics. IEEE Press, Piscataway, NJ, 2012; S. 293–299.

[FEH+13] Frank, T.; Eckert, K.; Hadlich, T.; Fay, A.; Diedrich, C.; Vogel-Heuser, B.: Erweiterung des V-Modells ® für den Entwurf von verteilten Automatisierungssystemen. In at - Automatisierungstechnik, 2013, vol. 61; S. 79–91.

[FHE+12a] Frank, T.; Hadlich, T.; Eckert, K.; Diedrich, C.; Vogel-Heuser, B.: Erweiterung des V-Modells für den Entwurf von verteilten Automatisierungssystemen. In (Jumar, U.; Schnieder, E.; Diedrich, C. Hrsg.): EKA 2012. Entwurf komplexer Automatisierungssysteme, Magdeburg, 2012a.

[FHE+12b] Frank, T.; Hadlich, T.; Eckert, K.; Fay, A.; Diedrich, C.; Vogel-Heuser, B.: Using contact points to integrate discipline spanning real-time requirements in modeling Networked Automation Systems for manufacturing systems: 2012 IEEE International Conference on Automation Science and Engineering (CASE). Green automation toward a sustainable society. IEEE, Piscataway, NJ, 2012b; S. 851–856.

[FHP+11] Feilkas, M.; Hölzl, F.; Pfaller, C.; Rittmann, S.; Schätz, B.; Schwitzer, W.; Sitou, W.; Spichkova, M.; Trachtenherz, D.: A Refined Top-DownMethodology for the Development of Automotive Software Systems - The KeylessEntry-System Case Study -. TUM-I1103. Report, München, 2011.

[Fou94] Foucault, M.: Die Ordnung der Dinge. Eine Archäologie der Humanwissenschaften. Suhrkamp, Frankfurt am Main, 1994.

[FP97] Fenton, N. E.; Pfleeger, S. L.: Software metrics. A rigorous and practical approach. PWS Pub., London, Boston, 1997.

[Fra14] Frank, T.: Entwicklung und Evaluation einer Modellierungssprache für den Architekturentwurf von verteilten Automatisierungsanlagen auf Basis der Systems Modelling Language (SysML). Dissertation, München, 2014.

[Fre02] Frey, G.: Software Quality in Logic Controller Programming. In Proc. of IEEE SMC 2002, Tunisia, 2002; S. 515–520.

[FVF+15] Fay, A.; Vogel-Heuser, B.; Frank, T.; Eckert, K.; Hadlich, T.; Diedrich, C.: Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns. In Journal of Systems and Software, 2015, vol. 101; S. 221–235.

[Ges10] Gester, C.: Generisches Konzept für die Integration von Klassifikationsschemata in semantische Modelle. Diplomarbeit, Magdeburg, 2010.

[GF08] Güttel, K.; Fay, A.: Beschreibung von fertigungstechnischen Anlagen mittels CAEX. In atp, 2008; S. 34–39.

[Gie90] Gierse, F. J.: Funktionen und Funktionsstrukturen: Zentrale Werkzeuge der Wertanalyse: Wertanalyse, Wertgestaltung, Value Management: Neue Impulse zum ganzheitlichen Problemelösen. Tagung Nürnberg, 13. und 14. November 1990. VDI-Verlag, Düsseldorf, 1990.

[Gla73] Glassman, R. B.: Persistence and loose coupling in living systems. In Behavioral Science, 1973, vol. 18; S. 83–98.

[Gol99] Goldstein, J.: Emergence As A Construct: History and Issues. In Emergence: Complexity and Organization, 1999, vol. 1; S. 49–72.

[Had12] Hadlich, T.: Complexity metrics for Engineering Models. IFAT-LIA 2/2012. Technical Report. http://ifatwww.et.uni-magdeburg.de/~hadlich/en/publications/Report2012-2_Complexity_metrics_for_Engineering_Models.pdf, Magdeburg, 2012.

[Hal94] Halbach, W. R.: Interfaces. Medien- und kommunikationstheoretische Elemente einer Interface-Theorie. Fink, München, 1994.

[HD13] Hadlich, T.; Diedrich, C.: Using properties in systems engineering: ETFA 2013. IEEE 18th International Conference on Emerging Technologies and Factory Automation. IEEE, Piscataway, NJ, 2013.

[HDE+11] Hadlich, T.; Diedrich, C.; Eckert, K.; Frank, T.; Fay, A.; Vogel-Heuser, B.: Common communication model for distributed automation systems: INDIN 2011. IEEE 9th International Conference on Industrial Informatics. IEEE Press, Piscataway, NJ, 2011; S. 131–136.

[HED+12] Hadlich, T.; Engel, C.; Diedrich, C.; Mühlhause, M.: Konzept und Erfahrungen beim Abgleich mehrerer Domain-Ontologien. In (Jumar, U.; Schnieder, E.; Diedrich, C. Hrsg.): EKA 2012. Entwurf komplexer Automatisierungssysteme, Magdeburg, 2012; S. 81–90.

[Hee05] Heeg, M.: Ein Beitrag zur Modellierung von Merkmalen im Umfeld der Prozessleittechnik. VDI-Verlag, Düsseldorf, 2005.

[Heh11] Hehenberger, P.: Computerunterstützte Fertigung. Eine kompakte Einführung. Springer Berlin, Berlin[u.a], 2011.

[Hel12] Helgermann, S.: Entwicklung einer generischen Darstellungsmethode für grafenbasierte Strukturen von Produktions- und Automatisierungssystemen mittels AutomationML. Bachelorarbeit, Magdeburg, 2012.

[HGH+15] Höme, S.; Grützner, J.; Hadlich, T.; Diedrich, C.; Schnäpp, D.; Arndt, S.; Schnieder, E.: Semantic Industry: Herausforderungen auf dem Weg zur rechnergestützten Informationsverarbeitung der Industrie 4.0. In at - Automatisierungstechnik, 2015, vol. 63.

[HHD+12] Hadlich, T.; Höme, S.; Diedrich, C.; Eckert, K.; Frank, T.; Fay, A.; Vogel-Heuser, B.: Time as non-functional requirement in distributed control systems.: ETFA 2012. IEEE Press, Piscataway, NJ, 2012.

[Hit07] Hitchins, D. K.: Systems engineering. A 21st century systems methodology. Wiley, Chichester, 2007.

[Hit13] Hitchins, D. K.: System Of Systems - The Ultimate Tautology?. http://www.hitchins.net/profs-stuff/profs-blog/system-of-systems---the.html, Geprüft am: 30.07.2013.

[HM03] Huang, G. Q.; Mak, K. L.: Internet Applications in Product Design and Manufacturing. Springer Berlin Heidelberg, 2003.

[HMD10] Hadlich, T.; Mühlhause, M.; Diedrich, C.: Discovery and integration of information in a heterogeneous environment: ETFA 2010. IEEE 15th International Conference on Emerging Technologies and Factory Automation. IEEE Press, Piscataway, NJ, 2010; S. 1–8.

[HSD13] Hadlich, T.; Sokolov, S.; Diedrich, C.: Beurteilung der Komplexität von Engineering Modellen: Automation 2013. 14. Branchentreff der Mess- und Automatisierungstechnik, Kongresshaus Baden-Baden, 25.und 26. Juni, 2013. VDI-Verlag, 2013; S. 405–410.

[Hun12] Hundt, L.: Durchgängiger Austausch von Daten zur Verhaltensbeschreibung von Automatisierungssystemen. Dissertation, Magdeburg, 2012.

[IEC03] IEC 61499-1:2003 Function Blocks - Part 1: Architecture, IEC, 2003.

[IEC04a] IEC 61360-2: Standard data element types with associated classification scheme for electric components - Part 2: EXPRESS dictionary schema, IEC, 2004-02.

[IEC04b] IEC 61360-1: Standard data element types with associated classification scheme for electric components - Part 1: Definitions - Principles and methods, IEC, 2004-01.

[IEC06a] IEC 60050-351: International Electrotechnical Vocabulary Part 351: Control technology, IEC, 16.10.2006.

[IEC06b] IEC 61069-1: Industrial-process measurement and control – Evaluation of system properties for the purpose of system assessment –, Ed 1.0, IEC, 30.10.2006.

[IEC08] IEC 62424 - Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools (CAEX), Ed. 1.0, IEC, 12.08.2008.

[IEC09a] IEC 61987-10: Industrial-process measurement and control - Data structures and elements in process equipment catalogues - Part 10: Lists of properties (LOPs) for industrial-process measurement and control for electronic data exchange - Fundamentals, Ed 1.0, IEC, 23.07.2009.

[IEC09b] IEC 61360-1: Standard data element types with associated classification scheme for electric components - Part 1: Definitions - Principles and methods, 3.0, IEC, 2009.

[IEC10] IEC 62507-1: Identification systems enabling unambiguous information interchange - Requirements - Part1: Principles and methods, 1.0, IEC, 2010.

[IEC11a] IEC 62714-1 Engineering data exchange format for use in industrial automation systems engineering (AutomationML) Part 1 - architecture and general requirements, Ed.1.0, IEC, 07.11.2011.

[IEC11b] IEC 62683: Low-voltage switchgear and controlgear – Product data and properties for information exchange, Ed. 1, IEC, 30.09.2011.

[IEC14a] IEC 62832: Industrial-process measurement, control and automation Reference model for representation of production facilities (Digital Factory) (CD), 1.0, IEC, April 2014.

[IEC14b] IEC 62424 - Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools (CAEX), Ed. 2.0, IEC, 2014.

[IEE00] IEEE 1284-2000: IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers, IEEE, 2000.

[IEE90] IEEE Std 610.12-1990. IEEE standard glossary of software engineering terminology, IEEE, 1990.

[Ins10] Klassifikationsstandards auswählen und einsetzen. Handlungsempfehlung zum Einsatz von eBusiness-Standards, Köln, 2010.

[ISO04a] ISO/IEC 11179-1: Information technology - Metadata registries (MDR) - Part 1: Framework, Second edition, ISO/IEC, 15.09.2004.

[ISO04b] ISO 15926-1: Industrial automation systems and integration - Integration of life-cycle data for process plants including oil and gas production facilities - Part 1: Overview and fundamental principles, 1.0, ISO, 15.07.2004.

[ISO07] ISO/TS 8000-110 Information quality — Part 110: Master data quality: Exchange of master data: Syntax, semantic encoding, and conformance to data specification, ISO, 02.11.2007.

[ISO09a] ISO 29002-10: Industrial automation systems and integration - Exchange of characteristic data - Part 10: Characteristic data exchange format, ISO, 01.12.2009.

[ISO09b] ISO 29002-5: Industrial automation systems and integration - Exchange of characteristic data - Part 5: Identification scheme, ISO, 15.02.2009.

[ISO09c] ISO 25010: Systems and software engineering - Software product quality requirements and evaluation - Quality models for software product quality and system quality in use, ISO, 2009.

[ISO09d] ISO 22745-30: Industrial automation systems and integration - Open technical dictionaries and their application to master data - Part 30: Identification guide representation, ISO, 2009.

[ISO10] ISO 13584-42: Industrial automation systems and integration -- Parts library -- Part 42: Description methodology: Methodology for structuring parts families, ISO, 2010.

[Jan08] Jannes M-SP: Sketchup-Modell "Screw and Screwnuts". https://3dwarehouse.sketchup.com/model.html?id=a0dc1e537184304c9f47ccf3f3013bed.

[JCS+12] Jäger, T.; Christiansen, L.; Strube, M.; Fay, A.: Durchgängige Werkzeugunterstützung von der Anforderungserhebung bis zur Anlagenstrukturbeschreibung mittels formalisierter Prozessbeschreibung und AutomationML. In (Jumar, U.; Schnieder, E.; Diedrich, C. Hrsg.): EKA 2012. Entwurf komplexer Automatisierungssysteme, Magdeburg, 2012; S. 239–251.

[Kay03] Kaye, D.: Loosely coupled. The missing pieces of Web services. RDS Press, Marin County, Calif, 2003.

[Kla68] Klaus, G. Hrsg.: Wörterbuch der Kybernetik. Dietz Verlag, Berlin, 1968.

[Kön12] König, H.: Protocol Engineering. Springer, Berlin, Heidelberg, 2012.

[KP11] Kerzhner, A. A.; Paredis, C. J.: Model-Based System Verification: A Formal Framework for Relating Analyses, Requirements, and Tests. In (Dingel, J.; Solberg, A. Hrsg.): Models in software engineering. Workshops and symposia at MODELS 2010, Oslo, Norway, October 3-8, 2010 reports and revised selected papers. Springer, Berlin, New York, 2011; S. 279–292.

[Kra86] Krauser, D.: Methodik zur Merkmalbeschreibung technischer Gegenstände. Dissertation. Beuth, Berlin, Köln, 1986.

[Lin83] Linné, C. v.: Philosophia Botanica. verfügbar unter: http://caliban.mpiz-koeln.mpg.de/linne/philosophia/index.html, 1783.

[LPZ07] Löffelmann, G.; Polke, B.; Zgorzelski, P.: PROLIST Lists of Properties – an important step toward an integrated electronic engineering and business workflow. In atp international, 2007, vol. 5; S. 34–40.

[LSG+14] Lüder, A.; Schmidt, N.; Graeser, O.; Thron, M.; John, M.: Semantikdefinition durch Integration von Klassifikationssystemen in Entwurfsdaten zum verlustfreien Datenaustausch in Werkzeugketten: Automation 2014. 15. Branchentreff der Mess- und Automatisierungstechnik, 2014; S. 29–42.

[Lüd14] Lüder, A.: Vortrag: Semantikdefinition durch Integration von Klassifikationssystemen in Entwurfsdaten zum verlustfreien Datenaustausch in Werkzeugketten. Automation Konferenz 2014, Baden-Baden, 30.06.2014.

[Mat02] Matthiesen, S.: Ein Beitrag zur Basisdefinition des Elementmodells “Wirkflächenpaare & Leitstützstrukturen” zum Zusammenhang von Funktion und Gestalt technischer Systeme. Dissertation. http://www.ipek.kit.edu/downloads/IPEK_FB_Band6_S_Matthiesen.pdf, Karlsruhe, 2002.

[MB89] McCabe, T. J.; Butler, C. W.: Design complexity measurement and testing. In Communications of the ACM, 1989, vol. 32; S. 1415–1423.

[Mer11] Mertens, M.: Verwaltung und Verarbeitung merkmalbasierter Informationen: vom Metamodell zur technologischen Realisierung. Dissertation, 2011.

[Mit09] Mitchell, M.: Complexity. A guided tour. Oxford University Press, Oxford [England], New York, 2009.

[Müh12] Mühlhause, M.: Konzept zur durchgängigen Nutzung von Engineeringmodellen der Automation. Logos-Verl, Berlin, 2012.

[Nam03] NE 100 – Merkmalleisten zur Erstellung von PLT-Gerätespezifikationen, 1.0, Namur, 02.06.2003.

[Nam05] Projektgruppe "Merkmalleisten". Ein Beitrag zur Optimierung der Geschäftsprozesse bei Herstellern und Anwendern von Geräten und Systemen aus dem Bereich der Prozessleittechnik, Namur, 2005.

[OF06] Osterloh, M.; Frost, J.: Prozessmanagement als Kernkompetenz. Wie Sie business reengineering strategisch nutzen können. Gabler Verlag, Wiesbaden, 2006.

[OHK93] Offut, A. J.; Harrold, M. J.; Kolte, P.: A Software Metric System for Module Coupling. In The Journal of Systems and Software, 1993, vol. 20; S. 295–308.

[OMG08] OMG Systems Modeling Language (OMG SysML™), Version 1.1, OMG, November 2008.

[OMG11a] Unified Modeling Language, Superstructure, Version 2.4.1, OMG, August 2011.

[OMG11b] Unified Modeling Language, Infrastructure, Version 2.4.1, OMG, August 2011.

[ORG+11] Opgenoorth, B.; Richter, J. H.; Grosch, T.; Wolff, D.; Fay, A.: Verlässlichkeitsanforderungen in der Prozess- und Ressourcenbeschreibung. Kombination von VDI/VDE 3682 und IEC 62424 (CAEX). In atp edition – Automatisierungstechnische Praxis, 2011, vol. 53; S. 44–53.

[Pag88] Page-Jones, M.: The practical guide to structured systems design. Prentice-Hall, Englewood Cliffs, N.J, 1988.

[Pat82] Patzak, G.: Systemtechnik-Planung komplexer innovativer Systeme. Grundlagen, Methoden, Techniken. Springer, Berlin, Heidelberg, New York, 1982.

[PI 94] PI STEP Consortium: Process Plant Engineering Activity Model, 1994.

[Poh04] Pohn, R.: DIN-Merkmallexikon - Genormte Produktmerkmale und Merkmallisten reduzieren die Komplexität von Geschäftsprozessen. In DIN-Mitteilungen 11, 2004; S. 38–43.

[PRO12] Namur: NAMUR: NE 100 und PROLIST®-Datenbank. http://www.namur.de/index.php?id=352, Geprüft am: 21.02.2014.

[RS12] Richardson, G.; Schwarz, E.: Vortrag: GeoSpatial: The Ingest, Storage, and Retrieval of Geopolitical Code Sets in a 11179 Metadata Registry. Open Forum on Metadata 2012, Berlin, 2012.

[RST09] Ratiu, D.; Schwitzer, W.; Thyssen, J.: A System of Abstraction Layers for the Seamless Development of Embedded Software Systems. TUM-I0928. Report, München, 2009.

[SBD+10] Schatten, A.; Biffl, S.; Demolsky, M.; Gostischa-Franta, E.; Oestreicher, T.; Winkler, D.: Best Practice Software-Engineering. Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen. Spektrum Akademischer Verlag, Heidelberg, 2010.

[Sch05] Schäfer, H.: Grundlagen der Elektrotechnik. Welz Vermittlerverl. Mannheim, 2005.

[Sch99] Schnieder, E.: Methoden der Automatisierung. Beschreibungsmittel, Modellkonzepte und Werkzeuge für Automatisierungssysteme ; mit 56 Tabellen. Vieweg, Braunschweig, Wiesbaden, 1999.

[SF10] Strube, M.; Fay, A.: Brückenschlag zwischen Prozess- und Anlagenbeschreibung. Abbildung der VDI 3682 auf CAEX und AutomationML. In atp edition – Automatisierungstechnische Praxis, 2010, vol. 52; S. 26–27.

[SFT+11] Strube, M.; Fay, A.; Truchat, S.; Figalist, H.: Modellgestützte Modernisierungsplanung. In atp edition – Automatisierungstechnische Praxis, 2011, vol. 53; S. 46–53.

[SMC74] Stevens, W. P.; Myers, G. J.; Constantive, L. L.: Structured design. In IBM Systems Journal, 1974, vol. 13; S. 115–139.

[SS10] Schnieder, E.; Schnieder, L.: Terminologische Präzisierung des Systembegriffs. Grundlage formaler Systembeschreibungen. In atp edition – Automatisierungstechnische Praxis, 2010, vol. 52; S. 46–59.

[Tal08] Talbot, M.: Vortrag: A romp through the history of philosophy from the Pre-Socratics to the present day. Philosophy for Beginners, iTunesU, Oxford, 13.11.2008.

[TRS+10] Thyssen, J.; Ratiu, D.; Schwitzer, W.; Harhurin, A.; Feilkas, M.; Thaden, E.: A System for Seamless Abstraction Layers for Model-based Development of Embedded Software. In (Engels, G.; Luckey, M.; Schäfer, W. Hrsg.): Proceedings of Envision 2020 Workshop // Software engineering 2010. Fachtagung des GI-Fachbereichs Softwaretechnik ; 22. - 26.02.2010 in Paderborn. Ges. für Informatik, Bonn, 2010.

[UGF09] Ulrich, A.; Güttel, K.; Fay, A.: Durchgängige Prozesssicht in unterschiedlichen Domänen - Methoden und Werkzeug zum Einsatz der formalisierten Prozessbeschreibung. In at - Automatisierungstechnik, 2009, vol. 57; S. 80–92.

[VDF13] Vogel-Heuser, B.; Diedrich, C.; Fay, A.: Abschlussbericht Funktionaler Anwendungsentwurf von verteilten Automatisierungssystemen (FAVA), 2013.

[VDI93] VDI 2221 - Methodik zum Entwicklen und Konstruieren technischer Systeme und Produkte, VDI, Mai 1993.

[VDI96] VDI 2803 Blatt 1 - Funktionenanalyse - Grundlage und Methode, VDI, Oktober 1996.

[Vog03] Vogel-Heuser, B.: Systems Software Engineering. [angewandte Methoden des Systementwurfs für Ingenieure]. Oldenbourg Industrieverl., München, 2003.

[VV02] VDI/VDE 3681 - Einordnung und Bewertung von Beschreibungsmitteln aus der Automatisierungstechnik, VDI; VDE, Oktober 2002.

[VV03] VDI/VDE 3682 - Formalisierte Prozessbeschreibungen, VDI; VDE, Oktober 2003.

[W3C10] Mathematical Markup Language (MathML) Version 3.0, W3C, 21.10.2010.

[War12] Warzel, D.: Vortrag: ISO/IEC 11179 Metadata Registries (MDR) & ISO/IEC 19763 Metamodel Framework for Interoperability (MFI). Open Forum on Metadata 2012, Berlin, 2012.

[Wei07] Weilkiens, T.: Systems engineering with SysML/UML. Modeling, analysis, design. Morgan Kaufmann OMG Press/Elsevier, Amsterdam, Boston, 2007.

[Wei08] Weilkiens, T.: Systems engineering mit SysML/UML. Modellierung, Analyse, Design. Dpunkt-Verl., Heidelberg, 2008.

[WFR+10] Wolff, D.; Fay, A.; Richter, J. H.; Grosch, T.; Opgenoorth, B.: Anlage zur Modellierung und Analyse von Verlässlichkeits-Anforderungen mittels Prozess- und Ressourcen-Beschreibung. Technischer Bericht. http://www.hsu-hh.de/download-1.4.1.php?brick_id=UbSj0CLoHOlfcU74, Hamburg, 2010.

[Wik14] Wikipedia: Merkmal. http://de.wikipedia.org/w/index.php?oldid=123485816, Geprüft am: 01.02.2014.

[Wil11] Wild, P.: Modulares Automatisierungskonzept für ein flexibles Fertigungssystem. Diplomarbeit, Magdeburg, 2011.

[WKT07] Wiegand, H.; Kloos, K.-H.; Thomala, W.: Schraubenverbindungen. Grundlagen, Berechnung, Eigenschaften, Handhabung. Springer-Verlag Berlin Heidelberg, Dordrecht, 2007.

[WV08] Witsch, D.; Vogel-Heuser, B.: Modellierungsansatz für Zeitanforderungen und Kommunikationsnetze. In atp, 2008; S. 44–52.

[Zgo07] Zgorzelski, P.: Vorteile durch Nutzung der NE 100 Version 3.0. In PROLIST atp Sonderausgabe 2007, 2007; S. 10–16.

[Zgo12] Zgorzelski, P.: Vortrag: Merkmalleisten für Verfahrungstechnik im PROLIST-Workflow. PROLIST open 2012, Karlsruhe, 15.11.2012.

# Glossar

**Außenzusammenhang**

(engl.: *extra relation*) Beziehung zwischen Merkmalen verschiedener Merkmalträger

**Beziehung**

(engl.: *relationship*) Zuordnung zwischen (zwei oder mehr) Betrachtungsgegenständen (Objekt, Konzept, Person), die eine Bedeutung hat

**Binnenzusammenhang**

(engl.: *intra relation*) Beziehung zwischen Merkmalen eines Merkmalträgers

**Eigenschaft**

(engl.: *attribute*) beschreibt eine wahrnehmbare Qualität eines Betrachtungsgegenstands (Objekt, Konzept, Person etc.)

**Elementare Systemmerkmale**

Merkmale des Systems, welche bereits als Merkmale der Systemelemente ausgeprägt sind und deren Ausprägung im System sich aus den Elementmerkmalen und der Systemstruktur herleiten lassen.

**Elementbeziehung**

(engl.: *element relation*) Beziehung zwischen Elementen eines Systems

**Emergente Systemmerkmale**

Merkmale des Systems, welche nicht bei den Systemelementen ausgeprägt sind und deren Ausprägung im System auf eine spezifische Systemstruktur zurückzuführen ist.

**Funktion**

lösungsneutral beschriebene Beziehung zwischen Eingangsgrößen und Ausgangsgrößen eines Systems.

**funktionale Verbindungen**

Verbindungen, die einen funktionalen Zusammenhang zwischen Systemelementen herstellen.

**funktionaler Pfad**

Kette von den Eingangsgrößen einer Funktion über die beteiligten Elemente und funktionalen Verbindungen bis hin zu den Ausgangsgrößen der Funktion

**funktionale Struktur**

beschreibt wie die Elemente eines Systems durch funktionale Verbindungen verbunden sind.

**Konzept**

beschreibt eine abstrahierte Repräsentation einer wahrgenommenen Realität

**kompositionelle Struktur**

beschreibt wie die Elemente eines Systems durch Bestandsbeziehungen (besteht aus) verbunden sind.

**Merkmal**

(engl.: *characteristic*) eine allgemein erkennbare Eigenschaft eines Betrachtungsgegenstandes, die zu einer Klassifizierung des Betrachtungsgegenstandes genutzt werden kann.

**Merkmalträger**

(engl.: *characterized entity*) beschreibt einen Betrachtungsgegenstand als Zuordnungspunkt für Merkmale

**Modell**

externalisiertes Konzept, die vereinfachte (abstrahierte) Abbildung einer Realität

**Property**

(engl.: *property*) datentechnische Umsetzung für ein Merkmal

**Prozess**

beschreibt die Ausführung von einer oder mehreren Funktionen eines Systems mit zeitlichem und räumlichem Bezug, mit Anfangs- und Endzustand der Systemumgebung

**Rolle**

beschreibt das Muster (Pattern) für den Teilnehmer einer Beziehung. Dabei kann eine Beziehung unterschiedliche Rollen für die unterschiedlichen Endpunkte der Beziehung definieren. Ein solches Muster definiert Erwartungen in Bezug auf die Eigenschaften und das Verhalten des Teilnehmers.

**Schnittstelle**

(engl.: *interface*) Punkt der Kopplung von einem Systemelement zu einem anderen Systemelement. Es wird charakterisiert durch die ihm zugeordneten Signale und ihren Signalcharakteristiken.

**System**

(engl.: *system*) eine Menge von Elementen, welche durch Relationen miteinander verknüpft sind. Ein System hat eine Systemgrenze, die es von seiner Umwelt trennt, über die hinweg das System jedoch mit seiner Umwelt interagieren kann.

**Verbindung**

(engl.: *association*) eine Beziehung, die sich in der modellierten Realität erkennbar ausprägt, entweder dadurch dass die verbundenen Systemelemente miteinander interagieren oder dass eine Bestandsbeziehung existiert

**Verhältnis**

eine Beziehung, die sich in der modellierten Realität nur indirekt erkennbar ausprägt.

# Index


## A

Abhängigkeiten 9

Aktivitätsdiagramm 4

Anlagenlebenszyklus 42

AutomationML 10

## B

Bestandsbeziehungen 9

block of properties 33

## C

CAEX 45

## D

data element type 27

## E

eCl@ss 32

Elementbeziehungen 7, 9, 7

eOTD 38

## F

FAVA 53

Formalisierte Prozessbeschreibung 2

Funktion 11

Funktionale Struktur 6

Funktionale Verbindungen 9

funktionaler Pfad 11

funktionales Modell 2

## I

Identification Guide 38

Identifikator 25

IEC CDD 27

IEC Common Data Dictionary 27

IEC Standards

- IEC 61360 27
- IEC 61987 33
- IEC 61987-10 35
- IEC 62507-1 25
- IEC 62683 33

ISO Standards

- ISO 13584-42 27
- ISO 15926-1 42
- ISO 22745 37
- ISO 290002-5 26
- ISO/IEC 11179 26

item class 27

## K

Komponentenmodell 5

Kompositionelle Struktur 6

Kopplung 15

## L

list of properties 33

LOP 33, 35

- DLOP 37
- OLOP 37

## M

Merkmal 1

Merkmalsleisten 30

Merkmalträger 2

## N

NE100 30

Netzwerkmodell 6

## P

PROLIST 30

property 27

Prozess 13

## R

Refinement 45

Rolle 15

## S

Sachmerkmal 29

Sachmerkmallisten 29

Schnittstelle 20

soziotechnisches System 41

System 5

## T

Topologie-Modell 6

## U

UML 7, 4

## V

Verbindungen 8

Verhältnis 8

# Anhang A **Beispiele**

## A.1 Beispiel für DIN 4000

### A.1.1 Schraube

Tabelle A.1 stellt die Sachmerkmalleiste für flach aufliegende Schrauben mit Außenantrieb dar. Diese Klasse wird mit der ID = DIN-AAA511 identifiziert.

**Tabelle A.1 – Sachmerkmalleiste für eine Schraube**

| Merkmal-Kennung | Bevorzugte Benennung | Einheit | Identifikator |
|---|---|---|---|
| BLD | Bildkennung | — | DIN-AAA058 |
| A01 | Formbuchstabe Schaftform | — | DIN-AAA062 |
| A02 | Gewindekennbuchstabe | — | DIN-AAA271 |
| A03 | Gewindenenndurchmesser | mm | DIN-AAA272 |
| A04 | Gewindesteigung | mm | DIN-AAA273 |
| A08 | Toleranzklasse Gewinde | — | DIN-AAA064 |
| A09 | Gewindezusatz | — | DIN-AAA063 |
| A10 | Gewindenenngröße | — | DIN-AAA265 |
| B | Schrauben-/Nennlänge | mm | DIN-AAA066 |
| C1 | Gewindelänge | mm | DIN-AAA270 |
| D21 | Antriebsart | — | DIN-AAA269 |
| D11 | Schlüsselweite | mm | DIN-AAA060 |
| D22 | Antriebsgröße | — | DIN-AAA069 |
| E12 | Flanschdurchmesser | mm | DIN-AAA071 |
| F1 | Kopfhöhe | mm | DIN-AAA072 |
| G1 | Formbuchstabe Gewindeende | — | DIN-AAA061 |
| G21 | Bestellzusatz Kombischraube/-mutter | — | DIN-AAA077 |
| C12[a] | Scheibendicke, Ringdicke | mm | DIN-AAA266 |
| B1[a] | Scheibenaußendurchmesser | mm | DIN-AAA267 |
| G24 | Formbuchstabe Unterlegelement | — | DIN-AAA767 |
| G3 | Bestellzusatz Gewindebeschichtung | — | DIN-AAA065 |
| G4 | Produktklasse/Ausführung | — | DIN-AAA078 |
| H1 | Festigkeitsklasse/Härteklasse | — | DIN-AAA902 |
| H2 | Werkstoff | — | DIN-AAA903 |
| J | Oberflächenschutz | — | DIN-AAA079 |
| [a] Für unverlierbare Unterlegelemente. | | | |

## A.2 Beispiele für eCl@ss-Beschreibungen

### A.2.1 Schraube

Die folgende Tabelle stellt die Merkmale einer flach aufliegend Schraube mit Sechskantkopf (Klasse 23-11-01-01) entsprechend der Basic-Version von eCl@ss dar. Die hier dargestellte Tabelle ist in Bezug auf Wertelisten gekürzt und wird nur als Beispiel dargestellt.

**Tabelle A.2 – eCl@ss Merkmale für eine Schraube**

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-AAQ326#001 | zusätzlicher Online-Verweis | - | STRING | - | Angabe einer Web-Adresse, die zusätzliche Informationen zum Produkt oder Kontaktdaten enthält | - |
| 0173-1#02-AAD931#004 | Zolltarifnummer (TARIC) | TARIC | INTEGER_COUNT | - | EG-einheitliche Warennomenklatur gemäß Verordnung (EWG) Nr. 2658/87 des Rates vom 23. Juli 1987 über die zolltarifliche und statistische Nomenklatur sowie über den Gemeinsamen Zolltarif | - |
| 0173-1#02-AAO663#001 | GTIN | GTIN | STRING | - | international abgestimmte, einheitliche und weltweit überschneidungsfreie Artikelnummer für Produkte und Dienstleistungen (Global Trade Item Number,ehemals EAN) | - |
| 0173-1#02-AAO677#001 | Hersteller-Name | - | STRING | - | Bezeichnung für eine natürliche oder juristische Person, die für die Auslegung, Herstellung und Verpackung sowie die Etikettierung eines Produkts im Hinblick auf das 'Inverkehrbringen' im eigenen Namen verantwortlich ist | - |
| 0173-1#02-AAO676#002 | Herstellerartikelnummer | MAN_PROD_NUM | STRING | - | eindeutiger Produktschlüssel des Herstellers | - |
| 0173-1#02-AAP805#002 | Artikelbezeichnung | - | STRING | - | vom Hersteller oder Lieferanten vergebener Name des Produkts | - |
| 0173-1#02-AAO847#002 | Produkttypbezeichnung | - | STRING | - | Name der Produktfamilie oder Variante des Produktes | - |
| 0173-1#02-AAO735#002 | Lieferantenname | - | STRING | - | Name des Lieferanten, welcher dem Kunden einen Artikel (Produkt oder Dienstleistung) bereitstellt | - |

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-AAO736#002 | Lieferantenartikelnummer | SUP_PROD_NUM | STRING | - | eindeutiger Bestellschlüssel des Lieferanten | - |
| 0173-1#02-BAA898#004 | Antriebsgröße | - | STRING_TRANSLATABLE | - | Dimensionslose Angabe für die Größe eines Antriebes, ohne die Kurzbezeichnung der Antriebsart | - |
| 0173-1#02-BAA916#003 | Kopfdurchmesser der Schraube | - | REAL_MEASURE | mm | Äußeres Hüllmaß des Schraubenkopfes quer zur Symmetrieachse | - |
| 0173-1#02-BAA917#006 | Kopfform | - | STRING | - | Geometrische Gestaltung des oberen Teiles von Verbindungselementen und anderen Elementen | 0173-1#07-AAQ430#001 - Kugelzylinder<br>0173-1#07-BAC462#001 - Hutmutter<br>...[14] |
| 0173-1#02-AAO723#002 | Kopfform (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen die zitierte Kopfform definiert ist | - |
| 0173-1#02-BAB162#005 | Kopfhöhe (gesamt) | - | REAL_MEASURE | mm | Abstand von der Werkstückoberkante bis zum Ende des Kopfes, gemessen längs der Rotations-Symmetrieachse | - |
| 0173-1#02-AAO868#002 | Schlüsselweite | - | STRING | - | Abstand von der Werkstückoberkante bis zum Ende des Kopfes, gemessen längs der Rotations-Symmetrieachse | 0173-1#07-AAR123#001 - SW 17x19<br>0173-1#07-AAR124#001 - SW 19x19" |
| 0173-1#02-BAA922#003 | Länge der Gewindebeschichtung | - | REAL_MEASURE | mm | Ausdehnung der Gewindebeschichtung längs der Rotations-Symmetrieachse | - |
| 0173-1#02-BAB664#010 | Werkstoff | - | STRING | - | Materialzusammensetzung, aus der ein einzelnes Bauteil hergestellt ist, als Ergebnis eines Herstellungsprozesses, in dem der/die Rohstoff(e) durch Extrusion, Verformung, Schweißen usw. in die endgültige Form gebracht werden | 0173-1#07-AAA005#001 - 10CrMo910<br>0173-1#07-AAA004#001 - 13CrMo44<br>...[14] |
| 0173-1#02-AAP013#002 | Werkstoff (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen die Kennung (Codierung) für den Werkstoff definiert ist | - |

[14] Die hier dargestellte Liste der Werte ist unvollständig, um Platz zu sparen. Für eine vollständige Liste der Werte sei auf die Definition in eCl@ss verwiesen.

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-AAP915#002 | Bestellzusatz (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen der Bestellzusatz für das verwendete Bauteil definiert ist | - |
| 0173-1#02-AAP916#001 | Bestellzusatz kodiert | - | STRING | - | Zusatzangabe(n) als Kennbuchstabe(n) nach einer Norm, in welcher der erweiterte Funktion-/Lieferumfang festgelegt ist | - |
| 0173-1#02-BAA918#003 | Lage der Gewindebeschichtung | - | REAL_MEASURE | mm | Anordnung und Lage einer Beschichtung des Gewindes, Maßangaben gemäß DIN. | - |
| 0173-1#02-BAB637#003 | Produktklasse | - | STRING_TRANSLATABLE | - | Klasse von Bauteilen, die vorgegebene technische Forderungen erfüllen, so dass Abweichungen dieser Bauteile innerhalb festgelegter Grenzen bleiben. Eine Produktklasse wird üblicherweise durch eine Zahl oder durch ein Symbol bezeichnet. Diese werden durch Vereinbarung festgelegt und Produktklassenzeichen genannt | - |
| 0173-1#02-AAO842#002 | Produktklasse (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen das verwendete Kürzel (Zusatz) für die Produktklasse definiert ist | - |
| 0173-1#02-BAB010#005 | Ausgabe der Maßnorm | - | STRING | - | Jahreszahl und Monat als Datumsangabe für die Veröffentlichung der Information | 0173-1#07-AAA255#003 - DIN 223-B (EN 22568)<br>0173-1#07-AAA254#003 - DIN 345<br>...[15] |
| 0173-1#02-BAE162#002 | Anforderung gemäß | - | STRING_TRANSLATABLE | - | Hinweis auf eine bestimmte Verwendung eines Produktes, Gebrauch | - |
| 0173-1#02-BAA919#003 | Schraubenlänge | - | REAL_MEASURE | mm | Maß für die effektiv nutzbare Länge der Schraube, gemessen von der Kopfauflage (Flachkopf) bzw. von der Werkstückoberkante (Senkkopf) bis zum Schraubenende | - |
| 0173-1#02-AAO832#002 | Normnummer (Formbuchstabe) | - | STRING | - | Kennzeichen zur Identifikation unterschiedlicher Produkte, die innerhalb der selben Norm dargestellt sind | - |
| 0173-1#02-BAB101#006 | Oberflächenschutz | - | STRING | - | Art der Maßnahmen zum Erreichen eines Korrosionsschutzes und bei Bedarf eines werbenden Aussehens | - |

[15] Die hier dargestellte Liste der Werte ist unvollständig, um Platz zu sparen. Für eine vollständige Liste der Werte sei auf die Definition in eCl@ss verwiesen.

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-AAO836#002 | Oberflächenschutz (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen die Kennung für den verwendeten Oberflächenschutz definiert ist | - |
| 0173-1#02-AAO654#002 | Gewindeausführung als Kennbuchstabe (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf die Norm, in welcher der Gewindekennbuchstabe als Kennbuchstabe definiert ist | - |
| 0173-1#02-BAA907#003 | Gewindenenndurchmesser | - | REAL_MEASURE | mm | Durchmesser, der die Größe des Gewindes kennzeichnet und zur Bezeichnung des Gewindes verwendet wird | - |
| 0173-1#02-BAA908#007 | Gewinderichtung | - | STRING | - | Festgelegte Bezeichnung RH für Rechtsgewinde bzw. LH für Linksgewinde | 0173-1#07-CAA162#003 – links<br>0173-1#07-CAA161#002 - rechts" |
| 0173-1#02-BAA997#003 | Gewindelänge | - | REAL_MEASURE | mm | mit einem Gewinde versehene (und zur Kraftübertragung nutzbare) Länge eines Verbindungselementes (ohne die Länge des Gewindeauslaufs) | - |
| 0173-1#02-BAA909#005 | Gewindesteigung am Mutterende | - | REAL_MEASURE | mm | Translatorische Strecke, die eine Schraube in einem feststehenden Gegengewinde je Umdrehung (360°) zurücklegt | - |
| 0173-1#02-AAO658#003 | Gewindegröße | - | STRING | - | Partielle Größenbezeichnung für Verschraubungen unabhängig von den Einheitensystemen, die zur Kennzeichnung von Gewindeanschlüssen verwendet werden | 0173-1#07-AAC528#001 – 3 Zoll<br>0173-1#07-AAC809#001 - 5,8 Zoll<br>...[16] |
| 0173-1#02-BAB072#005 | Toleranz | - | REAL_MEASURE | % | Zulässige (symmetrische) Abweichungen des Istwertes vom Nennwert. Das Verhältnis der Abweichung zum Nennwert wird in Prozent angegeben | - |
| 0173-1#02-AAO933#002 | Toleranzangabe (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen die Kennung für die verwendete Toleranz definiert ist | - |

[16] Die hier dargestellte Liste der Werte ist unvollständig, um Platz zu sparen. Für eine vollständige Liste der Werte sei auf die Definition in eCl@ss verwiesen.

### A.2.2 Mutter

Die folgende Tabelle stellt die Merkmale einer Sechskantmutter (Klasse 23-11-07-01) entsprechend der Basic-Version von eCl@ss dar. Die hier dargestellte Tabelle ist in Bezug auf Wertelisten gekürzt und wird nur als Beispiel dargestellt.

**Tabelle A.3 – eCl@ss Merkmale für eine Mutter**

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-AAQ326#001 | zusätzlicher Online-Verweis | - | STRING | - | Angabe einer Web-Adresse, die zusätzliche Informationen zum Produkt oder Kontaktdaten enthält | - |
| 0173-1#02-AAD931#004 | Zolltarifnummer (TARIC) | TARIC | INTEGER_COUNT | - | EG-einheitliche Warennomenklatur gemäß Verordnung (EWG) Nr. 2658/87 des Rates vom 23. Juli 1987 über die zolltarifliche und statistische Nomenklatur sowie über den Gemeinsamen Zolltarif | - |
| 0173-1#02-AAO663#001 | GTIN | GTIN | STRING | - | nternational abgestimmte, einheitliche und weltweit überschneidungsfreie Artikelnummer für Produkte und Dienstleistungen (Global Trade Item Number,ehemals EAN) | - |
| 0173-1#02-AAO677#001 | Hersteller-Name | - | STRING | - | Bezeichnung für eine natürliche oder juristische Person, die für die Auslegung, Herstellung und Verpackung sowie die Etikettierung eines Produkts im Hinblick auf das 'Inverkehrbringen' im eigenen Namen verantwortlich ist | - |
| 0173-1#02-AAO676#002 | Herstellerartikelnummer | MAN_PROD_NUM | STRING | - | eindeutiger Produktschlüssel des Herstellers | - |
| 0173-1#02-AAP805#002 | Artikelbezeichnung | - | STRING | - | vom Hersteller oder Lieferanten vergebener Name des Produkts | - |
| 0173-1#02-AAO847#002 | Produkttypbezeichnung | - | STRING | - | Name der Produktfamilie oder Variante des Produktes | - |
| 0173-1#02-AAO735#002 | Lieferantenname | - | STRING | - | Name des Lieferanten, welcher dem Kunden einen Artikel (Produkt oder Dienstleistung) bereitstellt | - |
| 0173-1#02-AAO736#002 | Lieferantenartikelnummer | SUP_PROD_NUM | STRING | - | eindeutiger Bestellschlüssel des Lieferanten | - |

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-AAA103#006 | Schlüsselweite | - | INTEGER_MEASURE | mm | Größter Abstand zweier paralleler gegenüberliegender Flächen eines n-kantes | |
| 0173-1#02-AAO868#002 | Schlüsselweite | - | STRING | | Schlüsselweiten (Eckenmaße) sind gemäß DIN ISO 272 genormt. Der Abstand paralleler Flächen an Verbindungselementen, an denen das Bedienwerkzeug angreifen (z. B. Sechskantmutter bzw. -schraubenkopf), übergreifen (z. B. Ringschlüssel) oder eingreifen (z. B. Schraube mit Innensechskant) kann. | 0173-1#07-AAR123#001 - SW 17x19<br>0173-1#07-AAR124#001 - SW 19x19 |
| 0173-1#02-BAB664#010 | Werkstoff | - | STRING | - | Materialzusammensetzung, aus der ein einzelnes Bauteil hergestellt ist, als Ergebnis eines Herstellungsprozesses, in dem der/die Rohstoff(e) durch Extrusion, Verformung, Schweißen usw. in die endgültige Form gebracht werden | 0173-1#07-AAA005#001 - 10CrMo910<br>0173-1#07-AAA004#001 - 13CrMo44<br>… |
| 0173-1#02-AAP013#002 | Werkstoff (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen die Kennung (Codierung) für den Werkstoff definiert ist | - |
| 0173-1#02-AAP915#002 | Bestellzusatz (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen der Bestellzusatz für das verwendete Bauteil definiert ist | - |
| 0173-1#02-BAB376#005 | Außendurchmesser | | | | | |
| 0173-1#02-BAB637#003 | Produktklasse | - | STRING_TRANSLATABLE | - | Klasse von Bauteilen, die vorgegebene technische Forderungen erfüllen, so dass Abweichungen dieser Bauteile innerhalb festgelegter Grenzen bleiben. Eine Produktklasse wird üblicherweise durch eine Zahl oder durch ein Symbol bezeichnet. Diese werden durch Vereinbarung festgelegt und Produktklassenzeichen genannt | - |
| 0173-1#02-AAO842#002 | Produktklasse (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen das verwendete Kürzel (Zusatz) für die Produktklasse definiert ist | - |
| 0173-1#02-BAB010#005 | Ausgabe der Maßnorm | - | STRING | - | Jahreszahl und Monat als Datumsangabe für die Veröffentlichung der Information | 0173-1#07-AAA255#003 - DIN 223-B (EN 22568)<br>0173-1#07-AAA254#003 - DIN 345<br>… |

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-BAE162#002 | Anforderung gemäß | - | STRING_TRANSLATABLE | - | Hinweis auf eine bestimmte Verwendung eines Produktes, Gebrauch | - |
| 0173-1#02-AAO727#001 | Kurzzeichen | | | | | |
| 0173-1#02-AAO832#002 | Normnummer (Formbuchstabe) | - | STRING | - | Kennzeichen zur Identifikation unterschiedlicher Produkte, die innerhalb der selben Norm dargestellt sind | - |
| 0173-1#02-BAB101#006 | Oberflächenschutz | - | STRING | - | Art der Maßnahmen zum Erreichen eines Korrosionsschutzes und bei Bedarf eines werbenden Aussehens | - |
| 0173-1#02-AAO836#002 | Oberflächenschutz (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen die Kennung für den verwendeten Oberflächenschutz definiert ist | - |
| 0173-1#02-BAA911#004 | Kegelwinkel | - | REAL_MEASURE | ° | eingeschlossener Winkel zwischen den Aussenkanten eines Kegels, gemessen in der Axial-Ebene | |
| 0173-1#02-AAJ413#001 | Gewinde (metrisch) | - | STRING_TRANSLATABLE | - | Angabe zur profilierten Einkerbung, die fortlaufend wendelartig um eine zylinderförmige Wandung in einer gewundenen Schraubenlinie verläuft | |
| 0173-1#02-AAO654#002 | Gewindeausführung als Kennbuchstabe (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf die Norm, in welcher der Gewindekennbuchstabe als Kennbuchstabe definiert ist | - |
| 0173-1#02-BAA907#003 | Gewindenenndurchmesser | - | REAL_MEASURE | mm | Durchmesser, der die Größe des Gewindes kennzeichnet und zur Bezeichnung des Gewindes verwendet wird | - |
| 0173-1#02-BAA908#007 | Gewinderichtung | - | STRING | - | Festgelegte Bezeichnung RH für Rechtsgewinde bzw. LH für Linksgewinde | 0173-1#07-CAA162#003 – links<br>0173-1#07-CAA161#002 - rechts |
| 0173-1#02-BAA997#003 | Gewindelänge | - | REAL_MEASURE | mm | mit einem Gewinde versehene (und zur Kraftübertragung nutzbare) Länge eines Verbindungselementes (ohne die Länge des Gewindeauslaufs) | - |
| 0173-1#02-BAA909#005 | Gewindesteigung am Mutterende | - | REAL_MEASURE | mm | Translatorische Strecke, die eine Schraube in einem feststehenden Gegengewinde je Umdrehung (360°) zurücklegt | - |

| ID | Label | Abk. | Datentyp | Einheit | Definition | Werte |
|---|---|---|---|---|---|---|
| 0173-1#02-AAO658#003 | Gewindegröße | - | STRING | - | Partielle Größenbezeichnung für Verschraubungen unabhängig von den Einheitensystemen, die zur Kennzeichnung von Gewindeanschlüssen verwendet werden | 0173-1#07-AAC528#001 – 3 Zoll<br>0173-1#07-AAC809#001 - 5,8 Zoll<br>… |
| 0173-1#02-AAK304#003 | Gewindegröße (imperial) | - | STRING | - | gibt das Maß des Gewindes in Zoll an | 0173-1#07-AAB974#001 - 1 1/2 Zoll<br>0173-1#07-AAB975#001 - 1 1/4 Zoll<br>… |
| 0173-1#02-BAB072#005 | Toleranz | - | REAL_MEASURE | % | Zulässige (symmetrische) Abweichungen des Istwertes vom Nennwert. Das Verhältnis der Abweichung zum Nennwert wird in Prozent angegeben | - |
| 0173-1#02-AAO933#002 | Toleranzangabe (in Anlehnung an Norm) | - | STRING | - | Bezugnahme auf eine oder mehrere Normen und Richtlinien, in denen die Kennung für die verwendete Toleranz definiert ist | - |
| 0173-1#02-BAA927#004 | gesamte Höhe der Mutter | | REAL_MEASURE | mm | Abstand zwischen Auflagefläche und Oberkante, gemessen parallel zum Gewinde | - |
| 0173-1#02-BAB180#005 | Gewindeart | - | STRING | - | Ausführung und Konstruktion eines Gewindes | 0173-1#07-AAM592#001 – Regelgewinde<br>0173-1#07-AAP621#001 - API |
| 0173-1#02-BAD656#007 | halogenfrei | - | BOOLEAN | - | Angabe, ob das Gerät oder das Bauteil Halogenfrei konstruiert ist | 0173-1#07-CAA017#003 – Nein<br>0173-1#07-CAA016#001 - Ja |

## A.3 CAEX++- Beschreibung der Bauelemente

### A.3.1 CAEX++ Beschreibung der Schraube

Im Folgenden ist die CAEX++ Beschreibung der Schraube basierend auf der eClass-Definition (siehe A.2.1) aufgelistet.

Teile der Definition (z.B. Description) wurden nicht in die Beschreibung importiert, weil sie bei Bedarf über die semantische Referenz (<RefSemantic>-Element) ermittelt werden können.

Die relevanten Attribute für die Herstellung der Schraubverbindung wurden dem entsprechenden Interface zugewiesen. Zu beachten ist das zusätzliche Attribut „GewindeArt“, das hinzugefügt wurde um die Kompatibilität mit der Mutter vollständig prüfen zu können.

```
<?xml version="1.0" encoding="UTF-8"?>
<CAEXFile SchemaVersion="3.0" FileName="" xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xsi:noNamespaceSchemaLocation="file:///CAEXpp_ClassModel_V1.00.xsd">
    <ExternalReference Path="http://www.eclasscontent.com/index.php&#63;action&#61;cc2prdet&amp;language&#61;de&am
p;version&#61;8.1&amp;pridatt&#61;" Alias="eclass"/>
    <SystemUnitClassLib Name="Test">
        <SystemUnitClass Name="Schraube">
            <Attribute Name="zusaetzlicherOnline-Verweis" ID="EAFEC516-5DA2-44F5-AA5B-4FE360D532E3"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAQ326#001"/>
            </Attribute>
            <Attribute Name="Zolltarifnummer" ID="10DBC08F-B51C-49DB-B98A-C1E39C0879A8"
                AttributeDataType="INTEGER_COUNT" Unit="TEST">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAD931#004"/>
            </Attribute>
            <Attribute Name="GTIN" ID="EC7125B4-359C-4BA3-AF39-2BD993BAC325" AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO663#001"/>
            </Attribute>
            <Attribute Name="Hersteller-Name" ID="420FD5B1-7E9D-4B51-93EF-B0BDBD1CDCCB"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO677#001"/>
            </Attribute>
            <Attribute Name="Herstellerartikelnummer" ID="CDAC41DE-D5AA-4005-A04B-F9E0E08D5982"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO676#002"/>
            </Attribute>
            <Attribute Name="Artikelbezeichnung" ID="A7E66B7F-12B3-4D34-9D93-78CAEBD41097"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAP805#002"/>
            </Attribute>
            <Attribute Name="Produkttypbezeichnung" ID="A39DEA3A-D657-4BBA-AF6A-351ECA3E3630"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO847#002"/>
            </Attribute>
            <Attribute Name="Lieferantenname" ID="2DEDF051-E3E1-4E69-BC84-CE2365D240CD"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO735#002"/>
            </Attribute>
            <Attribute Name="Lieferantenartikelnummer" ID="F8CF058F-627C-4A20-B789-37575372E1F0"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO736#002"/>
            </Attribute>
            <Attribute Name="Antriebsgroesse" ID="5A867AB4-8CB4-4B6E-A37E-127F81B5CCF3"
                AttributeDataType="STRING_TRANSLATABLE">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA898#004"/>
            </Attribute>
            <Attribute Name="KopfdurchmesserderSchraube" ID="A8EEA5DF-695F-4498-B19B-F91BD7608DE3"
```

```
        AttributeDataType="REAL_MEASURE" Unit="mm">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA916#003"/>
    </Attribute>
    <Attribute Name="Kopfform" ID="631BD0AE-BDA0-4BA3-B926-A7640F82C08E" AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA917#006"/>
    </Attribute>
    <Attribute Name="Kopfform(inAnlehnunganNorm)" ID="61621ABA-A6DD-4821-AA16-A23E3ABCA64F"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO723#002"/>
    </Attribute>
    <Attribute Name="Kopfhoehe" ID="BB4570DB-4E62-48EA-8A89-B5FE28C68CCF"
        AttributeDataType="REAL_MEASURE" Unit="mm">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB162#005"/>
    </Attribute>
    <Attribute Name="Schluesselweite" ID="64738B3F-2CC6-429A-9236-99DD0098B03A"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO868#002"/>
    </Attribute>
    <Attribute Name="Werkstoff" ID="73A637F8-11C9-437C-B282-AA05D033A24C" AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB664#010"/>
    </Attribute>
    <Attribute Name="Werkstoff(inAnlehnunganNorm)" ID="BD8AC2A1-DD85-44BC-86B1-66D528E3E818"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAP013#002"/>
    </Attribute>
    <Attribute Name="Bestellzusatz(inAnlehnunganNorm)" ID="B5E81EC9-B353-4478-8CBF-2E7635253AD2"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAP915#002"/>
    </Attribute>
    <Attribute Name="Bestellzusatzkodiert" ID="08FD6120-EA1C-4682-8FFA-CE1FAD326D75"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAP916#001"/>
    </Attribute>
    <Attribute Name="Produktklasse" ID="CD88EF7F-8AC3-4FAB-AD88-D93E63C25FFD"
        AttributeDataType="STRING_TRANSLATABLE">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB637#003"/>
    </Attribute>
    <Attribute Name="Produktklasse(inAnlehnunganNorm)" ID="7224F843-35AE-45CB-A110-26E14BBA0B9D"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO842#002"/>
    </Attribute>
    <Attribute Name="AusgabederMassnorm" ID="45A5D9FF-918B-4C17-AE19-2C16532D8F2C"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB010#005"/>
    </Attribute>
    <Attribute Name="Anforderunggemaess" ID="BF93E235-816D-47F9-BAB9-7D7443073AC3"
        AttributeDataType="STRING_TRANSLATABLE">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAE162#002"/>
    </Attribute>
    <Attribute Name="Schraubenlaenge" ID="03AF9972-9484-4236-8EB6-A960495395CE"
        AttributeDataType="REAL_MEASURE" Unit="mm">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA919#003"/>
    </Attribute>
    <Attribute Name="Normnummer" ID="72257B8A-B790-4BFE-BED4-9E331A2F2DDB"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO832#002"/>
    </Attribute>
    <Attribute Name="Oberflaechenschutz" ID="EC37FE7E-F53E-4307-B57E-B3E7564381EE"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB101#006"/>
    </Attribute>
    <Attribute Name="Oberflaechenschutz(inAnlehnunganNorm)" ID="D54F511D-6330-4E51-917D-32CEAB75E536"
        AttributeDataType="STRING">
        <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO836#002"/>
    </Attribute>
    <ExternalInterface Name="Außengewinde" ID="275EB6DA-D883-4988-A273-E7FA9E46B47B">
        <Attribute Name="LaengederGewindebeschichtung" ID="3EA08B84-B457-4EBA-A165-7E11F34FB3BE"
            AttributeDataType="REAL_MEASURE" Unit="mm">
            <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA922#003"/>
        </Attribute>
        <Attribute Name="LagederGewindebeschichtung" ID="3253E68C-4097-4E11-877C-96D87B91DBB9"
            AttributeDataType="REAL_MEASURE" Unit="mm">
            <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA918#003"/>
        </Attribute>
        <Attribute Name="GewindeausfuehrungalsKennbuchstabe"
            ID="2F4F490B-567E-477D-ADD0-3D68C567CC4F" AttributeDataType="STRING">
            <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO654#002"/>
```

```
                </Attribute>
                <Attribute Name="Gewindenenndurchmesser" ID="8B9FE5D9-BF7C-462C-A7D1-5B25E8C4D484"
                    AttributeDataType="REAL_MEASURE" Unit="mm">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA907#003"/>
                </Attribute>
                <Attribute Name="Gewinderichtung" ID="A14B7AF0-75A8-46AD-81D4-E9CB45D6F53F"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA908#007"/>
                </Attribute>
                <Attribute Name="Gewindelaenge" ID="E5416238-3D7A-4668-BA47-09F98B83139F"
                    AttributeDataType="REAL_MEASURE" Unit="mm">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA997#003"/>
                </Attribute>
                <Attribute Name="GewindesteigungamMutterende" ID="82DD0BEA-28F2-4379-8A33-0EB486F3D951"
                    AttributeDataType="REAL_MEASURE" Unit="mm">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA909#005"/>
                </Attribute>
                <Attribute Name="Gewindegroesse" ID="9FA1CD75-7074-4F1A-A752-A0CC32B30F09"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO658#003"/>
                </Attribute>
<!-- zusätzliches Attribute um eine vollständige Prüfung der Kompatibilität zu ermöglichen -->
                <Attribute Name="Gewindeart" ID="A4F659B4-A85B-4FAC-95E9-DD666472B119"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB180#005"/>
                </Attribute>
                <Attribute Name="Toleranz" ID="F0725111-A567-49B3-99DB-6F351E0FAF6B"
                    AttributeDataType="REAL_MEASURE" Unit="%">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB072#005"/>
                </Attribute>
                <Attribute Name="Toleranzangabe(inAnlehnunganNorm)" ID="8778FB74-ED8D-4C0C-8234-207ED489A7BA"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO933#002"/>
                </Attribute>
            </ExternalInterface>
        </SystemUnitClass>
    </SystemUnitClassLib>
</CAEXFile>
```

### A.3.2 CAEX++ Beschreibung der Mutter

Im Folgenden ist die CAEX++ Beschreibung der Mutter basierend auf der eClass-Definition (siehe A.2.2) aufgelistet.

Teile der Definition (z.B. Description) wurden nicht in die Beschreibung importiert, weil sie bei Bedarf über die semantische Referenz (<RefSemantic>-Element) ermittelt werden können.

Die relevanten Attribute für die Herstellung der Schraubverbindung wurden dem entsprechenden Interface („Innengewinde“) zugewiesen.

```
<?xml version="1.0" encoding="UTF-8"?>
<CAEXFile SchemaVersion="3.0" FileName="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///W:/Documents/Uni/Promotion/Diss/CAEX++/CAEXpp_ClassModel_V1.00.xsd">
    <ExternalReference
Path="http://www.eclasscontent.com/index.php&#63;action&#61;cc2prdet&amp;language&#61;de&amp;version&#61;8.1&amp;
pridatt&#61;" Alias="eclass"/>
    <SystemUnitClassLib Name="Test">
        <SystemUnitClass Name="Mutter">
            <Attribute Name="zusaetzlicherOnline-Verweis" ID="F459E4F5-EA7F-44A1-9703-E9B0BF0F3A93"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAQ326#001"/>
            </Attribute>
            <Attribute Name="Zolltarifnummer" ID="4ECBEBD6-DA52-45BA-B8D7-F662C095FAF8"
                AttributeDataType="INTEGER_COUNT">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAD931#004"/>
```

```
</Attribute>
<Attribute Name="GTIN" ID="3342C1A7-0209-46AC-B549-4CCD48E91FFF" AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO663#001"/>
</Attribute>
<Attribute Name="Hersteller-Name" ID="8763890F-294C-4C0A-A769-D39BD0A1B3B5"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO677#001"/>
</Attribute>
<Attribute Name="Herstellerartikelnummer" ID="63BC7A2F-5666-4DE1-8E9D-CB6AD6AB632D"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO676#002"/>
</Attribute>
<Attribute Name="Artikelbezeichnung" ID="03F84FC1-9969-4C07-94E8-DB5389065416"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAP805#002"/>
</Attribute>
<Attribute Name="Produkttypbezeichnung" ID="DD2E7070-2B6B-4ED8-A348-4819FD3B3B0C"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO847#002"/>
</Attribute>
<Attribute Name="Lieferantenname" ID="F9AEA67B-B41A-4DEA-A11F-78149C7B1D60"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO735#002"/>
</Attribute>
<Attribute Name="Lieferantenartikelnummer" ID="8B7F97A1-C959-4582-87EA-EF4BBDD15E92"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO736#002"/>
</Attribute>
<Attribute Name="Schluesselweite" ID="B8B6AAAC-4AB3-41DF-8A07-F06AD4AA0E47"
    AttributeDataType="INTEGER_MEASURE" Unit="mm">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAA103#006"/>
</Attribute>
<Attribute Name="Schluesselweite" ID="0CF9DE55-B5DE-426E-9D26-67F763974986"
    AttributeDataType="STRING" Unit="">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO868#002"/>
</Attribute>
<Attribute Name="Werkstoff" ID="72223F21-314B-43C1-B491-EE772A49ADE1" AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB664#010"/>
</Attribute>
<Attribute Name="Werkstoff(inAnlehnunganNorm)" ID="2CFE3BF3-5534-4BD9-92BA-002601625770"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAP013#002"/>
</Attribute>
<Attribute Name="Bestellzusatz(inAnlehnunganNorm)" ID="EBC4DD2D-F048-4157-BDD8-EBE6CD6D6E47"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAP915#002"/>
</Attribute>
<Attribute Name="Aussendurchmesser" ID="1ADCFB8A-055A-4807-B752-254E44F6109F"
    AttributeDataType="REAL_MEASURE">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB376#005"/>
</Attribute>
<Attribute Name="Produktklasse" ID="D8CE274A-7A3A-4784-9FCD-0B5291F9CD84"
    AttributeDataType="STRING_TRANSLATABLE">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB637#003"/>
</Attribute>
<Attribute Name="Produktklasse(inAnlehnunganNorm)" ID="5FC0C4B0-AEA5-4FAF-ABF5-BA8B09811B4B"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO842#002"/>
</Attribute>
<Attribute Name="AusgabederMassnorm" ID="B75E015F-C77A-4A22-87C3-90E321C84993"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB010#005"/>
</Attribute>
<Attribute Name="Anforderunggemaess" ID="AA6356A5-38D2-4CE0-A8BE-1BF0DF21B6AA"
    AttributeDataType="STRING_TRANSLATABLE">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAE162#002"/>
</Attribute>
<Attribute Name="Kurzzeichen" ID="176C53B3-C955-4CC7-AB19-C2D9205F071E" AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO727#001"/>
</Attribute>
<Attribute Name="Normnummer" ID="90BD6B4C-5566-4004-9691-3D15309C32D9"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO832#002"/>
</Attribute>
<Attribute Name="Oberflaechenschutz" ID="2CFB565E-36CD-4E8C-A258-97FC9A7FC8CB"
    AttributeDataType="STRING">
    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB101#006"/>
```

```
            </Attribute>
            <Attribute Name="Oberflaechenschutz_genormt" ID="FA3687AE-8FC8-40B9-A641-65F230B1E10E"
                AttributeDataType="STRING">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO836#002"/>
            </Attribute>
            <Attribute Name="Kegelwinkel" ID="6CF3D46D-B798-49BC-B8BD-8751839DC743"
                AttributeDataType="REAL_MEASURE" Unit=" °">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA911#004"/>
            </Attribute>
            <Attribute Name="Gewinde" ID="FE6C3927-6B06-4813-8E9A-BD77D1F4655C"
                AttributeDataType="STRING_TRANSLATABLE">
                <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAJ413#001"/>
            </Attribute>
            <ExternalInterface Name="Innengewinde" ID="E5A3ABA4-4C96-47F9-B118-608141B1D7B9">
                <Attribute Name="GewindeausfuehrungalsKennbuchstabe(inAnlehnunganNorm)"
                    ID="831F8AE7-58DE-4C79-B658-6E46938B1103" AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO654#002"/>
                </Attribute>
                <Attribute Name="Gewindenenndurchmesser" ID="237AC48E-345B-4275-97EA-4AA8D3AD6E9E"
                    AttributeDataType="REAL_MEASURE" Unit="mm">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA907#003"/>
                </Attribute>
                <Attribute Name="Gewinderichtung" ID="D81648CD-724A-41FC-AB19-4F2B60D1EE87"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA908#007"/>
                </Attribute>
                <Attribute Name="Gewindelaenge" ID="8136B0CB-7B50-4AC2-8CB0-3DA52D02EF03"
                    AttributeDataType="REAL_MEASURE" Unit="mm">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA997#003"/>
                </Attribute>
                <Attribute Name="GewindesteigungamMutterende" ID="E640A7B3-CE05-43F2-804B-888A7D9760DD"
                    AttributeDataType="REAL_MEASURE" Unit="mm">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA909#005"/>
                </Attribute>
                <Attribute Name="Gewindegroesse" ID="1C8BBE9F-E060-44F3-BE09-2C3977FED279"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO658#003"/>
                </Attribute>
                <Attribute Name="Gewindeart" ID="A9362F1D-85AF-4F0E-B324-A0F5F80B8FEB"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB180#005"/>
                </Attribute>
                <Attribute Name="Gewindegroesse" ID="0EEB15EF-786F-40CC-A001-60FB834FF97D"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAK304#003"/>
                </Attribute>
                <Attribute Name="Toleranz" ID="E8361B09-17EB-4977-8A4C-DD7DBD472A01"
                    AttributeDataType="REAL_MEASURE" Unit="%">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAB072#005"/>
                </Attribute>
                <Attribute Name="Toleranzangabe(inAnlehnunganNorm)" ID="5FB4CEE2-6CE3-4D3D-8A81-1A6058050DCB"
                    AttributeDataType="STRING">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-AAO933#002"/>
                </Attribute>
                <Attribute Name="gesamteHoehederMutter" ID="C8C58DA7-84D2-4323-BF5B-0A21D9BFE139"
                    AttributeDataType="REAL_MEASURE" Unit="mm">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAA927#004"/>
                </Attribute>
                <Attribute Name="halogenfrei" ID="CFBD6CFC-67D6-415C-BC10-62AD67B37082"
                    AttributeDataType="BOOLEAN">
                    <RefSemantic CorrespondingAttributePath="eclass@0173-1#02-BAD656#007"/>
                </Attribute>
            </ExternalInterface>
        </SystemUnitClass>
    </SystemUnitClassLib>
</CAEXFile>
```

# Anhang B **CAEX++ Schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- CAEX++ - Erweiterung des Computer Aided Engineering Data-Exchange-Metamodel -->
<!-- Version 1.00, 25.11.2014 -->
<!-- Mit XMLSpy v2014 rel. 2 sp1 (x64) (http://www.altova.com) von Thomas Hadlich (Otto-v.-Guericke Uni Magdeburg) bearbeitet -->
<!-- Based on CAEX Version 3.0, 22.11.2012 -->
<!-- Version 0.01, 26.03.2014 -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://ifatwww.et.uni-magdeburg.de/~hadlich/CAEX++/" targetNamespace="http://ifatwww.et.uni-magdeburg.de/~hadlich/CAEX++/" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- tha -->
    <xs:simpleType name="ChangeMode">
        <xs:restriction base="xs:string">
            <xs:enumeration value="state"/>
            <xs:enumeration value="create"/>
            <xs:enumeration value="delete"/>
            <xs:enumeration value="change"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:group name="Header">
        <xs:annotation>
            <xs:documentation>Defines a group of organizational information, like description, version, revision, copyright, etc.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="Description" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Textual description for CAEX objects.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="ChangeMode" type="ChangeMode" use="optional" default="state"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="Version" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Organizational information about the state of the version.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="ChangeMode" type="ChangeMode" use="optional" default="state"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="Revision" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Organizational information about the state of the revision.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="CAEXBasicObject">
                            <xs:sequence>
                                <xs:element name="RevisionDate" type="xs:dateTime"/>
                                <xs:element name="OldVersion" type="xs:string" minOccurs="0"/>
                                <xs:element name="NewVersion" type="xs:string" minOccurs="0"/>
                                <xs:element name="AuthorName" type="xs:string"/>
                                <xs:element name="Comment" type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="Copyright" minOccurs="0">
                <xs:annotation>
```

```
                    <xs:documentation>Organizational information about copyright.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="ChangeMode" type="ChangeMode" use="optional" default="state"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="AdditionalInformation" type="xs:anyType" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Optional auxiliary field that may contain any additional information about a CAEX
object.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="SourceObjectInformation" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Organizational information about the source of the corresponding CAEX object.
</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="OriginID" type="xs:string" use="required">
                                <xs:annotation>
                                    <xs:documentation>This attribute describes the ID of the origin of the belonging object, e.g.
a source engineering tool. The value is according to the vendor specific OriginID.</xs:documentation>
                                </xs:annotation>
                            </xs:attribute>
                            <xs:attribute name="SourceObjID" type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Optional attribute representing the ID of the source object in the source
data model.  </xs:documentation>
                                </xs:annotation>
                            </xs:attribute>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="CAEXBasicObject">
        <xs:annotation>
            <xs:documentation>CAEX basis object that comprises a basic set of attributes and header information which exist
for all CAEX elements.</xs:documentation>
        </xs:annotation>
        <xs:group ref="Header" minOccurs="0"/>
        <xs:attribute name="ChangeMode" type="ChangeMode" use="optional" default="state">
            <xs:annotation>
                <xs:documentation>Optionally describes the change state of a CAEX object. If used, the ChangeMode shall
have the following value range: state, create, delete and change. This information should be used for further change
management applications.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="CAEXObject">
        <xs:annotation>
            <xs:documentation>CAEX basis object derived from CAEXBasicObject, augmented by
Name (required) and ID (optional).</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXBasicObject">
                <xs:attribute name="ID" type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>Optional attribute that describes a unique identifier of the CAEX
object.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Name" type="xs:string" use="required">
                    <xs:annotation>
                        <xs:documentation>Describes the name of the CAEX object.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

```
    <xs:complexType name="InterfaceClassType">
        <xs:annotation>
            <xs:documentation>Shall be used for InterfaceClass definition, provides base structures for an interface class
definition. Tha: Additionally a reference to an equivalent class definition may be provided.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXObject">
                <xs:sequence minOccurs="0">
                    <xs:element name="Attribute" type="AttributeType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Characterizes properties of the InterfaceClass.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="ExternalInterface" type="InterfaceClassType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="RefSemanticClass" type="RefSemanticClassType" minOccurs="0"
maxOccurs="unbounded"/>
                    <!-- tha -->
                </xs:sequence>
                <xs:attribute name="RefBaseClassPath" type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>Stores the reference of a class to its base class. References contain the full path to
the refered class object.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="InterfaceFamilyType">
        <xs:annotation>
            <xs:documentation>Defines base structures for a hierarchical InterfaceClass tree. The hierarchical structure of an
interface library has organizational character only.  </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="InterfaceClassType">
                <xs:sequence minOccurs="0">
                    <xs:element name="InterfaceClass" type="InterfaceFamilyType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Element that allows definition of child InterfaceClasses within the class
hierarchy. The parent child relation between two InterfaceClasses has no semantic.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="RoleClassType">
        <xs:annotation>
            <xs:documentation>Shall be used for RoleClass definition, provides base structures for a role class
definition.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXObject">
                <xs:sequence minOccurs="0">
                    <xs:element name="Attribute" type="AttributeType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Characterizes properties of the RoleClass.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="ExternalInterface" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Description of an external interface. Tha: Additionally a reference to an
equivalent class definition may be provided.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="InterfaceClassType"/>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="RefSemanticClass" type="RefSemanticClassType" minOccurs="0"
maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Description of an equivalent link class.</xs:documentation>
                        </xs:annotation>
                        <!-- tha -->
                    </xs:element>
```

```
                </xs:sequence>
                <xs:attribute name="RefBaseClassPath" type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>Stores the reference of a class to its base class. References contain the full path to
the refered class object.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="RoleFamilyType">
        <xs:annotation>
            <xs:documentation>Defines base structures for a hierarchical RoleClass tree. The hierarchical structure of a role
library has organizational character only. </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="RoleClassType">
                <xs:sequence minOccurs="0">
                    <xs:element name="RoleClass" type="RoleFamilyType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Element that allows definition of child RoleClasses within the class hierarchy.
The parent child relation between two RoleClasses has no semantic.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SystemUnitClassType">
        <xs:annotation>
            <xs:documentation>Defines base structures for a SystemUnit class definition. Tha: The link definition was extended
and additionally a reference to an equivalent class may be provided</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXObject">
                <xs:sequence minOccurs="0">
                    <xs:element name="Attribute" type="AttributeType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Characterizes properties of the SystemUnitClass.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="ExternalInterface" type="InterfaceClassType" minOccurs="0"
maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Description of an external interface.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="InternalElement" type="InternalElementType" minOccurs="0"
maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Shall be used in order to define nested objects inside of a SystemUnitClass or
another InternalElement. Allows description of the internal structure of a CAEX object.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="SupportedRoleClass" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Allows the association to a RoleClass which this SystemUnitClass can play. A
SystemUnitClass may reference multiple roles.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXBasicObject">
                                    <xs:sequence minOccurs="0">
                                        <xs:element name="MappingObject" type="MappingType" minOccurs="0"/>
                                    </xs:sequence>
                                    <xs:attribute name="RefRoleClassPath" type="xs:string" use="required"/>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="InternalLink" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Shall be used in order to define the relationships between internal interfaces of
InternalElements. Tha: Additionally a link may be documented with attributes, reference to base class and reference to the
transported product.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
```

```
                        <xs:complexContent>
                            <xs:extension base="CAEXObject">
                                <xs:sequence>
                                    <!-- tha -->
                                    <xs:element name="Attribute" type="AttributeType" minOccurs="0"
maxOccurs="unbounded">
                                        <xs:annotation>
                                            <xs:documentation>Tha: Characterizes properties of the
InternalLink.</xs:documentation>
                                        </xs:annotation>
                                    </xs:element>
                                </xs:sequence>
                                <xs:attribute name="RefPartnerSideA" type="xs:string" use="required"/>
                                <xs:attribute name="RefPartnerSideB" type="xs:string" use="required"/>
                                <xs:attribute name="RefTransportedProduct" type="xs:string" use="optional"/>
                                <xs:attribute name="RefBaseClassPath" type="xs:string" use="optional"/>
                            </xs:extension>
                            <!-- tha -->
                            <!-- tha -->
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="RefSemanticClass" type="RefSemanticClassType" minOccurs="0"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Description of an equivalent link class.</xs:documentation>
                    </xs:annotation>
                    <!-- tha -->
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="SystemUnitFamilyType">
    <xs:annotation>
        <xs:documentation>Defines base structures for a hierarchical SystemUnitClass tree. The hierarchical structure of a
SystemUnit library has organizational character only.  </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="SystemUnitClassType">
            <xs:sequence minOccurs="0">
                <xs:element name="SystemUnitClass" type="SystemUnitFamilyType" minOccurs="0"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Element that allows definition of child SystemUnitClasses within the class
hierarchy. The parent child relation between two SystemUnitClasses has no semantic.</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="RefBaseClassPath" type="xs:string" use="optional">
                <xs:annotation>
                    <xs:documentation>Stores the reference of a class to its base class. References contain the full path to
the refered class object.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="LinkClassType">
    <xs:annotation>
        <xs:documentation>Tha: Defines base for a Link class definition.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="CAEXObject">
            <xs:sequence minOccurs="0">
                <xs:element name="Attribute" type="AttributeType" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Tha: Characterizes properties of the Link class.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="RefSemanticClass" type="RefSemanticClassType" minOccurs="0" maxOccurs="1">
                    <xs:annotation>
                        <xs:documentation>Tha: Description of an equivalent link class.</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="RefBaseClassPath" type="xs:string" use="optional">
```

```
                    <xs:annotation>
                        <xs:documentation>Stores the reference of a link type to its base class. References contain the full path
to the refered link class type.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="InterfaceSideA" type="xs:string" use="optional"/>
                <xs:attribute name="InterfaceSideB" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:complexContent>
        <!-- tha -->
    </xs:complexType>
    <xs:complexType name="LinkFamilyType">
        <xs:annotation>
            <xs:documentation>Tha: Defines base class for a Link.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="LinkClassType">
                <xs:sequence minOccurs="0">
                    <xs:element name="LinkClass" type="LinkFamilyType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Element that allows definition of child LinkClasses within the class
hierarchy.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
        <!-- tha -->
    </xs:complexType>
    <xs:complexType name="InternalElementType">
        <xs:annotation>
            <xs:documentation>Defines base structures for a hierarchical object instance. The instance maybe part of the
InstanceHierarchy or a SystemUnitClass.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="SystemUnitClassType">
                <xs:sequence minOccurs="0">
                    <xs:element name="RoleRequirements" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Describes role requirements of an InternalElement. It allows the definition of a
reference to a RoleClass and the specification of role requirements like required attributes and required interfaces.
</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXBasicObject">
                                    <xs:sequence>
                                        <xs:element name="Attribute" type="AttributeType" minOccurs="0"
maxOccurs="unbounded">
                                            <xs:annotation>
                                                <xs:documentation>Characterizes properties of the
RoleRequirements.</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="ExternalInterface" type="InterfaceClassType" minOccurs="0"
maxOccurs="unbounded"/>
                                        <xs:element name="MappingObject" type="MappingType" minOccurs="0">
                                            <xs:annotation>
                                                <xs:documentation>Host element for AttributeNameMapping and
InterfaceIDMapping.</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="RefBaseRoleClassPath" type="xs:string" use="required"/>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="RefBaseSystemUnitPath" type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>Stores the reference of an InternalElement to a class or instance definition.
References contain the full path information. </xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
```

```
    </xs:complexType>
    <xs:complexType name="ElementRelation">
        <xs:annotation>
            <xs:documentation>Tha: Shall be used to define the relationships between InternalElements of different
Models.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXObject">
                <xs:sequence>
                    <xs:element name="SupportedRoleClass" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Tha: Allows the association to definition of a Role which this ElementRelation
can play. </xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXBasicObject">
                                    <xs:sequence minOccurs="0">
                                        <xs:element name="MappingObject" type="MappingType" minOccurs="0"/>
                                    </xs:sequence>
                                    <xs:attribute name="RefRoleClassPath" type="xs:string" use="required"/>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="RefPartnerSideA" type="xs:string" use="required"/>
                <xs:attribute name="RefPartnerSideB" type="xs:string" use="required"/>
                <xs:attribute name="RelationType" type="xs:string" use="required" />
                <xs:attribute name="Direction" use="optional" default="none">
                    <xs:annotation>
                        <xs:documentation>Tha: Defines the direction of the relation.</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="to_SideA">
                                <xs:annotation>
                                    <xs:documentation>source is SideB, target is SideA</xs:documentation>
                                </xs:annotation>
                            </xs:enumeration>
                            <xs:enumeration value="to_SideB">
                                <xs:annotation>
                                    <xs:documentation>source is SideA, target is SideB</xs:documentation>
                                </xs:annotation>
                            </xs:enumeration>
                            <xs:enumeration value="none">
                                <xs:annotation>
                                    <xs:documentation>undirected or bidirectional relation</xs:documentation>
                                </xs:annotation>
                            </xs:enumeration>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
        <!-- tha -->
    </xs:complexType>
    <xs:complexType name="ElementRelationCollection">
        <xs:annotation>
            <xs:documentation>Tha: Defines the collection of ElementRelations for the respective container (CAEXFile or
InstanceHierarchy). </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXObject">
                <xs:sequence>
                    <xs:element name="ElementRelation" type="ElementRelation" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
        <!-- tha -->
    </xs:complexType>
    <xs:complexType name="RefSemanticClassType">
        <xs:annotation>
            <xs:documentation>A reference to a definition of an equivalent class, e. g. to a property-based
definition.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
```

```
            <xs:extension base="CAEXBasicObject">
                <xs:attribute name="CorrespondingClassPath" type="xs:string" use="required"/>
            </xs:extension>
        </xs:complexContent>
        <!-- tha -->
    </xs:complexType>
    <xs:complexType name="AttributeType">
        <xs:annotation>
            <xs:documentation>Defines base structures for attribute definitions. Tha: additional a formula for calculation of the attribute value may be provided</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXObject">
                <xs:sequence minOccurs="0">
                    <xs:element name="DefaultValue" type="xs:string" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>A predefined default value for an attribute.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="Value" type="xs:string" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Element describing the value of an attribute.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="RefSemantic" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>A reference to a definition of a defined attribute, e. g. to an attribute in a standardized library, this allows the semantic definition of the attribute.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXBasicObject">
                                    <xs:attribute name="CorrespondingAttributePath" type="xs:string" use="required"/>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Constraint" type="AttributeValueRequirementType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Element to restrict the range of validity of a defined attribute.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="Attribute" type="AttributeType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Element that allows the description of nested attributes.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="Formula" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Tha: Provides a formula (in MathML) to calculate the current value of the Attribute (e.g. depending on values of other attributes).</xs:documentation>
                        </xs:annotation>
                        <!-- tha -->
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="Unit" type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>Describes the unit of the attribute.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="AttributeDataType" use="optional">
                    <xs:annotation>
                        <xs:documentation>Describes the data type of the attribute using XML notation.</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="RefAttributeType" type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>Refences an attribute type in the attribute library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
```

```
    </xs:complexType>
    <xs:complexType name="AttributeFamilyType">
        <xs:annotation>
            <xs:documentation>Defines base structures for attribute type definitions.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="AttributeType">
                <xs:sequence>
                    <xs:element name="AttributeType" type="AttributeFamilyType" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="AttributeValueRequirementType">
        <xs:annotation>
            <xs:documentation>Defines base structures for definition of value requirements of an attribute.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CAEXBasicObject">
                <xs:choice>
                    <xs:element name="OrdinalScaledType">
                        <xs:annotation>
                            <xs:documentation>Element of to define constraints of ordinal scaled attribute
values.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence minOccurs="0">
                                <xs:element name="RequiredMaxValue" type="xs:string" minOccurs="0">
                                    <xs:annotation>
                                        <xs:documentation>Element to define a maximum value of an
attribute.</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="RequiredValue" type="xs:string" minOccurs="0">
                                    <xs:annotation>
                                        <xs:documentation>Element to define a required value of an attribute.
</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="RequiredMinValue" type="xs:string" minOccurs="0">
                                    <xs:annotation>
                                        <xs:documentation>Element to define a minimum value of an
attribute.</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="NominalScaledType">
                        <xs:annotation>
                            <xs:documentation>Element of to define constraints of nominal scaled attribute
values.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence minOccurs="0">
                                <xs:element name="RequiredValue" type="xs:string" minOccurs="0" maxOccurs="unbounded">
                                    <xs:annotation>
                                        <xs:documentation>Element to define a required value of an attribute. It may be defined
multiple times in order to define a discrete value range of the attribute.</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="UnknownType">
                        <xs:annotation>
                            <xs:documentation>Element to define constraints for attribute values of an unknown scale
type.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence minOccurs="0">
                                <xs:element name="Requirements" type="xs:string">
                                    <xs:annotation>
                                        <xs:documentation>Defines informative requirements as a constraint for an attribute
value.</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
```

```
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:choice>
            <xs:attribute name="Name" type="xs:string" use="required">
                <xs:annotation>
                    <xs:documentation>Describes the name of the contraint.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="MappingType">
    <xs:annotation>
        <xs:documentation>Base element for AttributeNameMapping and InterfaceIDMapping.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="CAEXBasicObject">
            <xs:sequence minOccurs="0">
                <xs:element name="AttributeNameMapping" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Allows the definition of the mapping between attributes of a related role class or
its interfaces and attributes of the hosting system unit</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="CAEXBasicObject">
                                <xs:attribute name="SystemUnitAttributeName" type="xs:string" use="required"/>
                                <xs:attribute name="RoleAttributeName" type="xs:string" use="required"/>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="InterfaceIDMapping" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Allows the definition of the mapping between interfaces of a related role class
and interfaces of the hosting system unit.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="CAEXBasicObject">
                                <xs:attribute name="SystemUnitInterfaceID" type="xs:string" use="required"/>
                                <xs:attribute name="RoleInterfaceID" type="xs:string" use="required"/>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="SourceDocumentInformationType">
    <xs:annotation>
        <xs:documentation>Defines a structure to model information about the data source of the present CAEX
document.</xs:documentation>
    </xs:annotation>
    <xs:attribute name="OriginName" type="xs:string" use="required">
        <xs:annotation>
            <xs:documentation>Name of the origin of the CAEX document, e.g. the source engineering tool or an exporter
software</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="OriginID" type="xs:string" use="required">
        <xs:annotation>
            <xs:documentation>Unique identifier of the origin of the CAEX document, e.g. a unique identifier of a source
engineering tool or an exporter software. The ID shall not change even if the origin gets renamed. </xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="OriginVendor" type="xs:string" use="optional">
        <xs:annotation>
            <xs:documentation>Optional: the vendor of the data source of the CAEX document</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="OriginVendorURL" type="xs:string" use="optional">
        <xs:annotation>
            <xs:documentation>Optional: the vendors URL of the data source of the CAEX document</xs:documentation>
        </xs:annotation>
```

```
        </xs:attribute>
        <xs:attribute name="OriginVersion" type="xs:string" use="required">
            <xs:annotation>
                <xs:documentation>Version of the origin of the CAEX document, e.g. the version of the source engineering tool or the exporter software.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="OriginRelease" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Optional: release information of the origin of the CAEX document, e.g. the version of the source engineering tool or the exporter software.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="LastWritingDateTime" type="xs:dateTime" use="required">
            <xs:annotation>
                <xs:documentation>Date and time of the creation of the CAEX document.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="OriginProjectTitle" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Optional: the title of the corresponding source project</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="OriginProjectID" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Optional: a unique identifier of the corresponding source project</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:attributeGroup name="lifecycleInformation">
        <xs:annotation>
            <xs:documentation>Tha: Groups attributes related to lifecycle information</xs:documentation>
        </xs:annotation>
        <!-- tha -->
        <xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
            <xs:annotation>
                <xs:documentation>Timestamp for the last change of the model.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="LifeCylePhase" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Describes the LifeCyclePhase in which the described model is used.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="ModelType" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Describes the type of the described model. (e.g. functional model, component model, equipment model, SW model, deployment model</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="RelationToOtherModels" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Describes the relation of the described model to other models.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:element name="CAEXFile">
        <xs:annotation>
            <xs:documentation>Root-element of the CAEX schema. </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="CAEXBasicObject">
                    <xs:sequence>
                        <xs:element name="SuperiorStandardVersion" type="xs:string" minOccurs="0" maxOccurs="unbounded">
                            <xs:annotation>
                                <xs:documentation>Describes the version of a superior standard, e.g. AutomationML x.y. The version string is defined in the superior standard.</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="SourceDocumentInformation" type="SourceDocumentInformationType" maxOccurs="unbounded">
                            <xs:annotation>
                                <xs:documentation>Provides information about the source(s) of the CAEX document.</xs:documentation>
                            </xs:annotation>
```

```
                    </xs:element>
                    <xs:element name="ExternalReference" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Container element for the alias definition of external CAEX files. Tha: instead of a path an UriRef may be provided, additional format information regarding the referenced document may be provided.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXBasicObject">
                                    <xs:attribute name="Path" type="xs:string" use="required">
                                        <xs:annotation>
                                            <xs:documentation>Describes the path of the external CAEX file. Absolute and relative paths are allowed.</xs:documentation>
                                        </xs:annotation>
                                    </xs:attribute>
                                    <xs:attribute name="Alias" type="xs:string" use="required">
                                        <xs:annotation>
                                            <xs:documentation>Describes the alias name of an external CAEX file to enable referencing elements of the external CAEX file.</xs:documentation>
                                        </xs:annotation>
                                    </xs:attribute>
                                    <xs:attribute name="Format" type="xs:string" use="optional">
                                        <xs:annotation>
                                            <xs:documentation>Tha: Describes the type(format) of the referenced document</xs:documentation>
                                        </xs:annotation>
                                        <!-- tha -->
                                    </xs:attribute>
                                    <xs:attribute name="UriRef" type="xs:anyURI" use="optional">
                                        <xs:annotation>
                                            <xs:documentation>Tha: URI, which may reference different types of external documents (e.g. CAEX, Collada, PLCopen) as well as standardized databases (e.g. IEC CDD). If UriRef is provided, path is not evaluated. </xs:documentation>
                                        </xs:annotation>
                                        <!-- tha -->
                                    </xs:attribute>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="InstanceHierarchy" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Root element for a system hierarchy of object instances. Tha: == System Model. The model is represented by a hierarchy of object instances. Additionally lifecycle information, a reference to an related model and a collection of element relations may be provided.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXObject">
                                    <xs:sequence>
                                        <xs:element name="RelatedModel" minOccurs="0" maxOccurs="unbounded">
                                            <xs:annotation>
                                                <xs:documentation>Tha: Reference to related models (e.g. if a model is derived from an other model).</xs:documentation>
                                            </xs:annotation>
                                            <!-- tha -->
                                            <xs:complexType>
                                                <xs:attribute name="ModelId" type="xs:string" use="required"/>
                                                <xs:attribute name="RelationType" use="required">
                                                    <xs:simpleType>
                                                        <xs:restriction base="xs:string">
                                                        <xs:enumeration value="equivalent"/>
                                                        <xs:enumeration value="implements"/>
                                                        <xs:enumeration value="partOf"/>
                                                        </xs:restriction>
                                                    </xs:simpleType>
                                                </xs:attribute>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="InternalElement" type="InternalElementType" minOccurs="0" maxOccurs="unbounded">
                                            <xs:annotation>
                                                <xs:documentation>Shall be used in order to define nested objects inside of a SystemUnitClass or another InternalElement. Allows description of the internal structure of a CAEX object.</xs:documentation>
                                            </xs:annotation>
```

```
                                        </xs:element>
                                        <xs:element name="ElementRelationCollection" type="ElementRelationCollection"
minOccurs="0">
                                            <xs:annotation>
                                                <xs:documentation>Tha: Collection of ElementRelations between elements
within the InstanceHierarchy.</xs:documentation>
                                            </xs:annotation>
                                            <!-- tha -->
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attributeGroup ref="lifecycleInformation"/>
                                </xs:extension>
                                <!-- tha -->
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="ElementRelationCollection" type="ElementRelationCollection" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Tha: Collection of ElementRelations between elements of different
InstanceHierarchy-Elements.</xs:documentation>
                        </xs:annotation>
                        <!-- tha -->
                    </xs:element>
                    <xs:element name="InterfaceClassLib" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Container element for a hierarchy of InterfaceClass definitions. It shall
contain any interface class definitions. CAEX supports multiple interface libraries.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXObject">
                                    <xs:sequence>
                                        <xs:element name="InterfaceClass" type="InterfaceFamilyType" minOccurs="0"
maxOccurs="unbounded">
                                            <xs:annotation>
                                                <xs:documentation>Class definition for interfaces.</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="RoleClassLib" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Container element for a hierarchy of RoleClass definitions. It shall contain
any RoleClass definitions. CAEX supports multiple role libraries.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXObject">
                                    <xs:sequence>
                                        <xs:element name="RoleClass" type="RoleFamilyType" minOccurs="0"
maxOccurs="unbounded">
                                            <xs:annotation>
                                                <xs:documentation>Definition of a class of a role type.</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="SystemUnitClassLib" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Container element for a hierarchy of SystemUnitClass definitions. It shall
contain any SystemunitClass definitions. CAEX supports multiple SystemUnitClass libraries.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="CAEXObject">
                                    <xs:sequence>
                                        <xs:element name="SystemUnitClass" type="SystemUnitFamilyType"
minOccurs="0" maxOccurs="unbounded">
                                            <xs:annotation>
                                                <xs:documentation>Shall be used for SystemUnitClass definition, provides
definition of a class of a SystemUnitClass type.</xs:documentation>
```

```
                                        </xs:annotation>
                                    </xs:element>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="AttributeTypeLib" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Container element for a hierarchy of Attribute type definitions. CAEX
supports multiple attribute type libraries.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="CAEXObject">
                                <xs:sequence>
                                    <xs:element name="AttributeType" type="AttributeFamilyType" minOccurs="0"
maxOccurs="unbounded">
                                        <xs:annotation>
                                            <xs:documentation>Class definition for attribute Types</xs:documentation>
                                        </xs:annotation>
                                    </xs:element>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="LinkClassLib" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Tha: Container element for a library of LinkClass
definitions.</xs:documentation>
                    </xs:annotation>
                    <!-- tha -->
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="CAEXObject">
                                <xs:sequence>
                                    <xs:element name="LinkClass" type="LinkFamilyType" minOccurs="0"
maxOccurs="unbounded">
                                        <xs:annotation>
                                            <xs:documentation>Shall be used for LinkClass definition, provides
definition of a class of a Linktype.</xs:documentation>
                                        </xs:annotation>
                                    </xs:element>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="SchemaVersion" type="xs:string" use="required" fixed="3.0">
                <xs:annotation>
                    <xs:documentation>Describes the version of the schema. Each CAEX document must specify
which CAEX version it requires. The version number of a CAEX document must fit to the version number specified in the CAEX
schema file. </xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="FileName" type="xs:string" use="required">
                <xs:annotation>
                    <xs:documentation>Describes the name of the CAEX file.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:element>
</xs:schema>
```