Carl von Ossietzky Universität Oldenburg

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

# Komplexitätsmessung von Produktmodellen

Dissertation zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften

**Vorgelegt von**
**Dipl.-Inform. Stephan große Austing**

Datum der Disputation:
**27.04.2012**

# Zusammenfassung

Die Bewertung der Produktivität in der Produktentwicklung ist eine entscheidende Aufgabe im Entwicklungscontrolling und beeinflusst entscheidend die Wettbewerbsfähigkeit von Unternehmen. Die moderne Produktentwicklung ist hierbei besonders durch die Einbindung verschiedener Domänen und Entwicklungsdisziplinen sowie den Einfluss externer Faktoren geprägt. Die Bewältigung der zahlreichen Beziehungen und Abhängigkeiten in und zwischen diesen Sichten stellt eine große Herausforderung dar und bestimmt maßgeblich den Aufwand. Diese Eigenschaft von Entwicklungsprojekten kann als „Komplexität“ beschrieben werden. Das Ziel dieser Arbeit ist die individuelle, quantifizierte Bewertung der Komplexität auf Grundlage der Produktmodelle.

Als Produktmodelle sind alle Teilmodelle der Entwicklung zu verstehen, die das zukünftige Produkt im Sinne einer bestimmten Entwicklungsdisziplin beschreiben. Dies umfasst zum Beispiel 3D-CAD-Modelle, Funktionsmodelle, Quellcode oder Anforderungsdokumente. Diese Modelle stellen den Output des Entwicklungsprozesses dar und ihnen kann ein Aufwand im Sinne von Arbeitsstunden oder Kosten zugeordnet werden.

Kernelement dieser Arbeit ist ein Konzept zur individuellen Analyse der Abhängigkeit des Aufwands von Attributen und Relation der Modellelemente und von Strukturen in einem Produktmodell. Hierzu wird das Produktmodell als Graph repräsentiert und durch eine Vielzahl generisch erzeugter Knotenmetriken beschrieben. Diese Beschreibung wird durch eine Regressionsanalyse ausgewertet und resultiert in einem Schätzmodell für die Komplexität im Sinne des Aufwands zur Erzeugung. Durch dieses Modell kann eine Kennzahl „Komplexität“ gemessen und Produktmodelle quantitativ miteinander verglichen werden. Die Kennzahl lässt sich strategisch zur Produktivitätsanalyse und operativ zur Fortschrittskontrolle anwenden.

Schlagwörter: Produktmodell, Komplexität, Produktivität

# Abstract

The evaluation of productivity in product development is a crucial task in development controlling and impacts the competitive capability of companies significantly. The modern product development is strongly characterized by the integration of different domains and development disciplines and also the influence of external factors. The management of these numerous relations and dependencies in and between these views is a daunting challenge and determines the required effort. This property of development projects can be described as “complexity”. The goal of this thesis is the individual, quantified evaluation of complexity based on product models.

As product models are considered all partial models in the development process that describe the future product in terms of a given development discipline. E.g., this includes 3D CAD models, functions models, source code and requirements documents. These models represent the output of the development process and they can be associated with an effort in terms of working hours or costs.

The basic element of this thesis is a concept for an individual analysis of the dependencies between effort and the attributes and relations of model elements and also the structures in a product model. For this the product model is represented as graph and described by a multitude of generically generated node metrics. The description is evaluated in a regression analysis und results in an estimation model for complexity in terms of effort required for creation. Using this model the key figure “complexity” can be measured, that allows the quantified comparison of product models. This key figure can be applied strategically for productivity analysis and operationally for progress controlling.



Keywords: Product model, complexity, productivity

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

Die Diskussion der Komplexität ist allgegenwärtig in der modernen Produktentwicklung. Zwar besteht keine allgemein akzeptierte Definition des Begriffes (Seth Lloyd 2001), jedoch besteht Einigkeit über den entscheidenden Einfluss auf die Produktentwicklung (Davis und LeBlanc 1988). Doch warum sollte man versuchen Komplexität zu messen und welcher Nutzen kann hierdurch im Zusammenhang der Produktentwicklung erzielt werden? In diesem Kapitel wird die Motivation dieses Themas erläutert und welche Herausforderungen mit diesem verbunden sind. Der Aufbau dieser Arbeit und die Rolle der einzelnen Kapitel für die Ziele der Arbeit werden beschrieben.

## 1.1 Motivation

Jeder wächst mit dem Problem der Komplexitätsbeherrschung auf: Wenn wir nun als Kind unseren ersten großen LEGO-Bausatz auspacken und zusammensetzen wollen, in der Schule einen Aufsatz verfassen oder einen Computer konfigurieren, spielt Komplexität in einer seiner vielen Ausprägungen eine Rolle. Wir lernen, dass bei komplexen Gegebenheiten viele Dinge miteinander in Beziehung stehen und wir diese Beziehungen in ihrer Gesamtheit betrachten müssen; ein Abarbeiten der Reihe nach ist nicht mehr möglich. Komplexität begegnet uns immer wieder im Alltag und in den meisten Fällen haben wir gelernt mit ihr umzugehen. In einigen Fällen sind Hilfsmittel erforderlich um den Überblick zu behalten.

Eine der bekanntesten Lösungen stellt die Neugestaltung des Planes (Abbildung 1) der Londoner U-Bahn (die „Tube“) im Jahr 1933 durch den Elektriker Harry Beck dar. Er orientierte sich bei der Gestaltung an elektrischen Schaltplänen und abstrahierte von den exakten geografischen Gegebenheiten. Hierdurch reduzierte er die Komplexität der Karte und schuf damit eine Ikone der Informationsdarstellung, die auf der ganzen Welt übernommen wurden (Billson 2002).


Abbildung 1: Londoner U-Bahn-Karte von 1933
Quelle: Transport for London Homepage

Heute helfen außerdem Navigationssysteme und Mobiltelefone bei der Orientierung und nehmen dem Benutzer die Interpretation von großen Straßenkarten ab. Die Bewältigung von Komplexität ist eine der großen Herausforderungen der Moderne. In einem Artikel der New York Times im Mai 2010 (Segal 2010) wird die Bedeutung von Komplexität in Krieg, Politik und Finanzsystemen behandelt und Segal stellt diese sogar als eine Gefahr für das Bestehen der Gesellschaft dar.

Ein Anwendungsbereich, der sehr stark auf die Beherrschung von Komplexität angewiesen ist, ist die Produktentwicklung. Zahlreiche Komponenten müssen so miteinander verbunden werden, dass das Produkt seine Funktion erfüllen kann. Ein unverzichtbares Hilfsmittel um dies zu erreichen sind Produktmodelle in Form von Plänen, Beschreibungen oder Computermodellen des Produktes. Zum Beispiel dienen die elektrischen Schaltpläne, die Beck damals als Vorbild nahm, der Komplexitätsreduzierung bei elektrischen Anlagen und helfen bei der Koordination der Schaltungselemente durch die Reduktion auf die wesentlichen Informationen. Diese Pläne sind ein Beispiel für den Einsatz von Modellen, ohne die heute keine Produktentwicklung mehr denkbar ist.

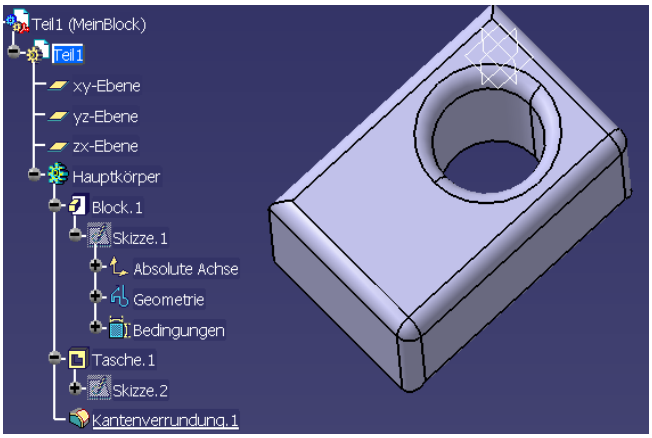


Abbildung 2: CAD-Modell – ein Produktmodell

Die Abbildung 2 zeigt ein CAD-Modell, welches den Aspekt der Geometrie eines Produktteiles beschreibt. Diese Information reicht jedoch noch nicht zur Herstellung des Produktes aus, sondern stellt nur eine Sicht auf das Produkt dar. Es fehlen zum Beispiel Informationen zum Material, welches in anderen Produktmodellen beschrieben ist. Produktentwicklungsprozesse beschäftigen sich somit mit mehreren Modellen für unterschiedliche Aspekte des zukünftigen Produktes. Sie ermöglichen Aussagen über das Aussehen, die Struktur und das zu erwartende Verhalten des Produktes (zum Beispiel Crash-Modelle oder physikalische Modelle) oder werden zur Erzeugung eingesetzt (zum Beispiel Quellcode, Steuerungsdaten für Werkzeugmaschinen). Moderne Produkte zeichnen sich durch eine hohe Anzahl unterschiedlicher und miteinander verknüpfter Modelle aus. Die große Herausforderung hierbei ist, dass aus der Vielzahl dieser Beziehungen ein bisher kaum gekanntes Maß an Komplexität in der Produktentwicklung entsteht. Das Management dieser Komplexität ist eine der Hauptaufgaben eines Ingenieurs und erfordert einen hohen Aufwand. Allerdings ist es schwierig, bis gar nicht möglich festzustellen, welchen Umfang dieser Aufwand einnimmt. Haben in früheren Projekten eher greifbare und lineare Größen, wie zum Beispiel die Größe eines Bauwerks, hinreichende Aussagen zum Aufwand gegeben, so wird der Faktor der Komplexität in den Produktmodellen immer ausschlaggebender in Zukunft.

Dies hat auch direkte Auswirkungen auf die Bewertung von Produktivität in der Produktentwicklung, da die Leistung nicht gemessen werden kann, die in der Entwicklung eines Produktmodells oder mehrerer miteinander verknüpfter Modelle liegt. Auch das weit verbreitete Überschreiten des Budgets und der Zeit in Projekten lässt sich mit Komplexität erklären, aber nicht mit den ursprünglichen Planungen bewerten. Zu unterschiedlich sind die Handhabung von Puffern und Anpassungen des Projektrahmens im Laufe der Projekte (T. Williams 2002, 4–8).

Man nehme als Beispiel eine Software, die zunächst nur für einen bestimmten Kunden gedacht ist. Da die entstehende Lösung sehr vielversprechend ist, wird beschlossen, es

als allgemeines Produkt zu entwickeln. Hierdurch ändert sich die Komplexität des Produktes stark, da andere Plattformen und neue Interessen berücksichtigt werden müssen. Der Umfang des Produktes bleibt zwar vergleichbar, aber die Entwicklung verzögert sich merklich. Es bleibt unklar, ob Probleme vorliegen oder ob solche Verzögerungen bei dieser Erweiterung des Projektrahmens akzeptabel sind. Wenn sich Projekte in ihrer Produktivität nicht anhand der Planung abschätzen lassen, so muss eine andere Größe gefunden werden. Durch eine Messbarmachung von Komplexität kann eine solche Größe realisiert werden.

In der Literatur sind bereits zahlreiche Ansätze zur Messung der Komplexität von Produktmodellen beschrieben (siehe Abschnitt 2.1.3.3). Allerdings können sich diese Ansätze nicht hinreichend an die verschiedenen Ausprägungen von Komplexität, insbesondere im domänenübergreifenden Bereich, anpassen. Sie liefern nur unter bestimmten Voraussetzungen verlässliche Zahlen und sind zum Teil mit hohem manuellem Aufwand verbunden.

Um diese Defizite zu beheben wird im Rahmen dieser Dissertation eine Methode zur Komplexitätsmessung von Produktmodellen entwickelt, die das Management von Entwicklungsprojekten und die Bewertung der Produktivität von Entwicklungsprozessen unterstützt. Hierbei wird insbesondere auf die Verknüpfungen zwischen verschiedenen Sichten auf das Produkt eingegangen und ein domänenunabhängiger Ansatz verwendet. Hiermit wird zum einen ein wissenschaftlicher Beitrag zum Verständnis von Komplexität und deren Auswirkung in der Produktentwicklung geleistet; die Methode soll ein Werkzeug für Untersuchungen zu Zusammenhängen in diesem Umfeld bieten. Zum anderen wird ein praktischer Beitrag zur Bewältigung von Komplexität geleistet, indem die Komplexität in konkreten Projektsituationen bewertet und entsprechend reagiert werden kann.

## 1.2 Problemstellung

Welche Herausforderungen ergeben sich aus der Motivation dieser Arbeit? Zunächst stellt der Begriff „Komplexität“ eine große Herausforderung für sich dar, da kein einheitliches Verständnis von Komplexität vorliegt. In vielen Fällen wird sie nur als eine negative Begleiterscheinung von Produktentwicklung betrachtet und die Methoden dienen entsprechend vor allem ihrer Vermeidung (Lindemann, Maurer, und Braun 2008, 21). Andere Autoren (zum Beispiel (Albrecht 1979)(Griffin 1993)(Feldhusen und Koy 2002)) sehen in Komplexität hingegen einen Ausdruck von Entwicklungsleistung und setzen diesen als Leistungsparameter in ihren Ansätzen ein. Doch was unterscheidet die Komplexität als „Problem“ von der Komplexität als „Leistung“? Es ist offensichtlich, dass verschiedene Verständnisse vorliegen, die nicht miteinander kompatibel sind. Um ein Komplexitätsmaß als Werkzeug zur Produktivitätsbewertung von Entwicklungspro-

jekten zu schaffen, muss dieser Arbeit zunächst eine feste begriffliche Grundlage gegeben werden; es ist daher notwendig den Komplexitätsbegriff gründlich zu erarbeiten. Hierzu zählt insbesondere die Untersuchung der verschiedenen Verständnisse in der Literatur um auch eine Abgrenzung oder Vergleich zu bestehenden Ansätzen zu ermöglichen. Die Festlegung der Begrifflichkeiten in dieser Arbeit soll sich an den in der Motivation genannten Grundzielen orientieren. Hieraus ergibt sich die folgende wissenschaftliche Fragestellung:

Theoretische Fragestellung: Wie können Verständnisse von Komplexität in der Produktentwicklung, unter anderem aus der Literatur, geordnet werden und wie kann eine Definition gestaltet werden, die sich an den Zielen der Produktivitätsmessung in der Produktentwicklung orientiert?

Unter der Voraussetzung, dass diese Fragestellung beantwortet wurde, kann die Verbindung zu den Produktmodellen erfolgen. Diese können sehr unterschiedlich in ihrer Syntax und Semantik gestaltet sein, so dass ein Ansatz auf einem generischen Metamodell basieren muss. Es muss geklärt werden, wie dieses Modell gestaltet werden muss, um Produktmodelle und insbesondere auch Relationen zwischen diesen Modellen abzubilden. Des Weiteren muss auf Basis dieses Modells der Bezug zur Komplexität beschrieben werden; das heißt wie ganze Modelle oder Modellelemente auf Komplexitäten abgebildet werden. Diese Abbildung soll quantitativ gestaltet werden, wobei Wertebereich und Einheit zu definieren sind. Eine wichtige Eigenschaft ist außerdem, dass die Abbildung nicht von modellexternen Faktoren abhängig ist. Dies bedeutet, die Komplexität des Produktmodells ist eine intrinsische Eigenschaft und hängt nicht vom Entwickler, Werkzeugen oder anderen Umständen ab. Insgesamt kann dieser Aspekt durch die folgende wissenschaftliche Fragestellung zusammengefasst werden.

Konzeptionelle Fragestellung: Wie kann ein Metamodell von Produktmodellen zur Komplexitätsbewertung definiert und die quantifizierende Analyse dieser Modelle gestaltet werden?

Wenn ein Modell der Komplexität von Produktmodellen und ein Konzept zur Bewertung dieser vorhanden sind, muss schließlich die Umsetzung in Form von Werkzeugen für die Produktivitätsmessung erfolgen. Für Produktentwicklungen, wie der in der Motivation als Beispiel genannten erweiterten Softwareentwicklung, muss eine praktikable Implementierung der Messung der Produktivität möglich sein. Eine große Herausforderung hierbei ist, dass Benutzer abhängig von ihrem Verständnis von Komplexität den Bezug dieser Messgröße zu ihren Aufgabenstellungen fehlinterpretieren. Es muss daher untersucht werden, wie Komplexität gegenüber dem Benutzer präsentiert wird. Die quantifizierende Analyse der Produktmodelle soll schließlich nicht selber zur Komplexität beitragen, sondern eine einfachere Bewertung ermöglichen. Die Einbindung in bestehende Werkzeuge der Produktentwicklung ist daher wünschenswert. Die Anforde-

rungen für den praktischen Einsatz müssen prototypisch evaluiert werden, um sowohl den wissenschaftlichen als auch praktischen Wert des Ansatzes zu belegen. Des Weiteren muss validiert werden, dass die Komplexitätsmessung ein geeignetes Mittel zur Bewertung der Produktivität in der Produktentwicklung ist. Dieser Aspekt wird durch die folgende praktische Fragestellung zusammengefasst:

Praktische Fragestellung: Wie muss ein Werkzeug zur Komplexitätsanalyse gestaltet sein, um eine Produktivitätsbewertung anhand von Produktmodellen durchzuführen?

Der Kern dieser Arbeit besteht somit aus zwei wissenschaftlichen und einer vorwiegend praktischen Fragestellung. Das Ziel dieser Arbeit ist die Beantwortung dieser Fragestellungen in ihrer Gesamtheit. Weder eine rein theoretische oder praktische Behandlung der Komplexität in der Produktentwicklung kann die in der Motivation beschriebene Problematik bewältigen. Zur Vereinfachung werden diese Aspekte als theoretische, konzeptionelle und praktische Fragestellung im Folgendem referenziert.

## 1.3 Aufbau der Arbeit

Der Aufbau der Arbeit orientiert sich an den drei wissenschaftlichen Fragestellungen, die im vorherigen Abschnitt formuliert wurden. Jede Fragestellung wird durch ein oder mehrere Kapitel adressiert. Die Bearbeitung der nachfolgenden Fragestellung baut hierbei jeweils auf den Ergebnissen der entsprechenden Kapitel auf.

Das Kapitel 1 „Einleitung“ bot eine Motivation und hat die Problemstellung dieser Arbeit entwickelt. Durch die Definition der wissenschaftlichen, konzeptionellen und praktischen Fragestellungen wurde die Grundlage dieser Arbeit gelegt.

Das Kapitel 2 „Messung von Komplexität“ erarbeitet den Komplexitätsbegriff in der Produktentwicklung und wie Komplexität im Umfeld der Aufgabenstellung sinnvoll definiert werden kann. Erst am Ende dieses Kapitel kann auf Basis dieser Betrachtung ein Anforderungskatalog erstellt werden. Die Definitionen und der Anforderungskatalog beantworten die theoretische Fragestellung dieser Arbeit.

Die Kapitel 3 und 4 beschreiben verwandte Ansätze aus dem Stand der Technik. Die dargestellten Ansätze zur Leistungsmessung besitzen vergleichbare Ziele; die verwandten Ansätze der Strukturanalyse von Produktmodellen besitzen methodische Bezüge zu dieser Arbeit. Die verwandten Ansätze werden außerdem in Abschnitt 5.9 gegenüber dem vorgestellten Ansatz abgegrenzt, beziehungsweise deren Einfluss auf die Gestaltung des Ansatzes dieser Arbeit wird aufgezeigt.

Das Kapitel 5 „Konzeption eines Ansatzes zur Komplexitätsmessung von Produktmodellen“ stellt den inhaltlichen Schwerpunkt dieser Arbeit dar und entwickelt auf Basis der Ergebnisse des Kapitels 2 und der verwandten Methoden den Ansatz zur Komplexitätsmessung. Das Konzept des Ansatzes wird detailliert und Einsatzmöglichkeiten skiz-

ziert. Die Ergebnisse dieses Kapitels beantworten die konzeptionelle Fragestellung dieser Arbeit.

Das Implementierungskapitel zeigt eine mögliche Implementierung des Ansatzes. Generell ist der Ansatz jedoch nicht an eine bestimmte Technologie gebunden. Das Kapitel demonstriert, wie das Konzept flexibel und mit relativ wenig Aufwand mit Hilfe des Eclipse Modeling Framework umgesetzt werden kann. Die Implementierung des Prototyps und die darauf basierende Validierung adressieren die praktische Fragestellung dieser Arbeit.

Im 7. Kapitel „Validierung des Ansatzes“ werden drei Fallbeispiele mit unterschiedlichem Fokus vorgelegt, anhand deren der Ansatz zur Erstellung eines Komplexitätsmaßes validiert wird. Es werden verschiedene Domänen und Arten von Produktmodellen hierbei betrachtet. Die Ergebnisse werden mit dem im Abschnitt 2.2.2 entwickelten Anforderungsmodell abgeglichen und eine Gesamtbewertung des Komplexitätsmaßes vorgenommen.

Im letzten Kapitel wird eine Bewertung der Ergebnisse der Arbeit durchgeführt und geprüft, wie diese die zu anfangs formulierten Fragestellungen adressieren. Außerdem werden eine Zusammenfassung des Beitrages zum Stand der Wissenschaft und ein Fazit gegeben.

Abbildung 3 veranschaulicht den Aufbau der Arbeit und stellt die inhaltlichen Bezüge dar.

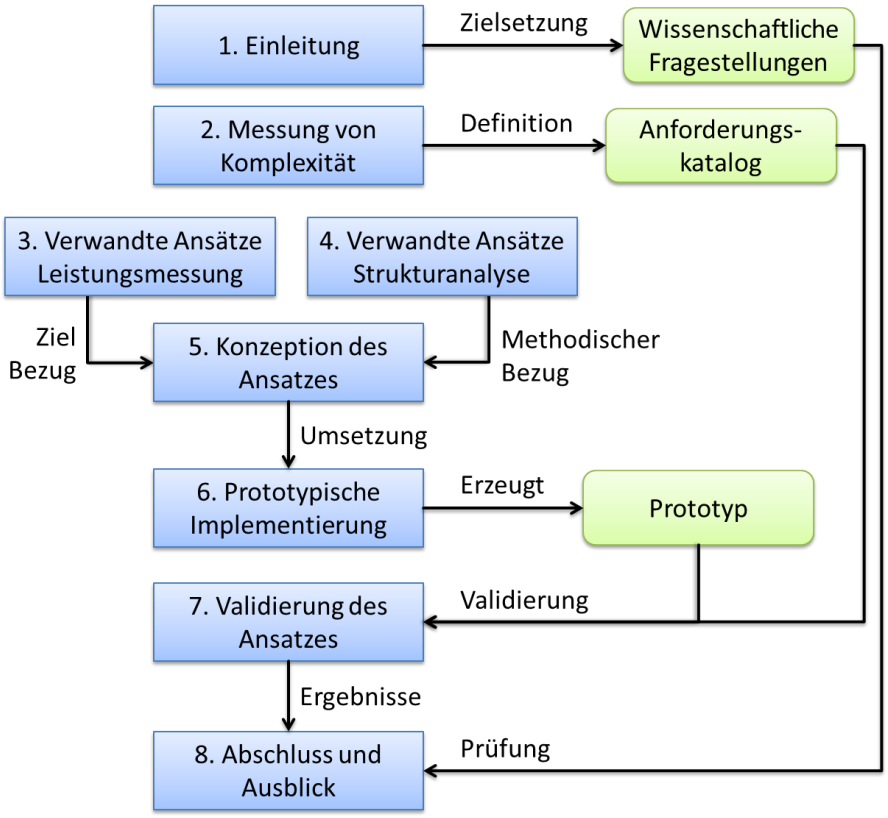


Abbildung 3: Aufbau der Arbeit

# 2 Messung von Komplexität

Dieses Kapitel bildet die theoretische Grundlage des Ansatzes zur Komplexitätsmessung und behandelt somit die theoretische Fragstellung dieser Arbeit und definiert den Bezugsrahmen. Hierzu wird der Komplexitätsbegriff in der Produktentwicklung durch eine Literaturrecherche im Abschnitt 2.1 erarbeitet. Auf dieser Recherche und einer Kategorisierung aufbauend werden Anforderungen in Abschnitt 2.2 erstellt, die als Ausgangspunkt für die Entwicklung des Ansatzes dienen.

## 2.1 Komplexität

Um Komplexität als solche zu messen, muss zunächst der Begriff der Komplexität genauer erarbeitet werden. Obwohl die meisten Menschen eine Vorstellung dieses Begriffs besitzen, existiert keine einheitliche Definition, wie in dem folgenden Abschnitt deutlich wird. Daher wird zunächst dieser Begriff aus einem historischen Kontext sowie verschiedenen wissenschaftlichen Disziplinen betrachtet und eine Kategorisierung der Verständnisse von Komplexität eingeführt. Es wird gezeigt, dass diese Kategorien bestimmten Zielen dienen und dass für die Verwendung im Bereich der Produktentwicklung diese Ziele formuliert werden müssen, um die notwendigen Eigenschaften eines geeigneten Komplexitätsbegriffes für Produktmodelle abzuleiten.

### 2.1.1 Aspekte des Komplexitätsbegriffes

Wann wird eine Sache als komplex betrachtet? Auch wenn Komplexität sehr unterschiedlich definiert und verwendet wird, treten bestimmte Aspekte in der Verwendung dieses Begriffes und in der Literatur immer wieder auf. Durch die Beschreibung dieser Aspekte erfolgt eine Annäherung an die quantitative Bewertung von Komplexität. Dieser Abschnitt dient somit als Vorbereitung für die Eingrenzung des Komplexitätsbegriffs im Rahmen dieser Arbeit.

Der erste Aspekt ist die Menge an Informationen, die durch komplexe Objekte repräsentiert wird. In der Quantentheorie sind Information und Komplexität zwei eng verwandte Begriffe. Die Komplexität des Universums (Stern, Planeten, Lebensformen) wird zum Beispiel durch Lloyd (Seth Lloyd 2007) als Information betrachtet, die durch das Universum selbst berechnet wird. In Anlehnung an das Infinite Monkey Theorem, nachdem bei unendlichen vielen Affen mit Schreibmaschinen einer dabei sein wird, der Shakespeares Werke fehlerlos niederschreibt, beschreibt er diesen Vorgang bildlich. Das „Programmieren des Universums“ ist hierbei ein Sinnbild für die Zusammenführung der Komplexitätsbegriffe der Physik und der Informationstheorie.

*„Quantum fluctuations are the monkeys that program the universe."*(Seth Lloyd 2007, 186).

Im Produktentwicklungsprozess werden ebenfalls Informationen über das zukünftige Produkt gesammelt. Je mehr Beziehungen zwischen den Komponenten existieren, umso mehr Informationen werden in dem Produktmodell dargestellt und umso komplexer ist es. Komplexität ist wird also oft als Informationen betrachtet, sei es auf quantentheoretischer Ebene, auf Ebene der Informationstheorie als auch in der Produktentwicklung.

Der zweite Aspekt sind die Variablen eines Systems. Weaver (Weaver 1948) unterscheidet hierbei zwei Arten von Komplexität: Unorganisierte und organisierte Komplexität. Unorganisierte Komplexität zeichnet sich durch eine unüberschaubare Menge an Variablen in einem System aus, dessen Gesamtverhalten jedoch trotz dessen genau beschrieben werden kann. Zum Beispiel kann das Verhalten eines Gases bestimmt werden, jedoch nicht das eines einzelnen Moleküls in diesem Gas. Weaver setzt dem Begriff der unorganisierten Komplexität den der organisierten Komplexität entgegen. Eine hohe aber greifbare Anzahl an Parametern, die miteinander in Beziehung stehen, bilden ein komplexes System, das weder berechnet noch statistisch beschrieben werden kann. Er betont außerdem die Wichtigkeit von interdisziplinären Teams bei der Lösung solcher Probleme.

Der letzte Aspekt ist die Emergenz von Eigenschaften. Grundlage dieses Aspekts ist eine durch Aristoteles formulierte Sichtweise: *„Das Ganze ist mehr als die Summe seiner Teile"*. Auf Grundlage der Arbeiten des Philosophen (Mill 1843) entwickelte die Philosophierichtung der englischen Emergentisten eine Theorie von verschiedenen Ebenen der Komplexität. Auf der untersten Ebene liegen grundlegende physikalische Gesetze auf denen weitere Ebenen wie Chemie oder Soziologie fußen. Ein komplexes System liegt vor, wenn neue Eigenschaften entstehen, die auf Grundlage der Ausgangsebene nicht erklärt werden können und auf einer neuen Ebene behandelt werden müssen. (Johnson 2005) beschreibt den Einfluss emergenter Eigenschaften auf das Engineering komplexer Systeme. Die hier genannten Aspekte von Komplexität sind in der Tabelle 1 zusammengefasst. Die Aspekte betonen sehr unterschiedliche Sichten auf Komplexität und ergänzen sich daher zum Teil.

Tabelle 1: Aspekte der Komplexität

| **Aspekt** | **Beispieldisziplinen** | **Autoren** |
|---|---|---|
| Information | Quantenphysik | (Seth Lloyd 2007) |
| Unüberschaubare Anzahl an Variablen (Unorganized) | Physik, Chemie | (Weaver 1948) |
| Hohe Anzahl Variablen (Organized) | Organisation, Engineering | (Weaver 1948) |
| Emergenz von Eigenschaften | Philosophie, Engineering | (Mill 1843), (Johnson 2005) |

Wenn Systeme ein oder mehrere dieser genannten Aspekte aufweisen, werden sie wahrscheinlich als komplex bezeichnet. Da diese Aspekte häufig miteinander korreliert sind, werden sie meistens nicht unterschieden. Zum Beispiel wird eine biologische Zelle aus unterschiedlichen Gründen als komplex betrachtet:

- Sie besitzt eine große Menge an Informationen, die in Form der DNA kodiert ist.
- Die chemischen Prozesse innerhalb der Zelle bestehen aus einer unüberschaubaren Menge an Variablen.
- Sie hat eine hohe aber fassbare Variabilität bezüglich ihres Phänotyps.
- Sie zeigt emergente Eigenschaften in Form ihrer Selbstreplikation.

Entsprechend werden die Wissenschaftler aus der Bioinformatik, organischen Chemie und Biologie übereinstimmen, dass organische Zellen komplex sind. Jedoch sind deren Verständnisse von Komplexität sehr verschieden gelagert. In der Produktentwicklung verhält sich dies mit dem Verständnis von Komplexität oftmals ähnlich. Dies führt dazu, dass bezüglich Komplexität oftmals aneinander vorbeigeredet wird und es zu Missverständnissen kommt. Die verschiedenen aber trotzdem oftmals schlecht zu differenzierenden Aspekte der Komplexität zeigen auf, dass Komplexität sich auf viele Arten ausdrücken kann. Entsprechend stellt sich für die Quantifizierung die Frage, welcher Ausdruck beziehungsweise welche Eigenschaften herangezogen werden sollen. Auf Basis der hier beschriebenen Aspekte wird daher im Folgenden eine Differenzierung diskutiert.

### 2.1.2 Quantität von Komplexität

Während Aspekte der Komplexität bereits sehr früh diskutiert wurden, entwickelten sich erst ab den 50er Jahren des letzten Jahrhunderts in der Komplexitätstheorie und in einigen Anwendungsdomänen wie der künstlichen Intelligenz Ansätze zur Quantifizie-

rung von Komplexität. In Analogie zu den Aspekten besteht hier ebenfalls keine Einigkeit. (Edmonds 1999) beschreibt im Anhang seiner Dissertation 48 verschiedene Maße für Komplexität. (Seth Lloyd 2001) nennt 31 Maße für Komplexität und bietet eine Kategorisierung in drei Grundverständnisse an. Diese drei Verständnisse werden in den folgenden Unterabschnitten erläutert und zur Kategorisierung genutzt.

Für die Entwicklung des Komplexitätsbegriffs dieser Arbeit spielt die Differenzierung der Ansätze zur Quantifizierung von Komplexität eine entscheidende Rolle: Erst hierdurch wird ein Vokabular verfügbar um den hier vorgestellten Ansatz von anderen Ansätzen sinnvoll abzugrenzen, die ebenfalls den Komplexitätsbegriff verwenden. Die im Rahmen dieser Arbeit verwendete Definition der Komplexität in Abschnitt 2.2.2 wird auf Basis dieser Diskussion entwickelt. Um den Bezug zur Produktentwicklung herzustellen werden einige Bespiele für Komplexitätsmaße genannt. Eine umfassende Übersicht der wichtigsten Komplexitätsmaße liefert hingegen der Abschnitt 2.1.3.3.

#### 2.1.2.1 Schwierigkeit der Beschreibung

Komplexitätsmaße dieser Kategorie bewerten ein System nach dem Aufwand es zu beschreiben. Ein einfaches Beispiel ist die Kodierung einer Zeichenkette, welche sich durch die Anzahl an Bits messen lässt. Die Informationstheorie, basierend auf der Arbeit von (Shannon 1948), beschreibt den Informationswert einer Nachricht auf Grundlage von Wahrscheinlichkeiten der Symbole. Shannons Maß für Information ist die durchschnittliche Anzahl an Bits (1/0) pro Symbol, die notwendig ist um eine Nachricht mit N verschiedenen Symbolen zu beschreiben. Jedes Symbol hat hierbei eine feste Wahrscheinlichkeit als das nächste Symbol der Nachricht zu erscheinen.

Shannons Maß geht von einem Zufall aus, der die Symbole erzeugt. Wenn jedoch eine Struktur in einer Zeichenkette zu erkennen ist, dann lässt sich dieser jedoch noch weiter komprimieren. Als Beispiel sei eine lange Zeichenkette der Form 10101... gegeben. Der Informationswert nach Shannon würde linear zur Länge der Zeichenkette steigen. Die algorithmische Komplexität oder Kolmogorow-Komplexität, zuerst beschrieben durch (Solomonoff 1964), misst die Komplexität hingegen als das kleinste Programm, das eine gegebene Zeichenfolge produzieren kann. Die oben genannte Zeichenfolge ließe sich durch ein Java-Programm wie folgt vereinfachen:

```
for(int i = 0; i < M; i++){System.out.print("10");}
```

Dieses Verständnis von Komplexität spielt vor allem bei der Komprimierung von Daten eine Rolle; sie sagt jedoch nichts über den semantischen Wert der Zeichenkette aus. Dies drückt sich auch dadurch aus, dass zufällige Zeichenketten wie ein Hintergrundrauschen schlecht komprimierbar sind und einen sehr hohen Komplexitätswert besitzen.

#### 2.1.2.2 Aufwand der Erzeugung

Die Komplexität eines Systems wird durch Maße aus dieser Kategorie über den Aufwand definiert, der zur Erzeugung oder Lösung notwendig ist. Dies lässt sich zum Beispiel auf Texte wie diese Dissertation beziehen, die sich besser durch den benötigten Aufwand als durch die Darstellung in Anzahl der Seiten charakterisieren lassen. Übliche Dimensionen in diesem Bereich sind Berechnungsschritte, Zeit sowie Kosten in monetären Größen.

In der Komplexitätstheorie der theoretischen Informatik und auch im Forschungsfeld Operations Research, werden Komplexitätsklassen von Algorithmen (nicht zu verwechseln mit algorithmischer Komplexität) beschrieben. Diese geben den Aufwand eines Algorithmus in Abhängigkeit einer Problemgröße an. In einem Turing-Maschinenmodell kann unterschieden werden zwischen Zeitkomplexität, der Anzahl notwendiger Rechenschritte, und der Raumkomplexität, der Anzahl benötigter Zustände. Zum Beispiel liegt der Sortieralgorithmus Quicksort (Hoare 1962) für den schlechtesten Fall in der Komplexitätsklasse $O(n^2)$. Das bedeutet, der maximale Aufwand für die Sortierung einer Menge wächst quadratisch zu deren Größe.

In der Komplexitätstheorie werden die Eigenschaften der Algorithmen betrachtet und nicht die Systeme oder Lösungen, die diese Algorithmen erzeugen. Die Komplexität eines Systems ist entsprechend keine inhärente Eigenschaft des Systems, sondern von externen Faktoren abhängt. Sind diese Faktoren nicht bekannt, so kann die Komplexität nicht mehr gemessen werden, wenn das System bereits erzeugt ist. Dies ist eine wichtige Einschränkung gegenüber der vorherigen Kategorie „Schwierigkeit der Beschreibung“ und hat wesentliche Auswirkungen auf die Messung von Produktmodellen. Ein einfaches Beispiel ist die Erstellung von präzisen CAD-Modellen im Maschinenbau: Die Gestaltung und Berechnung eines umfangreichen Entwurfs mit Papier und Stift ist wesentlich aufwändiger und fehleranfälliger als mit modernen CAD-Werkzeugen, da viele Aufgaben automatisiert sind und Entwurfsprobleme wie Kollisionen von Körpern geprüft werden. Ähnlich verhält es sich im Software-Engineering, das durch integrierte Entwicklungsumgebungen und höhere Programmiersprachen die Entwicklung von Programmen vereinfacht hat. Es ist vor dem Hintergrund also eine philosophische Frage, ob der Entwurf von Fords T-Modell mit heutigen Entwurfswerkzeugen dasselbe ist, wie damals zu Zeiten von Henry Ford.

Eine praktische Konsequenz der Abhängigkeit von externen Faktoren ist, dass beliebige Produktmodelle nicht direkt miteinander verglichen werden können. Außerdem sind die Definition und das Messen der externen Faktoren oftmals schwierig. Wie misst man Teammotivation und welchen Einfluss besitzt sie überhaupt? Trotzdem bieten Maße dieser Kategorie relativ stabile Abschätzungen von Komplexität. Die Function-Point-Metrik (Albrecht 1979) bewertet zum Beispiel die Komplexität von funktionalen Be-

nutzeranforderungen und wendet externe Anpassungsfaktoren auf diese Werte an um einen angepassten Function-Point-Wert zu definieren. Die COCOMO- (Boehm 1981) und COCOMO II-Methode (Boehm, Madachy, und Steece 2000) zur Kostenabschätzung in der Softwareentwicklung, die auf dem unangepassten Function-Point-Wert basieren, benutzen hingegen wieder andere externe Projektattribute wie „Volatilität der virtuellen Maschine“ und „Fähigkeiten der Analysten“. Für diese Attribute wurden Berechnungsformeln durch eine Regression auf Basis von historischen Softwareprojekten kalibriert. Die Kostenabschätzung in COCOMO erfolgt algorithmisch durch das Einsetzen der vorliegende Projektparameter in eine Tabelle. Als Ergebnis des Einsetzens wird ein Aufwandsanpassungsfaktor ermittelt, der mit dem Function-Point-Wert multipliziert wird. COCOMO umfasst hierbei drei Modelle, die sich in der Anzahl der zu ermittelnden Attribute und Genauigkeit unterscheiden. Problematisch erweisen sich externe Faktoren, die durch COCOMO nicht abgedeckt werden. Diese können zu verfälschten Ergebnissen führen. Projektmanager müssen sich bei der Anwendung von solchen Methoden der nichtoffensichtlichen Faktoren bewusst sein, die den Aufwand entscheidend beeinflussen können (Bailey und Basili 1981).

#### 2.1.2.3 Grad der Organisation

Der Grad der Organisation ist die Schwierigkeit Strukturen eines Systems zu beschreiben. Gell-Mann und Lloyd formulieren diesen Ansatz für ihre Effektive Komplexität als *„the length of a highly compressed description of its regularities“* (Gell-Mann und Lloyd 2004). Dies bedeutet im Gegensatz zur Schwierigkeit der Beschreibung betrachtet dieser Ansatz das Verhältnis von Regelmäßigkeiten und Zufällen in einem System. (McAllister 2003) kritisiert an der Effektiven Komplexität die Willkür der Unterscheidung zwischen diesen beiden Aspekten.

Bei einem System, das zufällig aufgebaut ist, ist die effektive Komplexität sehr gering, da keine Strukturen beschrieben werden müssen. Durch das Auftreten von Strukturen und Abhängigkeiten erhöht sich die Komplexität bis zu einem bestimmten Grad. Wenn jedoch das System sehr stark strukturiert ist, so sinkt wiederrum die effektive Komplexität, da das System so regelmäßig ist, dass es sich durch eine kürzere Beschreibung erklären lässt. Die Abbildung 4 verdeutlicht diesen Unterschied zu der Kategorie Schwierigkeit der Beschreibung.

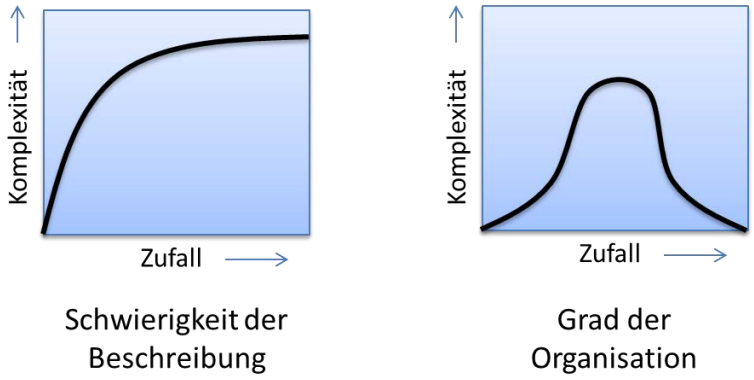


Abbildung 4: Unterschied des Einflusses von Zufall auf die Komplexitätsverständnisse
Quelle: Eigene Abbildung in Anlehnung an (Feldman und Crutchfield 1998)

Dieses Komplexitätsverständnis ist von einem Betrachter abhängig, der das System mit den Mitteln seiner Sprache beschreibt. Durch die Subjektivität und die Einbeziehung der Sprache ist diese Komplexität nur schwer objektiv messbar. Trotz dieser etwas ungewöhnlichen Definition entspricht es in den meisten Fällen dem subjektiven Verständnis von Komplexität. Zum Beispiel wird ein Buch mit vielen Charakteren und Verflechtungen, die sich nur durch eine lange Beschreibung erfassen lassen, als komplex empfunden. Ist das System der Charaktere sehr stark strukturiert (zum Beispiel in Gut und Böse), so verliert das Buch an Komplexität.

Im Kontext der Produktentwicklung können nichtzufällige Strukturen im Produktmodell als erarbeitete Designentscheidungen des Entwicklers betrachten werden. Im Gegensatz hierzu stehen willkürliche Designentscheidungen. Zum Beispiel ist die konkrete Wahl des Namens einer lokalen Variablen in eine Softwaremethode irrelevant. Die Unterscheidung dieser Fälle bei einer Messung ist offensichtlich schwierig und muss auf Basis einer Anforderungsbeschreibung erfolgen. Produkteigenschaften, die nicht auf eine oder mehrere Anforderung basieren, stellen hierbei willkürliche Elemente dar.

Die Bedeutung von Regelmäßigkeit verdeutlichen Entwurfsmuster, wie sie zum Beispiel im Software Engineering gebräuchlich sind. Sie verringern die Komplexität gemäß dieser Betrachtungsweise, indem sie Strukturen vorgeben und Entscheidungen über die Implementierung reduzieren. Das Produktmodell wird oftmals expliziter und umfangreicher aber trotzdem weniger komplex.

Im Axiomatischen Design werden Designprozesse durch Abbildungsmatrizen zwischen Entwurfsebenen beschrieben. Wenn eine Matrix diagonal ist, ist die Information und somit die Komplexität minimiert (Nam P. Suh 1999). Jede Designentscheidung beruht auf genau einer Anforderung, was die einfachste Struktur darstellt. Die Regelmäßigkeit ist am höchsten und kann durch eine kurze Beschreibung widergegeben werden.

### 2.1.3 Komplexität in der Produktentwicklung

Die Darstellung der Aspekte und der Ansätze zur Quantifizierung von Komplexität dienen zur allgemeinen Abgrenzung des Komplexitätsverständnisses in dieser Arbeit. Zur weiteren spezifischen Abgrenzung gegenüber Komplexitätsmaßen in der Produktentwicklung werden in diesem Abschnitt weitere Kategorien entwickelt.

In jeder Ingenieursdisziplin sind Methoden entwickelt worden, um Komplexität zu beherrschen. Zum Beispiel wird im Axiomatischen Design derjenige Entwurf als besser bewertet, der weniger Informationen benötigt (Nam P. Suh 1990). Ein weiteres Beispiel ist der Bereich des Software Engineerings: Hier wird Komplexität insbesondere durch abstrakte Schnittstellen unter dem Paradigma „Teile und Herrsche“ begegnet. Jedes Komplexitätsmaß beruht auf einem anderen Kontext, so dass sich die Ausgestaltung stark unterscheiden kann. Bei der Ausgestaltung können zwei Aspekte jedoch deutlich identifiziert werden. Der erste Aspekt beschäftigt sich mit dem Gegenstand der Messung, also woran die Komplexität gemessen wird. Der zweite Aspekt sind die unterschiedlichen Ziele, die mit der Messung verfolgt werden.

#### 2.1.3.1 Gegenstand der Komplexitätsmessung

Die Entwicklung von Produkten erfolgt als schrittweise Erstellung, Überarbeitung und Umwandlung von Partialmodellen. Als erste entstehen in der Regel Modelle zur Beschreibung des Entwicklungsproblems; solche Partialmodelle sind zum Beispiel Anforderungslisten. Auf Grundlage dieser Modelle werden Entwürfe entwickelt und überarbeitet. Am Ende entsteht ein vollständiges Produktmodell, das idealerweise den am Anfang ermittelten Anforderungen optimal entspricht. Je nach Disziplin und verwendeten Vorgehensmodell ist dieser Ablauf in verschiedene Phasen untergliedert, die auch zyklisch auftreten können. Beispiele hierfür sind die VDI-Richtlinie 2221 in der Konstruktionslehre (VDI2221 1985), das Wasserfallmodell in der Softwareentwicklung (Royce 1970) oder der Chipentwurf. Die Abbildung 5 beschreibt ein abstraktes Modell der Produktentwicklung und Partialmodelle, die als Ergebnis einer Phase entstehen.

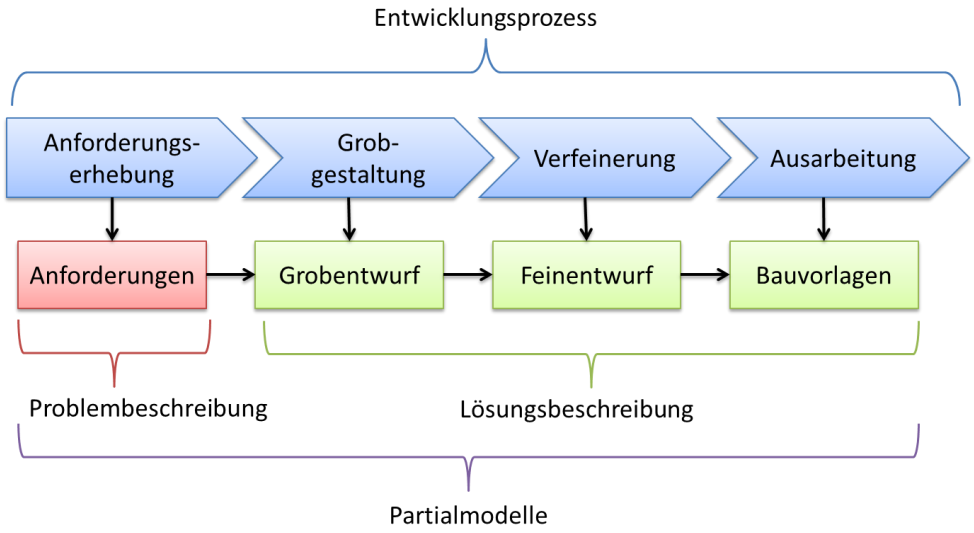


Abbildung 5: Abstraktes Modell der Produktentwicklung

In der Arbeit von (Ameri u. a. 2008) werden bestehende Komplexitätsmaße untersucht und unter anderem nach dem Gegenstand der Messung kategorisiert. Die Gegenstände werden auf Basis eines solchen abstrakten Vorgehensmodells abgegrenzt, welches an dieser Stelle an das Vokabular dieser Arbeit angepasst wurde.

#### 2.1.3.1.1 Messung des Entwicklungsprozesses

Die Beschreibung der Komplexität des Entwicklungsprozesses kann auf Basis des Graphen der Entwicklungsschritte erfolgen. In diesem Fall lassen sich Maße aus der Graphentheorie übernehmen, wie zum Beispiel das Verhältnis der Anzahl von Kanten und Knoten. (Ahn, Ramaswamy, und Crawford 1996) zeigen eine Formalisierung von Entscheidungsprozessen durch Petri-Netze. Andere Eigenschaften des Prozesses wie benötigte Zeit und Mitarbeiter lassen sich ebenfalls als Komplexitätsmaße heranziehen.

#### 2.1.3.1.2 Messung der Partialmodelle der Problembeschreibung

Diese Partialmodelle beschreiben die Umgebung, die Vorrausetzungen oder auch in strukturierter Form als Anforderungsmodell das Entwicklungsproblem. Komplexe Problemstellungen indizieren einen komplexen Entwicklungsprozess, um alle Anforderungen zu behandeln. Ein Produkt ist dann komplex, wenn dessen Anforderungen komplex sind. Dies gilt auch, wenn die Lösung durch einen geschickten Entwurf und Konstruktion relativ einfach ist. Diese Kategorie ist eng an der Beschreibung von Komplexität als Aufwand der Erzeugung gebunden.

#### 2.1.3.1.3 Messung der Partialmodelle der Lösungsbeschreibung

Als Lösungsbeschreibung können alle Partialmodelle verstanden werden, die das Produkt im engeren Sinne beschreiben. Sie definieren zu einer bestimmten Granularität

Eigenschaften des entstehenden Produktes wie zum Beispiel Struktur oder Geometrie. Die Beschränkung der Komplexitätsmessung auf diesen Aspekt hat den Vorteil, dass der Kontext nicht betrachtet werden muss. Die Partialmodelle können auch noch im Nachhinein vermessen werden. Einfache Beispiele hierfür sind Anzahl an Komponenten, Programmzeilen oder Speicherplatz. Diese Kategorie von Komplexitätsmaßen ist vorwiegend durch das Verständnis der Quantität von Komplexität als Schwierigkeit der Beschreibung geprägt.

#### 2.1.3.2 Ziele der Komplexitätsmessung

Zur Abgrenzung von Komplexitätsmaßen ist es sinnvoll, das Ziel der Messungen zu beschreiben. So ist die Shannon-Einheit aus dem Abschnitt 2.1.2.1 vor allem von Shannons Arbeit bei den Bell Labs geprägt und eignet sich zur Beschreibung von Datenübertragungskapazitäten. (Kearney u. a. 1986) und (Davis und LeBlanc 1988) kritisieren, dass Softwarekomplexitätsmaße, die ohne Berücksichtigung des Ziels eingesetzt werden, Probleme wie demotivierte Entwickler verursachen. (Ameri u. a. 2008) beschreiben zwei Perspektiven auf ein Produkt in der Konstruktion, deren Ziele unterschiedliche Komplexitätsmaße benötigen.

Das Bewusstsein für das Ziel der Messung ist natürlich nicht nur für Komplexität, sondern auch für alle Arten von Messungen an Entwicklungsprozessen wichtig (Basili, Caldiera, und Rombach 1994), (Ojanen und Vuola 2006). Die im Folgenden beschriebenen Ziele orientieren sich jedoch nur an Literatur mit Bezug zu Maßzahlen, die Komplexität oder eine verwandte Eigenschaft wie Umfang oder Aufwand beschreiben.

##### 2.1.3.2.1 Designbewertung

Komplexität wird als ein Hilfsmittel zur Bewertung eines oder mehrerer Designs benutzt. Sie gibt dem Entwickler Hinweise zur Verbesserung des Produktes oder zur Strukturierung seiner Komponenten.

Insbesondere im Bereich des Software Engineering ist die Ermittlung einer optimalen, beherrschbaren Modulgröße eine wichtige Anwendung. (McCabe 1976) gibt zur Motivation seiner Softwaremetrik an, dass Module ab einem bestimmten Komplexitätsgrad nicht zu warten sind. (Henry und Kafura 1981) wenden ihr Maß auf den UNIX-Kernel an um zu zeigen, dass eine weitere Abstraktionsschicht auf einer Ebene eingebracht werden sollte. Komplexitätsmaße können helfen diejenigen Designelemente zu identifizieren, die überarbeitet oder ersetzt werden müssen.

Komplexität kann als Qualitätsmaß verwendet werden, um Designalternativen zu vergleichen. (Nam P. Suh 1999) definiert Komplexität als Maß der Unsicherheit bestimmte funktionale Anforderungen zu erreichen. Somit ist das Design mit der geringsten Komplexität das Beste. Diese Bewertung ist die Grundlage des von ihm entwickelten Axio-

matischen Designs (Nam P. Suh 1990). (J. D. Summers und Shah 2003) schlagen die Verwendung zur Bewertung von studentischen Übungsaufgaben vor.

#### 2.1.3.2.2 Projektmanagement

Projektmanager müssen Komplexität als wichtigen Teil des Entwicklungsprojektes einschätzen können. (Bashir und Thomson 1999a) identifizieren in ihrer Studie Produktkomplexität, Fähigkeiten des Teams und Managementkomplexität als wichtigste Projekt-Metriken in der Literatur. Projektkomplexität wird oftmals als Oberbegriff für verschiedene projektbezogenen Komplexitätsaspekte verwendet. Basierend auf den Arbeiten von (Baccarini 1996) und (Turner und Cochrane 1993) werden die Aspekte durch (T. M. Williams 1999) in eine Hierarchie aufgegliedert. Wie in Abbildung 6 dargestellt wird, werden je nach Art des Aspekts verschiedene Verständnisse der Quantität von Komplexität zugrunde gelegt.

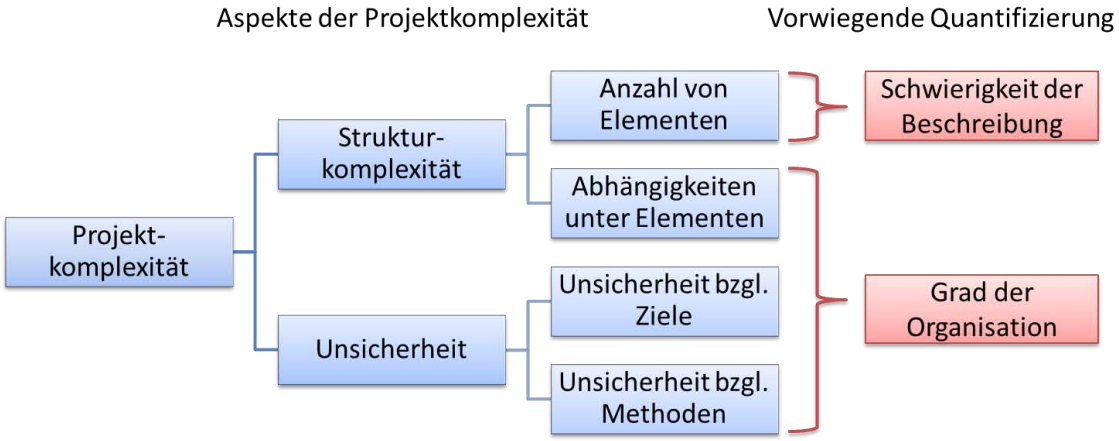


Abbildung 6: Hierarchie der Projektkomplexität
Quelle: Eigene Abbildung, eine Erweiterung von (T. M. Williams 1999)

Die Messung von Komplexität kann helfen Entwurfsprobleme vorherzusehen, die Projektaufgaben verzögern oder stoppen können. Ein wichtiger Indikator für aufkommende Probleme ist ein ausartendes Wachstum der Komplexität der Entwicklungsergebnisse und -prozesse. Komplexität ist eine Risikodimension in Projekten (Wallace 1999), welche einen wichtigen Einflussfaktor, selbst für kleine Projekte, darstellt (Wallance, Keil, und Rai 2004).

Komplexität wirkt sich auch auf die benötigte Zeit und Mittel für Entwicklungsprojekte aus. Das IT-Controlling benötigt die Abschätzung dieser Parameter um die Unternehmensressourcen auf die profitablen Projekte zu verteilen. Insbesondere bei Arbeiten auf Basis eines bestehenden Designs spielt Komplexität eine wichtige Rolle, da durch viele Abhängigkeiten auch kleinere Änderungen weitere Änderungen nötig machen und so-

mit in der Summe einen hohen Aufwand verursachen können. (Eckert, Earl, und Clarkson 2005) zeigen dies am Beispiel der Anpassung von Hubschraubermodellen.

#### 2.1.3.2.3 Produktivitätsmessung

Über dem Umfang eines einzelnen Projektes hinaus, ist die Produktivität eine Kennzahl, die als das Verhältnis zwischen erbrachter Leistung (Output) und eingesetzten Mitteln (Input) definiert wird. Der Input wird zumeist monetär gemessen, während der Wert des Outputs zunächst abstrakt ist. Komplexitätsmaße werden hier als Parameter für die Quantifizierung des Outputs genutzt, zum Beispiel in der Chipentwicklung (Häusler u. a. 2007), Konstruktion (Feldhusen und Koy 2002) und insbesondere Software ((Kearney u. a. 1986) gibt einen Überblick). Es ist sinnvoll Komplexitätsmaße auf Basis des „Schwierigkeit der Erzeugung"-Verständnisses zu wählen. Trotzdem werden oftmals Maße eingesetzt, die eher der „Schwierigkeit der Beschreibung" zuzuordnen sind und dazu führen können, dass Entwickler Produktmodelle künstlich aufblähen. Ein bekanntes Beispiel hierfür ist die Lines-of-Code-Metrik (LOC), die als Produktivitätsmaßstab eingesetzt zu komplexeren Entwürfen und schwerer zu wartenden Quellcode führen kann.

### 2.1.3.3 Komplexitätsmaße in der Produktentwicklung

In den vorherigen Abschnitten wurde bereits eine Reihe von Komplexitätsmaßen beschrieben. Diese Beschreibung des Stands der Technik der Komplexitätsmessung soll im Folgenden vervollständigt werden. Da Komplexitätsmaße in sehr vielen Publikationen auftreten, kann nur eine Auswahl der wichtigsten Vertreter getroffen werden. Insbesondere im Bereich des Software Engineering existieren eine hohe Anzahl an Komplexitätsmaßen, die im Detail im Buch von (Zuse 1990) beschrieben werden.

Um die Auflistung der Ansätze übersichtlich und kompakt zu gestalten werden nur kurz die wichtigsten Eigenschaften genannt und eine Kurzzusammenfassung gegeben. Die in den Abschnitten 2.1.2 (Quantifizierung) 2.1.3.1 (Gegenstand der Messung) und 2.1.3.2 (Ziele der Messung) definierten Kategorien werden zur Differenzierung der Einträge genutzt. Außerdem wird nach Möglichkeit der Kontext der Disziplin bzw. Domäne genannt. Die Ansätze sind chronologisch geordnet.

#### 2.1.3.3.1 Lines of Code (diverse Autoren)

- Disziplin: Software
- Gegenstand der Messung: Lösungsbeschreibung in Form von Quellcode
- Ziel der Messung: Designbewertung, Produktivitätsmessung
- Quantität: Schwierigkeit der Beschreibung

Misst Quellcode durch das Zählen von Codezeilen. In den meisten Fällen wird zuvor ein vordefinierter Syntaxstil angewendet und Kommentare entfernt. Zeilen können phy-

sikalisch als Textzeilen oder logisch als Befehle der Programmiersprache gezählt werden. In einigen Ansätzen werden Kommentare eingeschlossen, da sie ebenfalls als wesentlicher Teil des Quellcodes gesehen werden. Diese Metrik ist trotz der Nachteile, die in der Literatur (zum Beispiel (Fenton und Pfleeger 1996)(Salman und Dogru 2006)) ausgiebig diskutiert werden, eine gebräuchliche Größe in der Softwareentwicklung, da sie sehr einfach zu ermitteln ist.

#### 2.1.3.3.2 Zyklomatische Komplexität (McCabe 1976)

- Disziplin: Software
- Gegenstand der Messung: Lösungsbeschreibung in Form von eines Kontrollflussgraphen
- Ziel der Messung: Designbewertung
- Quantität: Schwierigkeit der Beschreibung

McCabe misst die Anzahl linear unabhängiger Pfade in einem Programmodul. Die Messung basiert auf dem Kontrollflussgraphen des Programms. Ein vollständig lineares Programm besitzt die Komplexität 1. Die Komplexität erhöht sich durch Kontrollelemente wie If-Then-Else-Anweisungen.

#### 2.1.3.3.3 Halstead-Metriken (Halstead 1977)

- Disziplin: Software
- Gegenstand der Messung: Lösungsbeschreibung in Form von Quellcode
- Ziel der Messung: Designbewertung, Projektmanagement
- Quantität: Schwierigkeit der Beschreibung

Die Halstead-Metriken umfassen fünf statische Softwaremetriken, die auf der Erfassung von Operatoren und Operanden im Quellcode basieren. Als Parameter der Berechnungsformeln treten die Anzahl verschiedener Operatoren oder Operanden und die absolute Anzahl von Operatoren und Operanden auf. Die fünf Metriken heißen Programmlänge, Programmvokabular, Umfang, Schwierigkeit und Aufwand.

#### 2.1.3.3.4 Verarbeitungskomplexität (Albrecht 1979)

- Disziplin: Software
- Gegenstand der Messung: Problembeschreibung in Form von Entwicklungsparameter
- Ziel der Messung: Produktivitätsmessung
- Quantität: Schwierigkeit der Erzeugung

Die Verarbeitungskomplexität ist ein Justierungsfaktor für den Function-Point-Wert. Sie wird über eine Menge von Entwicklungsparameter berechnet, die von den Entwicklern

mit 1 bis 5 eingestuft werden müssen. Beispiele dieser Parameter sind Menge der Kommunikation, Anforderung an Performanz und Wiederverwendbarkeit.

#### 2.1.3.3.5 Programmkomplexität (Oviedo 1980)

- Disziplin: Software
- Gegenstand der Messung: Lösungsbeschreibung
- Ziel der Messung: Designbewertung
- Quantität: Schwierigkeit der Beschreibung

Oviedos Komplexität wird in Kontrollflusskomplexität und Datenflusskomplexität unterschieden. Beide Werte werden an einem Graphen gemessen, dessen Knoten durch Aufteilung eines Programmes in Blöcke bestimmt werden. Das Ziel des Ansatzes ist die Verbesserung der Softwarequalität.

#### 2.1.3.3.6 Prozedurkomplexitätswert (Henry und Kafura 1981)

- Disziplin: Software
- Gegenstand der Messung: Lösungsbeschreibung
- Ziel der Messung: Designbewertung
- Quantität: Schwierigkeit der Beschreibung

Diese Maß basiert auf der Messung von Informationsflüssen (Fan-in und Fan-out) zwischen Systemkomponenten. Ähnlich wie die Verarbeitungskomplexität wird die Messung des Informationsflusses als Anpassungsfaktor für ein Maß der Quellcodelänge verwendet. Für die Quellcodelänge kann die Halstead Programmlänge oder Lines of Code genutzt werden. Der Wert der Softwaremetrik ergibt sich durch die folgende Formel:

$$\text{Softwarewert} = \text{Quellcodelänge} * \underbrace{(\text{Fan-in} * \text{Fan-Out})^2}_{\text{Prozedurkomplexität}}$$

#### 2.1.3.3.7 Projektkomplexität (Griffin 1993)

- Disziplin: Engineering
- Gegenstand der Messung: Entwicklungsprozess
- Ziel der Messung: Produktivitätsmessung
- Quantität: Schwierigkeit der Erzeugung

Vor dem Hintergrund der Messung eines Produktentwicklungszyklus wird durch diese Komplexität beschrieben, wie schwierig und zeitaufwändig die Entwicklung sein wird durch die schiere Größe des Projektes. Die bezieht sich vor allem auf Größen wie die Anzahl an Projektbeteiligten, Arbeitsergebnisse, Projektphasen und Meilensteine. Der Parameter „technische Spezialisierungen“ entspricht der Anzahl beteiligter Disziplinen.

Der Komplexitätswert ist eine nicht genauer festgelegte Funktion, die wie folgt aufgebaut ist.

$$\text{Projektkomplexität} = f(\text{Anzahl Produktfunktionen, Anzahl beteiligte Disziplinen})$$

#### 2.1.3.3.8 Projektunsicherheit (Turner und Cochrane 1993)

- Disziplin: Allgemein
- Gegenstand der Messung: Problembeschreibung
- Ziel der Messung: Projektmanagement
- Quantität: Schwierigkeit der Erzeugung

Die Projektunsicherheit ist kein echtes Komplexitätsmaß aber ein wichtiger Aspekt für Projektkomplexität und wird daher hier aufgeführt. Die Projektunsicherheit wird als Parameter für die Projektkomplexität durch (T. M. Williams 1999) eingebunden (siehe Abbildung 6).

#### 2.1.3.3.9 Projektstrukturkomplexität (Baccarini 1996)

- Disziplin: Bauindustrie
- Gegenstand der Messung: Entwicklungsprozess
- Ziel der Messung: Projektmanagement
- Quantität: Schwierigkeit der Beschreibung und Grad der Organisation

Aus Sicht der Autoren besteht Projektkomplexität aus vielen miteinander in Beziehung stehenden Elementen. Diese kann in organisationale und technologische Komplexität unterschieden werden. Die organisationale Komplexität ist die Anzahl verschiedener Elemente im Projekt wie Aufgaben und Teile und entspricht der Sicht von Komplexität als Schwierigkeit der Beschreibung. Die technologische Komplexität ist der Grad der Verknüpfung zwischen diesen Komponenten und entspricht dem Verständnis als Grad der Organisation.

#### 2.1.3.3.10Komplexitätsfaktor (Jacome und Lapinskii 1997)

- Disziplin: Chipentwicklung
- Gegenstand der Messung: Entwicklungsprozess
- Ziel der Messung: Kostenabschätzung
- Quantität: Schwierigkeit der Erzeugung

Dieses Komplexitätsmaß ist Teil eines prozessorientierten Modells für die Kostenabschätzung, welches Größe, Komplexität und Produktivität als Parameter verwendet. Der Komplexitätsfaktor ist definiert als die relative Komplexität einer Aktivität im Kontext eines bestimmten Bausteins, verglichen zu einer durchschnittlichen Aktivität desselben Typs in derselben Umgebung. Bei der Komplexität handelt es sich also um einen Justierungsfaktor wie bei den Function Points.

#### 2.1.3.3.11Produktkomplexität (Bashir und Thomson 1999b)

- Disziplin: Engineering
- Gegenstand der Messung: Lösungsbeschreibung
- Ziel der Messung: Projektmanagement
- Quantität: Schwierigkeit der Erzeugung, Schwierigkeit der Beschreibung

Dieses Maß von Produktkomplexität basiert auf einer funktionalen Dekomposition eines Produktes in einem Hierarchiebaum. Für die Nachvollziehbarkeit der Dekomposition wird ein Vorgehensmodell entwickelt. Innerhalb des Baumes bezeichnet die Variable $F_j$ die Anzahl der Funktionen auf dem Level $j$ und $l$ bezeichnet die Baumtiefe. Die Produktkomplexität wird dann wie folgt berechnet:

$$Produktkomplexität = \sum_{j=1}^{l} F_j j$$

#### 2.1.3.3.12Axiomatische Designkomplexität (Nam P. Suh 1999) (Nam Pyo Suh 2001)

- Disziplin: Engineering
- Gegenstand der Messung: Lösungsbeschreibung
- Ziel der Messung: Designbewertung
- Quantität: Grad der Organisation

Der Autor definiert Komplexität als *„ein Maß der Unsicherheit die spezifizierten funktionalen Anforderungen zu erreichen“*. Je mehr Information ein Entwurf benötigt um die funktionalen Anforderungen zu erfüllen, desto größer ist die Komplexität. Die Information kann minimiert werden, in dem eine Anforderung jeweils nur einem Element in anderen Partialmodellen zugeordnet wird.

#### 2.1.3.3.13Produktkomplexität (Dierneder und Scheidl 2001)

- Disziplin: Engineering
- Gegenstand der Messung: Lösungsbeschreibung
- Ziel der Messung: Designbewertung
- Quantität: Schwierigkeit der Erzeugung und Grad der Organisation

Die Produktkomplexität ist eine Aggregation aus drei verschiedenen Komplexitätsmaßen: Eine funktionale Produktkomplexität basierend auf funktionalen Verbindungen, eine technische Produktkomplexität, die aus den Beziehungen zwischen funktionalen Anforderungen und technischen Designparametern abgeleitet wird, und der Produktkomplexität der Verlässlichkeit, die auf der Komplexität von Suh beruht.

#### 2.1.3.3.14Produktkomplexität (Feldhusen und Koy 2002)

- Disziplin: Engineering
- Gegenstand der Messung: Lösungsbeschreibung
- Ziel der Messung: Produktivität
- Quantität: Schwierigkeit der Erzeugung

Die Autoren stellen ein computerunterstütztes System zur Produktivitätsmessung in der Konstruktion vor. Das Ergebnis einer Aufgabe wird durch einen Komplexitätsindex normalisiert, so dass die Ergebnisse verschiedener Aufgaben miteinander verglichen werden können. Hierbei wird der Aufwand der Konstruktion eines Bauteils nicht in Stunden sondern in „Aufwandspunkten“ gemessen. Die Verwendung eines Komplexitätsindexes besitzt einige Ähnlichkeit zu dem in Abschnitt 5.2 beschriebenen Ansatz eines Referenzproduktmodells. Jedoch bezieht sich dieser Index auf einzelne Komponenten und muss von den Entwicklern subjektiv abgeschätzt werden.

#### 2.1.3.3.15Informationskomplexität (Eckert, Earl, und Clarkson 2005)

- Disziplin: Engineering
- Gegenstand der Messung: Problembeschreibung, Lösungsbeschreibung und Entwicklungsprozess
- Ziel der Messung: Designbewertung und Projektmanagement
- Quantität: Grad der Organisation

Die Autoren stellen kein konkretes Komplexitätsmaß aber ein Modell der Komplexität vor: Der Entwicklungsprozess wird in die drei Ebenen Hintergrund, Beschreibungen und Aktionen aufgeteilt. Dies entspricht im Groben den Kategorien des Gegenstands der Messung in 2.1.3.1. Komplexität wird in Form von Abhängigkeiten zwischen diesen Ebenen beschrieben, welche wesentlich den Aufwand von Änderungen bestimmen.

#### 2.1.3.3.16Komplexität des Entwurfsartefakts (Häusler u. a. 2007)

- Disziplin: Chipentwicklung
- Gegenstand der Messung: Problembeschreibung, Lösungsbeschreibung
- Ziel der Messung: Produktivität
- Quantität: Schwierigkeit der Beschreibung

Die Komplexität einer Produktentwicklung wird basierend auf den Anforderungen definiert und ist über den Entwicklungsverlauf konstant. Die tatsächliche Messung geschieht jedoch auf Basis der Partialmodelle der Lösungsbeschreibung. Für die Disziplin der Chipentwicklung wird eine Menge von Indikatoren (zum Beispiel Chipgröße, Spannung und Leistung) identifiziert, die Komplexität repräsentieren und durch messbare Eigenschaften des Produktes abgebildet werden können.

#### 2.1.3.3.17Funktionskomplexität (Zickert und Beck 2010)

- Disziplin: Software
- Gegenstand der Messung: Lösungsbeschreibung
- Ziel der Messung: Projektmanagement
- Quantität: Schwierigkeit der Erzeugung

Dieser Ansatz dient der Aufwandsschätzung für das Projektmanagement. Es werden Semantiken aus der KAOS-Methode (Dardenne, van Lamsweerde, und Fickas 1993) auf bestimmte Strukturen abgebildet. Diesen sind feste Komplexitäten zugeordnet, die als Punkte im Sinne der Function-Point-Analyse gezählt werden. Die Gesamtkomplexität einer Funktion ergibt sich aus der Komplexität der mit ihr verknüpften Strukturen.

#### 2.1.3.3.18Prozessstrukturkomplexität (Kreimeyer 2010)

- Disziplin: Engineering
- Gegenstand der Messung: Entwicklungsprozess
- Ziel der Messung: Projektmanagement
- Quantität: Schwierigkeit der Beschreibung, Grad der Organisation

Diese Doktorarbeit betrachtet Prozesse in Entwicklungsprojekten und beschreibt eine große Anzahl von strukturellen Maßen, die die Komplexität dieser Prozesse messen. Auf Basis einer vereinheitlichenden Graphdarstellung werden Indikatoren für Schwächen in der Gestaltung des Entwicklungsprozesses gegeben. In diesem Kontext wird Komplexität nicht als direkt messbare Eigenschaft betrachtet, sondern als einen Aspekt der Produktentwicklung. Es wird also kein Komplexitätsmaß beschrieben, sondern ein Analyse-Framework, das komplexitätsbezogene Messungen durchführt.

## 2.2 Anforderungen an ein Komplexitätsmaß für Produktmodelle

Im vorherigen Abschnitt wurde die Vielfältigkeit des Komplexitätsbegriffes in der bestehenden Literatur verdeutlicht. Die meisten Ansätze, die sich auf „Komplexität“ beziehen, sind bei genauerer Betrachtung nicht miteinander vergleichbar. In der Literatur finden sich aber trotzdem oftmals Vergleiche zu anderen „Komplexitäten“ (zum Beispiel (Lindemann, Maurer, und Braun 2008, 3)), deren Beurteilung durch die jeweilige Sicht der Autoren geprägt ist. Wenn von dem Problem „Komplexität“ innerhalb einer Entwicklung allgemein gesprochen wird, kann dies entsprechend sehr Unterschiedliches bedeuten. Die folgende Liste kann nicht vollständig sein und beschreibt nur die häufigsten Interpretationen:

- Ein Projekt unterscheidet sich durch die Art oder Anzahl der Systeme von Vorgängerprojekten und erfordert ein angepasstes Vorgehen bei der Entwicklung.

Zum Bespiel führt (Hubbert 2003) Probleme bei der Entwicklung der Autoelektronik in der Mercedes E-Serie auf die „Komplexität der Systeme“ zurück.

- Es handelt sich um ein sehr großes oder auch sehr langfristiges Projekt. Dies ist insbesondere in der englischen Literatur eine Interpretationsmöglichkeit, da „Complexity“ direkt mit „Umfang“ übersetzt werden kann.
- Die Implementierung gestaltet sich aufgrund der Nichtvertrautheit der Projektmitarbeiter mit eingesetzten neuen Techniken als schwierig.
- Die Anforderungen sind schwer zu formulieren, da verschiedene Projektinteressenträger zunächst ein gemeinsames Verständnis und eine Sprache finden müssen um sich über Sachinhalte zu einigen.
- Oftmals ist die Verwendung des Wortes „Komplexität“ sogar auch ein Ausdruck dafür, dass keine präzisere Formulierung oder ein tieferes Verständnis für auftretende Probleme im Projekt vorhanden ist.

Es zeigt sich somit das Problem, dass die Verwendung des Begriffes „Komplexität“ für wissenschaftliche Zwecke zu ungenau sein kann. Daher soll in diesem Abschnitt der für diese Arbeit gültige Komplexitätsbegriff und dessen Grundlagen explizit herausgearbeitet werden. Die Kategorisierung im vorherigen Abschnitt dient hierzu als Werkzeug. Diese Definition bildet die Basis für die weitere Verwendung in dieser Arbeit und besitzt natürlich keine Allgemeingültigkeit, sondern entspricht in seiner Ausprägung dem in der Einleitung genannten Ziel der Komplexität als Ausgangswert für eine Produktivitätsbewertung. Des Weiteren werden wünschenswerte Eigenschaften aus konzeptioneller und praktischer Sicht beschrieben. Die Beschreibung der Anforderungen an das Komplexitätsmaß ist somit direkt mit der Definition des Begriffs verbunden. Am Ende des Abschnitts werden die Anforderungen in einem Katalog zusammengefasst.

### 2.2.1 Komplexitätsbegriff für die Leistungsmessung

Die zentrale Motivation des in dieser Arbeit vorgestellten Ansatzes ist, wie in der Einleitung beschrieben, die Leistungsmessung in der Produktentwicklung. Der Komplexitätsbegriff dieser Arbeit soll eine entsprechende Ausgestaltung besitzen, so dass er für die Leistungsmessung sinnvoll eingesetzt werden kann. Die Anforderungen, die sich aus diesem konzeptionellen Rahmen ergeben, beziehen sich insbesondere auf den *Komplexitätsbegriff* und werden im Folgenden ausgearbeitet. Des Weiteren werden Anforderungen entwickelt, die sich vor allem auf die praktische Anwendbarkeit und somit auf ein entsprechendes *Komplexitätsmaß* beziehen. Ein theoretisch und konzeptionell ausgefeiltes Komplexitätsverständnis kann nicht die Ziele der Leistungsmessung erreichen, wenn dessen Messung in der Praxis nicht praktikabel ist.

#### 2.2.1.1 Konzeptionelle Anforderungen

##### 2.2.1.1.1 Quantifizierung als Schwierigkeit der Erzeugung

Im Zusammenhang mit Komplexität steht in der Literatur sehr häufig der Aufwand. (Salman und Dogru 2006) verbinden Komplexität und Aufwand in ihrer Untersuchung von Vorhersagemodellen. (Bashir und Thomson 1999b) definieren Komplexität in Anlehnung an (Norden 1964) als proportional zur Anzahl an Mannwochen zur Fertigstellung einer Aufgabe. In der Function-Point-Analyse (Albrecht 1979) wird Komplexität als ein Gewichtungsfaktor benutzt, aber noch manuell abgeschätzt (Abran und Robillard 1994). Als eine Erweiterung hierzu leiten (Zickert und Beck 2010) Function-Point-Komplexität durch ein Mapping von Graphstrukturen einer Anforderungsmodellierung ab. Dies ist außerdem ein Brückenschlag zur Berücksichtigung der Relationen, was in der folgenden Anforderung diskutiert wird. Allerdings begründen die Autoren nicht, warum zum Beispiel eine Art von Verknüpfung mit 2 und eine andere mit 1 gewichtet wird.

Neben der Quantifizierung von Produkten für die Leistungsmessung soll an dieser Stelle außerdem die Bewertung für die Projektfortschrittskontrolle diskutiert werden. In diesem Bereich kann eine Quantifizierung von Komplexität einen erheblichen Beitrag zur Projektkontrolle leisten, wie durch Burghardt deutlich wird:

*„Die Sachfortschrittskontrolle stellt für den Entwickler und Projektleiter wohl die wichtigste Kontrollaufgabe dar; sie ist aber auch die schwierigste. Da es normalerweise keine unmittelbaren Messgrößen für den Sachfortschritt gibt, muss auf Ersatzgrößen zurückgegriffen werden, die nur einen indirekten Bezug haben und deshalb nur eingeschränkt eine Aussage auf den Sachfortschritt zulassen.“ (Burghardt 2006, 18)*

Komplexität kann als eine solche indirekte Messgröße verstanden werden, wenn man deren Quantifizierung als Schwierigkeit der Erzeugung versteht. Sie erlaubt ebenfalls keine exakten Aussagen zum Fortschritt, kann aber bei entsprechender Gestaltung ausreichend für die Zwecke der Sachfortschrittskontrolle sein.

Insgesamt ist es somit im Sinne der in der Motivation beschriebenen Produktivitätsmessung sinnvoll, die Quantifizierung von Komplexität als Schwierigkeit der Erzeugung gemäß Abschnitt 2.1.2 zu beschreiben. Durch diese Sichtweise kann Komplexität sinnvoll eingesetzt werden, um zum Beispiel zwei unterschiedliche Entwicklungssysteme miteinander zu vergleichen. Auch eine Bewertung von Verzögerungen, wie in dem in der Motivation genannten Beispiel des angepassten Softwareprojektes, wird hierdurch ermöglicht. Gegenstand der Bewertung ist in diesem Fall das Artefakt.

#### 2.2.1.1.2 Bewertung der Semantik von Relationen

Die Literaturübersicht zur Komplexität hebt außerdem immer wieder die Rolle der Beziehungen, Verknüpfungen oder Relationen hervor. Diese synonymen Begriffe finden sich in fast jeder Publikation bezüglich Komplexität. Die Komplexität einer Aufgabe, wird durch die Beziehungen der beteiligten Elemente bestimmt (Campbell 1988). (Bashir und Thomson 1999b) beschreiben dies durch die Dekompositionsbeziehung von Funktionen. Die Projektstrukturkomplexität von (T. M. Williams 1999) basiert auf den Beziehungen der Projektelemente untereinander. Interessant ist hierbei, dass durch „Projektelemente“ auch der Aspekt der partialmodellübergreifenden Beziehungen inbegriffen ist. (Eckert, Earl, und Clarkson 2005) charakterisieren Komplexität ebenfalls durch Verknüpfungen. (El-Haik und Yang 1999) definieren Komplexität als eine Eigenschaft eines Objekts, welches mit vielen miteinander verflochtenen Elementen, Aspekten und Attributen verbunden ist. Die Komplexität soll also über die Semantik von Relationen bewertet werden, da diese entscheidenden Einfluss auf die Schwierigkeit der Erzeugung haben.

#### 2.2.1.1.3 Domänenunabhängigkeit

Gegenstand der Messungen sind die Entwicklungsergebnisse. Jedem Entwicklungsartefakt sollte ein Wert zugeordnet werden können. Dies stellt eine Herausforderung für einen domänenübergreifenden Ansatz dar, da von Art und Struktur unterschiedliche Produktmodelle miteinander verglichen werden sollen. In vielen Entwicklungsprozessen werden durch unterschiedliche Sichten auf das Produkt außerdem mehrere parallele Produktmodelle definiert. Diese als Partialmodelle bezeichneten Abbilder des künftigen Produktes basieren auf unterschiedlichen Metamodellen, wodurch das Komplexitätsmaß also keine Voraussetzungen an diese Modelle stellen kann. Um für die Analyse auf alle Informationen zugreifen zu können, muss daher eine Lingua franca definiert werden, in der sich diese Modelle beziehungsweise Metamodelle beschreiben lassen.

Neben den Beziehungen der Artefakte untereinander können auch Eigenschaften des Entwicklungssystems Einfluss auf die Komplexität haben. Wie in Abschnitt 2.1.2.2 beschrieben, beeinflussen die Eigenschaften dieses Systems wie verwendete Werkzeuge, beteiligte Personen oder Methoden entscheidend den Aufwand. Zum Beispiel ist in vielen Projekten die Anzahl und Art der Stakeholder ein wesentlicher Faktor für den benötigten Aufwand. Diesem Umstand muss die Komplexitätsbewertung Rechnung tragen. Die Schwierigkeit an diesen Einflüssen ist, ähnlich wie bei den Artefakten, dass je nach Umgebung andere Faktoren ausschlaggebend sein können. Dies kann sich nicht nur zwischen Domänen sondern auch zwischen Unternehmen oder Abteilungen unterscheiden. Es kann also wieder von keinem festen Metamodell ausgegangen werden.

#### 2.2.1.2 Praktische Anforderungen

Die erfolgreiche Anwendung eines Komplexitätsmaßes hängt nicht nur von einer fundierten theoretischen Basis ab, wie sie in diesem Kapitel herausgearbeitet wurde, sondern auch von praktischen Aspekten ihrer Implementierung ab. Dieser Abschnitt behandelt Anforderungen, die sich auf die praktische Anwendung und die Akzeptanz von Komplexitätsmaßen beziehen. Für allgemeinere Betrachtungen zur Anwendung von Maßen im Entwicklungsmanagement wird auf (Ojanen und Vuola 2006), (Basili, Caldiera, und Rombach 1994) verwiesen.

##### 2.2.1.2.1 Minimierung des Messaufwandes

Das Sammeln und Auswerten von Daten für die Messungen ist aufwendig und kann keine hohe Priorität in engen Projektzeitplänen besitzen. Entwickler, die mehr als eine minimale Menge an Zeit für Messungen aufwenden müssen, werden sich von ihrer eigentlichen Arbeit abgelenkt fühlen und daher diese Metriken ablehnen. Insbesondere gilt dies für schätzbasierte Verfahren, die eine sinnvolle Abwägung zwischen ausreichender Datenmenge und -qualität und Zeitaufwand für die Entwickler finden müssen. Zum Beispiel beruht der Anpassungswert der Function-Point-Metriken auf Schätztabellen, die auf jedes Ein- und Ausgabeelement angewendet werden müssen. Bei Zeitknappheit, wie sie leider häufig gegeben ist, werden solche Schätzungen nicht mehr gründlich durchgeführt sondern „über den Daumen gepeilt“. Daher ist es wichtig, dass das Verfahren mit möglichst wenig manuellem Aufwand verbunden ist.

##### 2.2.1.2.2 Individuelles Komplexitätsmaß

Jedes Projekt und jede Projektumgebung ist verschieden. Komplexitätsmodelle, die in einer bestimmten Umgebung entwickelt wurden, basieren auf Faktoren, die in dieser Umgebung variabel sind. Sie ignorieren hingegen Faktoren, die in dieser Umgebung konstant sind. Jedoch können diese konstanten Faktoren in anderen Unternehmen, Märkten oder Anwendungsdomänen variabel sein. Zum Beispiel kann bei Produkten, die international vertrieben werden, die Komplexität von den Mengen an Märkten mit unterschiedlichen Eigenheiten oder gesetzlichen Vorschriften abhängen. Ein Komplexitätsmaß, das auf der Entwicklung eines reinen Inlandsproduktes basiert, kann dem gegenüber blind sein. Die Autoren (DeMarco 1983, 154f)(Bailey und Basili 1981) sehen dies als Grund, warum Kostenmodelle in ihren ursprünglichen Umgebungen genaue Vorhersagen treffen konnten, aber bei anderen Firmen scheiterten. Dies gilt ebenso für Komplexitätsmodelle. Daher müssen Kostenmaße jeweils angepasst werden und oftmals ist es notwendig ein „maßgeschneidertes“ Maß von Grund auf zu entwickeln um eine ausreichende Genauigkeit sicherzustellen.

#### 2.2.1.2.3 Partialmodellübergreifende Beschreibung

Wie bereits in Abschnitt 2.2.1.1.3 beschrieben, basieren Artefakte oftmals auf verschiedenen Partialmodellen, die eine Domänensicht auf das Produkt darstellen. Insbesondere die Beziehungen zwischen diesen Repräsentationen eines Produktes spielen eine wichtige Rolle in der Komplexität. Eine Komplexitätsanalyse muss also domänenübergreifend sein und zusätzlich zur Modellierung dieser Domänen auch diese partialmodellübergreifenden Beziehungen zwischen Artefakten unterschiedlichen Typs beschreiben können. Bereits in einer einzelnen Domäne können verschiedene Partialmodelle auftreten. So werden in der Softwareentwicklung unterschiedliche Programmiersprachen für verschiedene Aspekte desselben Programmes verwendet.

#### 2.2.1.2.4 Greifbare Komplexitätseinheit

Wie in diesem Kapitel gezeigt wurde, gibt es aufgrund der abstrakten Natur der Komplexität unterschiedliche Verständnisse. Es ist daher oftmals sinnvoll den Begriff „Komplexität“ ganz zu vermeiden und sich stattdessen auf eine greifbare Einheit zu beziehen. Da das Komplexitätsmaß dieser Arbeit im Sinne von Schwierigkeit der Erzeugung definiert werden soll und die Produktivitätsbewertung das primäre Ziel ist, ist es sinnvoll Aufwandseinheiten wie Kosten und Mannstunden zu verwenden. Dies überdeckt sich mit der Anforderung der zielorientierten Messung.

### 2.2.2 Komplexitätsdefinition

Die formulierten Anforderungen liefern die Grundlagen für die Erstellung einer Komplexitätsdefinition, die für die weitere Arbeit als Basis dient. Die wichtigsten Anwendungsaspekte sind das Verständnis als Schwierigkeit der Beschreibung sowie die domänenübergreifende Erfassung von Relationen von Artefakten, die sich aus den Zielen der Produktivtitätsbewertung ergeben.

In der Definition von Komplexität für diese Arbeit sollen der Aufwand und die Relationen miteinander in Beziehung gesetzt werden. Es ist offensichtlich, dass diese beiden Aspekte voneinander abhängen. Eine erste Idee ist es, den Verknüpfungsgrad eines Artefakts proportional mit seiner Komplexität und seinem Aufwand zur Erzeugung, also dem Konstruktionsaufwand zu setzen. Hier zeigt sich jedoch das Problem, dass nicht nur eine Art von Relation wie zum Beispiel bei (Bashir und Thomson 1999b) betrachtet wird, sondern aufgrund des domänenübergreifenden Gedankens verschiedene Arten betrachtet werden. Intuitiv ist erkennbar, dass nicht jede Relation denselben Einfluss auf die Komplexität des Artefakts besitzt; Eine Beziehung „implementiert Schnittstelle“ ist wesentlich anders zu bewerten als „ruft Methode auf“. Es müssen Semantiken jeweils anders gewichtet werden, wie dies auch durch (Zickert und Beck 2010) vorgenommen wird. Es bleibt die Frage bestehen, wie die Gewichtung vorgenommen wird und hiermit

bietet sich die Verknüpfung zum Aufwand an. Insgesamt sei somit für diese Arbeit bezogen auf den Bereich der Produktentwicklung definiert:

**Komplexität ist die durch den Aufwand gewichtete Semantik einer Beziehung.**

Für die Ziele dieser Arbeit ist dies jedoch noch eine unvollständige Festlegung, da sich die Definition nicht auf Entwicklungsartefakte bezieht. Es muss die Verbindung von den Relationen eines konkreten Artefaktes zu seinen Relationen gezogen werden. Da die Komplexität mit der Anzahl der Relationen steigt, spricht dies für eine Aggregation. Deswegen wird die Definition bezogen auf Artefakte wie folgt gestaltet:

**Die Komplexität eines Artefaktes ist die Summe der Komplexitäten der Semantiken seiner Beziehungen.**

Da zur Abbildung der Relationen in einem Modell weitere Elemente abgebildet werden müssen, die nicht in einem Entwicklungsprozess entstehen (zum Beispiel Stakeholder oder Werkzeuge), benutzen wir eine Schlussfolgerung aus der oben gegebenen Definition um eine Abgrenzung zu ziehen.

**Artefakte sind Modellelemente, denen eine Komplexität zugeordnet werden kann.**

### 2.2.3 Anforderungskatalog

Mit den Anforderungen und den Basisdefinitionen ist die theoretische Basis für diese Arbeit aufgestellt. Zur Ausarbeitung eines Ansatzes für ein Komplexitätsmaß werden die Anforderungen in folgender Tabelle 2 nochmal strukturiert, die in diesem Kapitel erhoben wurden. Sie werden in konzeptionelle Anforderungen (KA) und praktische Anforderungen (PA) unterschieden. Konzeptionelle Anforderungen ergeben sich aus der grundlegenden Motivation dieser Arbeit wie in Abschnitt 2.2.1.1 dargestellt. Praktische Anforderungen ergeben sich aus generellen Rahmenbedingungen für eine effektive Anwendung von Messansätzen in der Entwicklung und sind in Abschnitt 2.2.1.2 erarbeitet worden.

Tabelle 2: Konzeptionelle und praktische Anforderungen an das Komplexitätsmaß

| ID | Beschreibung | Siehe |
|---|---|---|
| KA1 | **Quantifizierung als Schwierigkeit der Erzeugung**<br>Die Quantifizierung von Komplexität erfolgt als Schwierigkeit der Erzeugung gemäß Abschnitt 2.1.2 um die Ziele der Leistungsmessung zu unterstützen. Komplexität (im Sinne der quantitativen Bewertung in dieser Arbeit) ist eine Eigenschaft der Entwicklungsartefakte und nicht von anderen Projektelementen abhängig. | 2.2.1.1.1 |
| KA2 | **Bewertung der Semantik von Relationen**<br>Die Komplexität wird über die Semantik von Relationen bewertet, da diese entscheidenden Einfluss auf die Schwierigkeit der Erzeugung besitzen. | 2.2.1.1.2 |
| KA3 | **Domänenunabhängigkeit**<br>Der Ansatz soll in der Entwicklung mit mehreren beteiligten Domänen eingesetzt werden. Daher sollen die zugrundeliegenden Komplexitätsbewertungen nicht an eine bestimmte Domäne oder Modellierung gebunden sein. | 2.2.1.1.3 |
| PA1 | **Minimierung des Messaufwandes**<br>Um den Entwickler in seiner Arbeit nicht zu beeinträchtigen soll der Messaufwand so gering wie möglich gehalten werden. | 2.2.1.2.1 |
| PA2 | **Individuelles Komplexitätsmaß**<br>Der Einfluss der Semantik hängt von der Entwicklungsumgebung ab und kann zwischen verschiedenen Umgebungen stark schwanken. Daher ist es explizit kein Ziel ein allgemeingültiges Maß zu entwickeln, sondern ein Werkzeug zur Definition eines umgebungsbezogenen Maßes zu bieten. | 2.2.1.2.2 |
| PA3 | **Partialmodellübergreifende Beschreibung**<br>Die Beschreibung und Bewertung von Komplexität sollte auch Relationen berücksichtigen, die verschiedene Partialmodelle der Entwicklung miteinander verbinden. | 2.2.1.2.3 |
| PA4 | **Greifbare Komplexitätseinheit**<br>Der abstrakte Charakter des Komplexitätsbegriffes erschwert die Zuordnung zu den Zielen der Messung. Daher soll dieser Bezug über eine Ersatzgröße hergestellt werden. | 2.2.1.2.4 |

# 3 Verwandte Ansätze der Leistungsmessung in der Produktentwicklung

In diesem Abschnitt wird auf Ansätze eingegangen, die das Management von Entwicklungsprojekten ebenfalls durch Bereitstellung von quantitativen Zahlen unterstützen aber nicht zentral auf einem Komplexitätsbegriff aufsetzen. Dies dient als Grundlage für die Bearbeitung der konzeptionellen Fragestellung. Die Bezüge des in dieser Arbeit entwickelten Ansatzes zu diesen Ansätzen wird in Abschnitt 5.9.1 beschrieben.

## 3.1 Produktivitäts- und Effizienzmessung

Die betriebswirtschaftliche Theorie zur Effizienz- und Produktivitätsanalyse liefert die Grundlage zur Bewertung von produzierenden Unternehmen in der Ökonometrie. Obwohl die Theorie von der Produktion von Sachgütern abgeleitet ist, beschränkt sie sich nicht nur auf diesen Bereich, sondern kann auch auf Dienstleistungen angewendet werden. Des Weiteren können nicht nur einzelne Unternehmen betrachtet werden, sondern auch alle produzierenden Entitäten wie Unternehmensteile, Werke, Maschinen oder Volkswirtschaften. Im Kontext dieser Arbeit sind als Gegenstand der Betrachtung insbesondere Entwicklungsumgebungen interessant. In der Theorie wird häufig der Begriff „decision making unit“ für diese Entitäten benutzt, der aber im Folgenden aufgrund der Verwechselungsgefahr vermieden wird. Stattdessen wird im Allgemeinen von Firmen gesprochen. (vgl. (Coelli u. a. 2005, 1).

Zunächst werden die Grundlagen des Produktivitäts- und Effizienzbegriffes beschrieben. Diese beiden Kennzahlen sind vom besonderen Interesse für das Management von Entwicklungsprozessen. Darauffolgend werden zwei wichtige Methoden zur Bestimmung der technischen Effizienz vorgestellt, die Data Envelopment Analysis (DEA) und die Stochastic Frontier Analysis (SFA). Die konzeptionellen und methodischen Unterschiede zu dem in dieser Arbeit dargestellten Ansatz werden jeweils separat diskutiert.

### 3.1.1 Produktivität und Effizienz

Unter Produktivität wird im Allgemeinen das Verhältnis zwischen Ausbringungsmenge (Output) und Einsatzmenge (Input) eines Unternehmens verstanden. Der Input besteht aus allen Produktionsfaktoren, die ein Unternehmen aufwendet um einen Output zu erzielen. Hierzu gehören im klassischen Sinn Rohstoffe, Materialien und Gehälter. Im Rahmen der Produktentwicklung sind vor allem die Gehälter und andere Personalkosten wie Schulungen, Lizenzkosten für Softwarewerkzeuge und Hardwarekosten für Entwicklungsrechner wie Simulatoren ausschlaggebend (Häusler u. a. 2007). Materialver-

brauch für die Entwicklerbüros und Prototypen spielen in der Regel eine geringere Rolle. Der Output ist durch den Wert der erzeugten Güter gegeben. Die Produktentwicklung erzeugt jedoch zunächst nur Produktmodelle und der monetäre Nutzen dieser Modelle ist in der Regel nur mittelbar durch den Verkauf von Produktinstanzen gegeben. Jedoch sind auch andere Geschäftsmodelle möglich. So verkaufen Open-Source-Software-Hersteller ihre Beratungsleistung bezüglich ihres Produktes oder komplementäre Produkte, während das Kernprodukt frei zur Verfügung steht. An diesem Beispiel lässt sich die Schwierigkeit der Bewertung des Outputs in der Produktentwicklung veranschaulichen. Die unmittelbare Bewertung der Produktmodelle durch Kennzahlen ist hierbei eine gängige Methode und auch Komplexitätskennzahlen werden hierzu verwendet (siehe Abschnitt 2.1.3.2.3). Auch der in dieser Arbeit entwickelte Ansatz soll im Rahmen einer Produktivitätskennzahl genutzt werden.

Produktivität und Effizienz werden häufig im Sprachgebrauch synonym verwendet. Jedoch drücken sie zwei wesentlich unterschiedliche Aspekte der Leistungsbewertung aus. Der Unterschied soll anhand der Abbildung 7 erläutert werden.

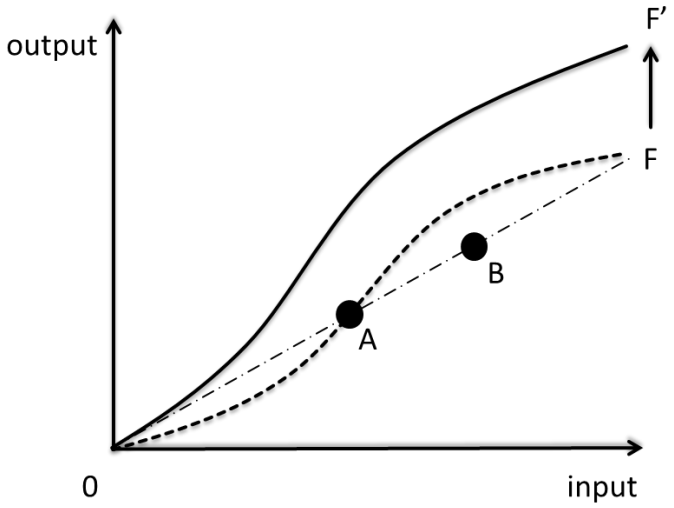


Abbildung 7: Produktivität und Effizienz
Quelle: Eigene Abbildung in Anlehnng an (Coelli u. a. 2005)

Die X-Achse beschreibt den Input und die Y-Achse den Output. Die beiden Firmen A und B besitzen das gleiche Verhältnis von Input und Output und somit die gleiche Produktivität. Die gestrichelte Linie F beschreibt den maximal möglichen Output für einen gegebenen Input (Produktivitätsgrenze). Dieser hängt von den technischen Möglichkeiten und Skaleneffekten ab. Da Firma A auf dieser Linie liegt, wird es als technisch effizient beschrieben. Firma B ist hingegen ineffizient, da sie theoretisch mehr Output ohne zusätzlichen Input erzielen könnte. Durch technische Erneuerungen wie zum Beispiel ein neu verfügbares Entwicklungswerkzeug kann die Produktivitätsgrenze nach oben

verschoben werden, wodurch Unternehmen A ebenfalls ineffizient werden würde. Durch die Anwendung neuer Technologien können beide Unternehmen wieder effizient werden. (vgl. Coelli u. a. 2005, 3)

Die Bewertung der technischen Effizienz kann auf Grundlage dieses Modells über Distanzfunktionen zwischen Unternehmen und Produktivitätsgrenze sowie verschiedene Methoden erfolgen. An dieser Stelle werden Data Envelopment Analysis und Stochastic Frontier Analysis diskutiert, die beide auf dem Konzept der Produktivitätsgrenze beruhen.

### 3.1.2 Data Envelopment Analysis

Die Data Envelopment Analysis (DEA) beruht auf der von (Farrell 1957) vorgeschlagenen Methode zur Berechnung der technischen Effizienz, die in den folgenden zwei Dekaden jedoch wenig Beachtung fand. Die Methode sieht vor, dass die Produktionsgrenze über die Menge der beobachteten Firmen durch ein Optimierungsproblem abgeleitet wird. Die Arbeit von (Charnes, Cooper, und Rhodes 1978) stellte ein Algorithmus in Form eines Linearen Programms zur Lösung des Problems von Farrell vor und prägte damit den Begriff DEA. Die Methode ist seitdem vom großen Interesse und wird in vielen Bereichen eingesetzt und erweitert.

Die Ausgangdaten des Optimierungsproblems sind die Datenmatrizen $N \times I$ der Inputs und $M \times I$ der Outputs einer Menge Firmen $I$. Für die i-te Firma repräsentiert der Spaltenvektor $x_i$ den Input- und $q_i$ den Output-Vektor. Die Vektoreinträge werden mit Gewichten versehen; zum Beispiel für einen Inputvektor aus Materialen und Arbeitszeit können Materialien mit den Einkaufskosten und die Arbeit mit den Stundenkosten gewichtet werden. Diese Gewichtungen $u$ und $v$ sind für alle Firmen gleich. Das Optimierungsproblem besteht darin, die Gewichtungen für eine Firma so zu optimieren ohne, dass das Verhältnis einer beliebigen Firma von Output zu Input 1 überschreitet. Das Verhältnis für die jeweilige Firma bezeichnet die technische Effizienz. Das Optimierungsproblem kann wie folgt zusammengefasst werden.

$$\begin{aligned} \text{Maximiere } u, v \quad & (u' q_i / v' x_i) \\ \text{Unter Bedingungen} \quad & \left(u' q / v' x_j\right) \leq 1, \qquad j = 1,2, \dots, I. \\ & u, v \geq 0 \end{aligned}$$

Zu den bereits im Text erwähnten Bedingungen wurde lediglich ergänzt, dass $u$ und $v$ nicht negativ sein dürfen. Diese Formulierung hat jedoch das Problem, dass es unendlich viele Lösungen besitzt. Daher wird es in ein Lineares Programm umgewandelt. Dies geschieht durch das Einfügen einer weiteren Bedingung $v' x_i = 1$, so dass der Nenner der Zielfunktion auf 1 normiert wird. Hierdurch erhält man:

$$\begin{aligned} &\text{Maximiere } u, v && (u'q_i) \\ &\text{Unter Bedingungen} && v'x = 1 \\ & && u'q - v'x_j \leq 0, \qquad j = 1,2, \dots, I. \\ & && u, v \geq 0 \end{aligned}$$

Das obige Lineare Problem muss für jede Firma gelöst werden und liefert einen Wert zwischen 1 und 0 für die technische Effizienz und kann somit als Distanzfunktion angesehen werden. Die jeweils effizientesten Firmen definieren die Produktivitätsgrenze anstatt, dass diese vorgegeben werden muss. Dies wird insbesondere durch das visualisierte Beispiel mit zwei Input- und einer Output-Variablen in Abbildung 8 deutlich.

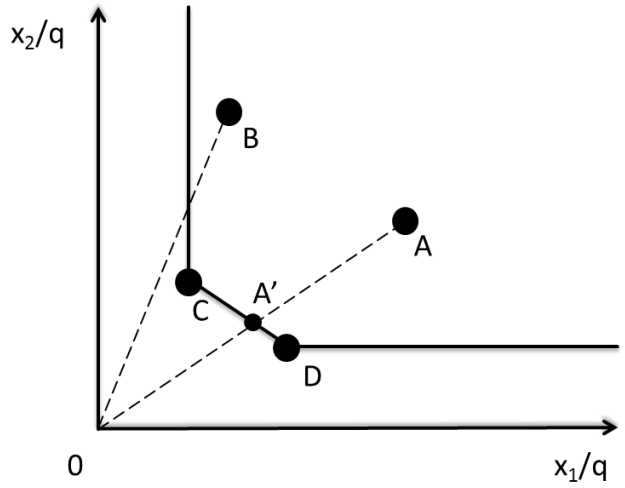


Abbildung 8: Effizienzmessung durch DEA
Quelle: Eigene Abbildung, angelehnt an (Coelli u. a. 2005, 165)

Auf den Achsen ist das Verhältnis der Inputs $x_1$ und $x_2$ zum Output $q$ angegeben. Die Firmen C und D definieren die Produktivitätsgrenze, da sie jeweils bezüglich einer der beiden Inputvariablen am wenigsten benötigen bezüglich eines festen Outputs $q$. Die technische Effizienz dieser Firmen ist 1, die der Firma A ist das Verhältnis des Abstands von 0 zum projizierten Punkt A´ zu dem Abstand von 0 zu dem Punkt A.

DEA lässt sich auch auf das Entwicklungsmanagement anwenden. Die Methode ist jedoch davon abhängig, dass In- und Output quantitativ gemessen werden können und diese für alle Firmen vergleichbar sind. Unter diesen Voraussetzungen ist ein Vergleich zwischen Firmen möglich.

### 3.1.3 Stochastic Frontier Analysis

Die Produktivitätsgrenze wird in diesem Ansatz nicht durch ein Lineares Programm wie in DEA, sondern durch eine Funktion abgebildet. Grundlage dieser Methode ist eine Cobb-Douglas-Produktionsfunktion. Die allgemeine Form dieser Funktion ist:

$$q_i = \prod_j x_{ij}^{a_j} - u_i \text{ mit } a_i, u_i > 0; i = 1, \dots, I$$

Die Variable $q_i$ bezeichnet den Output der i-ten Firma und $x_{ij}$ ist ein Input dieser Firma bezüglich eines Produktionsfaktors $j$. Diese sind in der klassischen Cobb-Douglas-Produktionsfunktion zum Beispiel Arbeit und Kapital. Die Variable $a_j$ ist ein sogenannter Elastizitätsfaktor des Produktionsfaktors und $u_i$ bezeichnet eine Zufallsvariable, die die technische Ineffizienz bezeichnet. Diese lässt sich in Logarithmen-Form darstellen als:

$$\ln q_i = x_i'\beta - u_i \quad i = 1, \dots, I$$

Die Variable $x_i'$ ist ein $j \times 1$ Vektor der Logarithmen der Inputs, ß ist der Vektor der Elastizitätsfaktoren. Dieses Modell berücksichtigt jedoch keine Schwankungen der Produktionsergebnisse durch den Messfehler und statistische Ungenauigkeiten. Unabhängig voneinander wurde durch (Aigner, Lovell, und Schmidt 1977) und (Meeusen und Broeck 1977) das Modell der stochastischen Produktionsgrenze eingeführt.

$$\ln q_i = x_i'\beta + v_i - u_i \quad i = 1, \dots, I$$

Die Variable $v_i$ wurde eingeführt um den statistischen Fehler abzubilden. Dieser Fehler kann negativ oder positiv sein und bezeichnet den nicht-deterministischen Teil des Modells. Die Parameter dieses Modells können anhand ausreichender Daten durch verschiedene Methoden abgeschätzt werden, auf die im Rahmen dieser Arbeit nicht eingegangen wird. Hierzu verweisen wir auf (Coelli u. a. 2005, 242). Die Bedeutung der Variablen $v_i$ und $u_i$ kann im Falle einer Produktion mit einem Input und einem Output grafisch deutlich gemacht werden. Das SFA-Modell kann in diesem Fall vereinfacht werden durch:

$$\ln q_i = \beta_0 + \beta_1 \ln x_1 + v_i - u_i$$

zu

$$q_i = \exp(\beta_0 + \beta_1 \ln x_1 + v_i - u_i)$$

zu

$$q_i = \underbrace{\exp(\beta_0 + \beta_1 \ln x_1)}_{\text{deterministischer Teil}} \times \underbrace{\exp(v_i)}_{\text{stat. Fehler}} \times \underbrace{\exp(-u_i)}_{\text{tech. Ineffizienz}}$$

Durch die Zerlegung in der letzten Zeile kann die deterministische Produktionsgrenze beschrieben werden. Die Funktion ist in Abbildung 9 aufgetragen und wird anhand von zwei Firmen *a* und *b* erläutert. Firma a produziert dem Input $x_a$ und den Output $q_a$ und Firma b entsprechend $x_b$ und $q_b$. Der statistische Fehler der Firma a ist positiv, während der der Firma b negativ ist. Bereinigt um die aufgetretene technische Ineffizienz ist die Grenzproduktivität der Firmen im Diagramm durch $q_a'$ und $q_b'$ beschrieben. Obwohl der statistische Fehler $v_i$ eine Gleichverteilung aufweist wird die beobachtete Effizienz in der Regel unterhalb der Grenzproduktivität liegen, da die technische Ineffizient $u_i$ positiv ist.

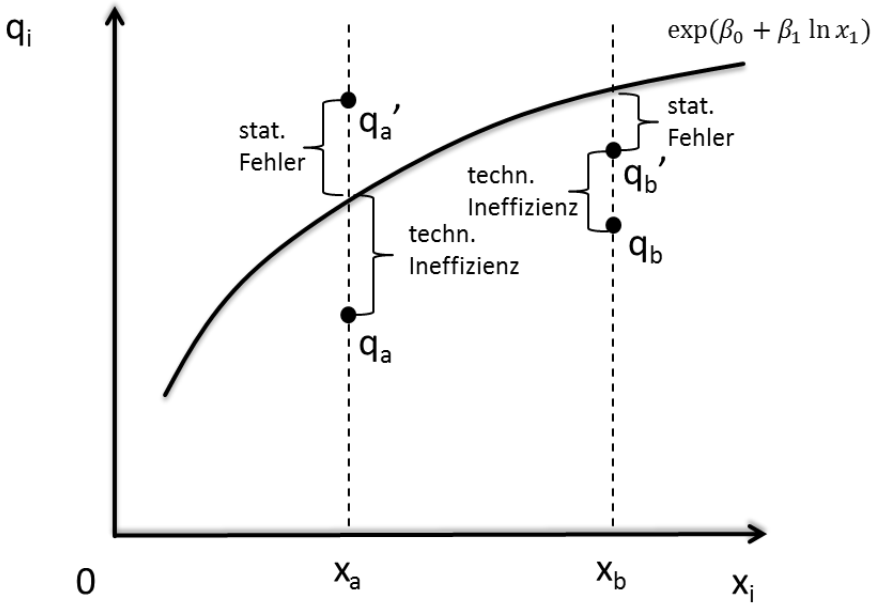


Abbildung 9: Einfaches SFA-Modell
Quelle: Eigene Abbildung, angelehnt an (Coelli u. a. 2005, 244)

## 3.2 Controlling von Forschung und Entwicklung

Die zukünftige Wettbewerbsfähigkeit von Unternehmen wird durch die Leistung der Forschung und Entwicklung im Unternehmen bestimmt (Golder 2000). Dieser Funktionsbereich ist daher kritisch für das Unternehmen; Er muss jedoch trotzdem den zu erwartenden Nutzen von Investitionen im Vergleich zu anderen Investitionsalternativen belegen. Früher wurden die Aktivitäten der Forschungs- und Entwicklung (FuE) als isolierte Funktion betrachtet, die nur schwer zu bewerten und systematisch zu managen ist. Eine bestimmte Menge an Geldern wurden in den Bereich FuE investiert und erwartet, dass ein Nutzen langfristig entsteht (Ojanen und Vuola 2006). Inzwischen ist die Bedeutung des Controllings von FuE jedoch gestiegen.

Laut (Horvath 1996) ist Controlling eine Funktion *„die durch die Koordination von Planung, Kontrolle sowie Informationsversorgung die Führungsfähigkeit von Organisationen verbessern hilft“*. (Ziegenbein 2004, 23) beschreibt Controlling als die *„Bereitstellung von Methoden (Techniken, Instrumente, Modelle, Denkmuster) und Informationen für arbeitsteilig ablaufende Planungs- und Kontrollprozesse sowie die funktionsübergreifende Koordination (Abstimmung dieser Prozesse)“*. Das Controlling ist somit vor allem als eine Managementunterstützungsfunktion zu verstehen, die durch die Bereitstellung von entscheidungsrelevanten Informationen und Methoden die Entscheidungen im Unternehmen verbessert. Das FuE-Controlling ist nach (Bürgel 1989, 1) *„ein funktionsbereichsbezogenes Controlling mit gleichem Inhalt, jedoch spezifisch ange-*

*wandt auf den Bereich FuE"*. Bezogen auf das FuE treten zum Beispiel folgende typische Managemententscheidungen auf:

- **Wie viel soll das Unternehmen in FuE-Maßnahmen investieren?**
  Durch die Budgetierung wird die Leistungsfähigkeit der FuE bestimmt. Interessant ist insbesondere die Frage, ob durch höhere Investitionen in FuE eine höhere Rendite erzielt werden kann.
- **Für welche Projekt oder Themen sollen die Mittel eingesetzt werden?**
  In der Regel gibt es viele Investitionsalternativen im Bereich FuE, die unterschiedlichen Nutzen aufweisen. Experimentelle Produktentwicklungen können zum Beispiel den Weg zu neuen Produkten und Märkten weisen; Prozessverbesserungen durch neue Werkzeuge oder Methoden können die Produktivität im Allgemeinen erhöhen. Durch das FuE-Controlling muss ein geeigneter Mix aus innovativen und risikoreichen und einfachen Vorhaben gefunden werden. Die Zusammenstellung eines Projektportfolios erfordert nicht nur interne Information sondern auch Marktprognosen und Technologiebewertungen.
- **Welche Leistung erbringen die FuE-Bereiche?**
  Die Produktivität von Entwicklungsbereichen kann sehr unterschiedlich sein. Diese Unterschiede sind jedoch nicht so leicht erkennbar wie in der Fertigung, da das „Produkt" abstrakter und schwieriger zu bewerten ist. Das FuE-Controlling steht hier vor allem vor der Aufgabe Methoden im Sinne der Definition von (Ziegenbein 2004, 23) bereitzustellen, welche Entwicklungsergebnisse durch Informationen abbilden, die für Managemententscheidungen geeignet sind. Das FuE-Controlling muss daher ein hohes Fachwissen besitzen, um fachliche Aussagen wie „Abteilung X hat einen Prototyp entwickelt" in ökonomische Informationen wie „Abteilung X hat Forschungsergebnisse mit diesem Renditepotential und Risiko produziert" umzuwandeln. Die Auswahl und Auswertung von FuE-Performanceindikatoren ist hierbei eine herausfordernde Aufgabe (Kerssens-van Drongelen und Bilderbeek 1999).

Die Leistungsbewertung als Teilaspekt des Controllings von FuE stellt eines der Anwendungsszenarien für den Ansatz dieser Arbeit dar. Eine zentrale Rolle spielen in diesem Bereich die FuE-Metriken[1]. In (Hauser und Zettelmeyer 1996) werden einige typische Metriken beschrieben und Handlungsempfehlungen für verschiedene Forschungsstufen gegeben. Diese werden aufgeteilt in Basisforschung (Tier 1), Technologiekompetenzentwicklung (Tier 2) und angewandte Entwicklung für Kunden (Tier 3). Jede Tier-Ebene besitzt andere Anforderungen an Metriken und die Anwendung ungeeigneter Metriken kann kontraproduktiv sein. Eine stärkere Strukturierung bieten (Ojanen und

[1] In diesem Abschnitt wird der Begriff „FuE-Metriken" verwendet, da er in der Literatur geläufiger ist. In der Regel handelt es sich aber um Maße.

Vuola 2006), welche eine Analyse auf Basis von fünf Dimensionen der FuE-Leistungsbewertung definieren.

1. Messperspektive: Für welche Benutzergruppe erfolgt die Messung?
2. Messzweck: Welches Ziel wird mit der Messung verfolgt?
3. Messlevel: Wo und in welchem Umfang wird gemessen?
4. FuE-Typ: Art der FuE. Dies entspricht der Kategorisierung durch Hauser und Zettelmeyer.
5. Prozessphase: Wann erfolgt die Messung?

Die Ausprägungen dieser Dimensionen sind in Tabelle 3 dargestellt. Diese Kategorien haben Überschneidungen zu denen der Komplexitätsmaße im Abschnitt 2.1.3. Hierbei ist jedoch die Kategorie „Prozessphase“ und nicht „Messlevel“ mit der der Kategorie „Gegenstand der Messung“ zu vergleichen.

Tabelle 3: Dimensionen des FuE-Controllings
Quelle: (Ojanen und Vuola 2006)

| **Perspektive** | **Messzweck** | **Messlevel** | **FuE-Typ** | **Prozessphase** |
|---|---|---|---|---|
| Kunde | Begründung für Forschung | Industrie | Grundlagen-forschung | Input (Anforde-rungen) |
| Intern | Benchmarking | Firma | Kompetenz-aufbau | Entwicklungs-begleitend |
| Shareholder | Zuordnung von Ressourcen | Abteilung | Angewandte Forschung | Output (Fertiges Produkt) |
| Andere Stakeholder | Entwicklung von Verbesserungs-programmen | Projekt | Produkt-entwicklung | Rendite (durch Verkäufe etc.) |
| Forschung | Motivation und Belohnung | Individuen | Produkt-verbesserungen | |

Im Folgenden werden einige typische FuE-Metriken beschrieben, die ebenfalls die Entwicklungsleistung als Output-Maßgröße charakterisieren. Einige Beispiele wurden aus (Bürgel 1989, 79) entnommen. Zu jeder Metrik werden die wichtigsten Einschränkungen genannt:

- **DIN-A4-Seiten, Skizzen oder Zeichnungen:** Ein wesentliches Problem dieser Metrik ist, dass die Größe stark von Schriftgröße und Seitenformatierung abhängt, was sich durch eine Standardisierung zum Teil beheben lässt. Es besteht aber weiterhin das generelle Problem, dass der inhaltliche Wert, also die Sem-

antik des Dargestellten, nicht beschrieben wird; Es lässt sich nicht bewerten, ob es sich um „Schmierereien“ oder konzeptionelle Ausarbeitungen handelt.

- **Befehle (Lines of Code):** Siehe hierzu auch 2.1.3.3.1. Problematisch bei der Verwendung als FuE-Metrik sind vor allem Missverständnisse in ihrer Anwendung (Rosenberg 1997) und der umstrittene Nutzen als Output-Metrik. Durch die zunehmende parallele Verwendung mehrerer Programmiersprachen in einem Programm stellen sich außerdem Probleme der Vergleichbarkeit der Leistungen in diesen Sprachen ein. Diese Metrik ist daher als robustes Maß für eine Leistungsmessung nicht geeignet.
- **Patente:** Die Anzahl eingereichter Patente wird insbesondere für Grundlagenforschung angewendet, deren Ergebnis ansonsten schwer zu fassen ist. Da für die Patentierung eine erkennbare Schaffensleistung durch das Patentamt anerkannt werden muss, kann in dieser Phase erstmals eine objektive Wertung der Forschungsergebnisse vorgenommen werden. Allerdings ist diese Metrik nur für die Grundlagenforschung und den Kontext einer Entwicklungsabteilung sinnvoll einsetzbar und ist daher nicht für das angestrebte Ziel der Arbeit brauchbar.
- **Schnittstellen:** Mit der Anzahl benötigter Schnittstellen steigt der Aufwand bei der Gestaltung eines Systems. Diese Metrik geht von einer nicht beeinflussbaren Anzahl an Schnittstellen aus, welches jedoch nicht sehr häufig der Fall ist. In der Regel kann durch einen klugen (und aufwändigeren) Entwurf die Anzahl der Schnittstellen reduziert werden. Die Messung der Anzahl der Schnittstellen als Leistung wäre also in diesem Fall kontraproduktiv. Sie würde zu einem schlechteren Design des Produktes motivieren. Des Weiteren kann der wesentliche Entwicklungsaufwand innerhalb der Komponenten liegen und die Gestaltung der Schnittstellen nur einen geringfügigen Mehraufwand bedeuten.
- **Finanzielle Kennzahlen:** Durch die Erfassung der Entwicklungskosten und der Erlöse durch die entwickelten Produkte können Rentabilitätskennzahlen wie Return-On-Investment ermittelt werden. Dies setzt voraus, dass die FuE-Aktivitäten bestimmten Produkten oder ferner deren Erlösen zugeordnet werden können, was oftmals nicht eindeutig möglich ist. Für die Verwendung als Leistungswert spricht außerdem, dass die Kennzahl nur sehr spät, nach Ende des Produktlebenszyklus, gemessen werden kann und dass die Erlöse von FuE-externen Faktoren wie Qualität des Produktmarketings abhängen können.
- **Subsysteme oder Subfunktionen:** Produkte können in hierarchisch gegliederte Subsysteme unterteilt werden. Gemessen werden kann zum einem die Anzahl an direkten oder indirekten Subsystemen und zum anderen baumbasierte Graphmetriken. Dies setzt voraus, dass die Einteilung in Subsysteme in einer konsistenten Art und Weise erfolgt (Bashir und Thomson 1999b). Andernfalls

hat die Metrik keine Aussagekraft, denn es werden mit unterschiedlich bewerteten Subsystemen ansonsten „Äpfel mit Birnen“ verglichen. Diese Konsistenz ist jedoch in vielen Fällen nicht gegeben, denn eine objektive Bewertung setzt eine tiefgehende Kenntnis aller Produkte voraus, die bei größeren Organisationen jedoch niemand haben kann. Die Strukturierung des Produktes in Subsysteme stellt außerdem bereits eine Entwicklungsleistung dar, die mit diesem Ansatz nicht erfasst wird.

Alle genannten Leistungs-Metriken weisen Probleme auf, die sich vor allem auf die Vergleichbarkeit von Artefakten beziehen. In jeder Entwicklungsumgebung können andere Faktoren aus unterschiedlichen Perspektiven einen wesentlichen Einfluss auf den mit Artefakten verbundenen Aufwand haben. Daher können Metriken, die in bestimmten Unternehmen mit Erfolg eingesetzt wurden, nicht ohne weitere Prüfung auf ein anderes Unternehmen übertragen werden. (DeMarco 1983) beschreibt diesen Fall anhand von Software-Metriken, deren Umsetzung in anderen Umgebungen nicht die vorher in der ursprünglichen Organisation gemessenen Genauigkeiten erzielen konnten.

## 3.3 Prozesskennzahlen

Die Bewertung von Produktentwicklungsprozessen auf Basis der Entwicklungsergebnisse ist in vielen Fällen sehr schwierig durchzuführen, was unter anderem die im vorherigen Abschnitt genannten Beispiele gezeigt haben. Eine andere Herangehensweise für die Leistungsbewertung ist die Untersuchung des Entwicklungsprozesses statt dessen Ergebnisse. In Analogie zur Einordnung von Komplexitätsmessungen bezüglich des Gegenstandes der Messung in Abschnitt 2.1.3.1 können Ansätze zur Leistungsmessung in der Produktentwicklung entsprechend unterschieden werden. In diesem Abschnitt werden der Grundgedanke und die wichtigsten Vertreter der Messung auf Basis des Entwicklungsprozesses vorgestellt.

Entwicklungsprozesse lassen sich durch viele Eigenschaften wie beteiligte Personen, durchschnittliche Dauer von Einzelschritten und Anteil an Überarbeitungen charakterisieren. Viele dieser Kennzahlen können als Qualitätsmerkmale herangezogen und für eine Bewertung der Leistungsfähigkeit aggregiert werden. Die Definition der qualitativen Eigenschaften beruht in der Regel auf gesammelten Erfahrungswerten (Best Practises) und normativen Vorgaben. Die diesem Ansatz zugrundeliegende Prämisse ist, dass qualitativ hochwertige Prozesse eine bessere Leistung im Sinne von Qualität und Kosteneffizienz erzielen.

Insbesondere besitzen in diesem Umfeld Reifegradmodelle eine hohe Bedeutung. Entwicklungsorganisationen werden durch diese Modelle in eine Ordinalskala von Reifegraden eingeordnet. Die Einordnung beruht auf Kriterien wie dem Vorhandensein bestimmter Qualitätssicherungsmaßnahmen, der Dokumentation der Entwicklungsprozes-

se und der quantitativen Bewertung von Entwicklungsergebnissen. Die Prüfung dieser Kriterien erfolgt für eine Organisation im Rahmen eines Audits, dass durch eine berechtigte Person oder Organisation vorgenommen wird. Wichtige Reifegradmodelle sind:

- Capability Maturity Model for Development (CMMI-DEV) für die Softwareentwicklung (Carnegie Mellon University - Software Engineering Institute 2010). Dieser Standard besitzt einen Fokus auf Maßnahmen zur stetigen Prozessverbesserung und teilt Organisation in fünf Reifegrade ein.
- Software Process Improvement and Capability Determination (SPICE) für die Softwareentwicklung und standardisiert als ISO/IEC 15504. Dieser Standard wurde durch die Automotive Special Interest Group (AUTOSIG) speziell für den Automobilbereich als Automotive SPICE erweitert (Automotive SIG - The SPICE User Group 2010).

Einen ähnlichen Grundgedanken wie diese Reifegradmodelle verfolgt die Arbeit von Balazova (Balazova 2004), welche sich auf die Mechatronikentwicklung bezieht: Für eine Organisation werden Organisations- und Prozesskennzahlen (Handlungselemente) ermittelt sowie deren gegenseitige Abhängigkeiten als auch deren Beitrag zu den Unternehmenszielen in Form von Matrizen beschrieben. Dies hat von der Form der Methode Ähnlichkeit zum House-of-Quality, welches in Abschnitt 4.1.1 vorgestellt wird; allerdings sind Entwicklungsprozesse und keine Produkte Gegenstand der Untersuchung. Die Sollkennzahlen werden dann mit Istkennzahlen abgeglichen, die durch Erhebungsbögen in Arbeitssitzungen ermittelt werden. Die Abbildung 10 zeigt die Gegenüberstellung der Profile, die Grundlage einer Maßnahmenermittlung ist.

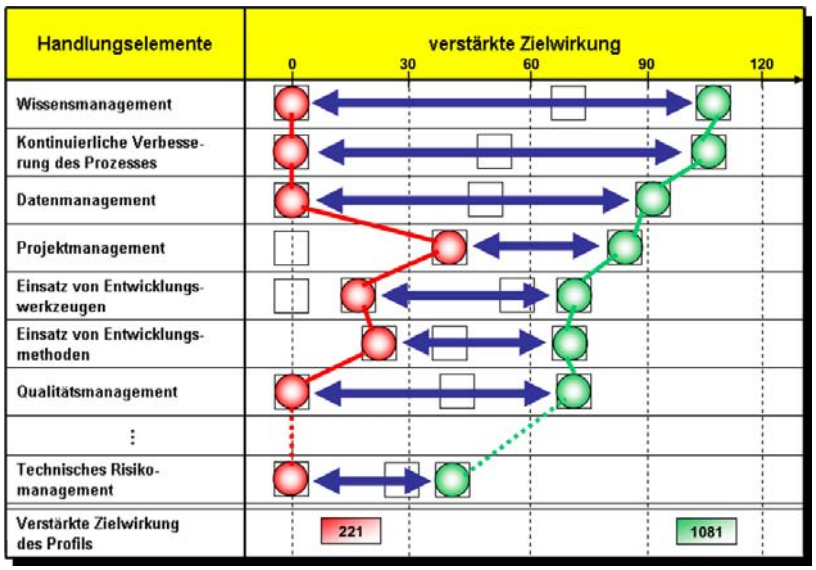


Abbildung 10: Gegenüberstellung von Ist- und Sollprofil
Quelle: (Balazova 2004, 111)

Über die Formalisierung der Zielwirkung und den geschätzten Kosten und Wirkung von Maßnahmen wird durch Balazova ein Rechenmodell beschrieben. Dieses ermittelt die Effizienz von Investitionen in die Verbesserung der Kennzahlen und legt dadurch eine Strategie zur schrittweisen Verbesserung der Prozesse bis zur Erreichung des Sollprofils fest.

Das Ziel dieser Arbeit ist die Messbarmachung von Komplexität als Produktivitätskennzahl auf Basis des Produktmodells (vgl. Abschnitt 2.2.1.1). Daher sind die Ansätze zur Leistungsbewertung durch Prozesskennzahlen als alternative Ansätze nicht geeignet. Sie können vielmehr als Anwendungsrahmen gesehen werden; durch eine Komplexitätsmessung kann eine Quantifizierung des Entwicklungsoutputs erfolgen und somit eine entscheidende Prozesskennzahl beigetragen werden.

## 3.4 Projektmanagement

Die Projektkontrolle als Teil des Projektmanagements wurde bereits in Abschnitt 2.2.1 als Hintergrund der Anforderungen an ein Komplexitätsmaß erwähnt. In diesem Abschnitt werden andere verwandte quantitative Methoden der Projektkontrolle beschrieben. Die wesentlichen Einschränkungen und Defizite werden herausgearbeitet.

### 3.4.1 Earned-Value-Analyse

Die Earned-Value-Analyse (EVA) gehört zu den Methoden der Projektfortschrittskontrolle und basiert auf dem Vergleich der geplanten Projektkosten (Planwert), der tatsächlichen Kosten (Istkosten) und des Sachfortschritts zu einem Zeitpunkt (Leistungswert). Sie ist sowohl zur Kosten- als auch zur Leistungskontrolle geeignet und lässt sich auch zur Prognose einsetzen. Das Verfahren wurde zuerst in den 60er Jahren vom Militär der USA eingesetzt und ist inzwischen ein Standard für die Projektkontrolle bei öffentlichen Projekten in den USA. Die Methode wird in zahlreichen Büchern, insbesondere aus dem angloamerikanischen Raum, beschrieben (zum Beispiel (Fleming und Koppelman 2006), (Humphreys 2002)). Das Project Management Body of Knowledge (PMBOK) (Project Management Institute 2008) und andere Projektleitfäden beinhalten die Methode. Im deutschsprachigen Raum ist die Methode allerdings nicht stark verbreitet. Sie wird zwar im Grundgedanken in der DIN 69903 erwähnt, aber dort nicht näher beschrieben (Stelzer und Bratfisch 2007).

Grundlage der EVA sind die Parameter eines einzelnen Projektes. Da in der Praxis die Englischen Begriffe üblich sind, werden diese verwendet.

- **Budget at Completion (BAC):** Das zu Anfang geplante Gesamtbudget des Projektes in Euro oder Dollar.

- **Schedule at Completion (SAC):** Die zu Anfang geplante Projektdauer. Die derzeitige Dauer sei durch Schedule at Time (SAT)[2] beschrieben.
- **Budgeted Cost of Work Scheduled (BCWS):** Die Kennzahl BCWS beschreibt die geplanten Kosten des Projektes zum Betrachtungszeitpunkt. Dieser Wert kann auf unterschiedliche Weise berechnet werden, je nach Modell des Kostenverlaufs. Eine einfache Variante ist der lineare Kostenverlauf bei dem sich der Wert aus dem Gesamtbudget und der derzeitigen Projektdauer ergibt.

$$\mathrm{BCWS} = \mathrm{BAC} \times \left(\frac{\mathrm{SAT}}{\mathrm{SAC}}\right)$$

  Alternativ hierzu können zum Beispiel logarithmische Modelle verwendet werden, wenn zum Beispiel zu Anfang tendenziell mehr Kosten entstehen als später. Ein genauerer Wert lässt sich mit Hilfe eines Projektplanes mit budgetierten Teilaufgaben erstellen. In diesem Fall wird die zum Zeitpunkt vorgesehene abgelaufene Dauer jeder Teilaufgabe mit deren Budget verrechnet und der BCWS ergibt sich aus der Summe dieser Werte.

$$\mathrm{BCWS} = \sum \mathrm{BAC}_{\mathrm{Aufgabe}} \times \frac{SAT_{\mathrm{Aufgabe}}}{\mathrm{SAC}_{\mathrm{Aufgabe}}}$$

  Die Berechnung über Teilaufgaben stellt also lediglich eine Hierarchisierung des linearen Modells dar. Dieses Modell lässt sich auch iterativ für weitere Unteraufgaben einsetzen oder durch nichtlineare Kostenmodelle für bestimmte Aufgaben verfeinern.
- **Actual Cost of Work Performed (ACWP):** Dieser Wert beschreibt die zum Zeitpunkt bereits entstanden Kosten (Istkosten).
- **Budgeted Cost of Work Performed (BCWP) oder Earned Value:** Die Plankosten der zum Zeitpunkt tatsächlich erbrachten Arbeiten (Leistungswert) beschreiben den Wert der bisher im Projekt erbrachten Leistungen. Sie ergeben sich aus dem Fertigstellungsgrad einer Arbeit und deren Budget.

Durch die EVA können auf Grundlage dieser Parameter Indizes berechnet werden, die den Zustand eines Projektes bezüglich relativer Kostenabweichungen und relativer Terminabweichung charakterisieren. Die Berechnung der Indizes lässt sich auch grafisch durch Auftragen von BCWS, ACWP und BCWP als Funktionen wie beschreiben. Dies ist in Abbildung 11 anhand eines Projektes mit Termin- und Kostenüberschreitung dargestellt. CPI und SPI sind in der Abbildung als absolute Differenzen dargestellt.

- Cost Performance Index (CPI): Dieser Index beschreibt das Verhältnis der zum Zeitpunkt erbrachten Leistung zu dem entstandenen Aufwand. Er beantwortet die Frage wie Effizient das Projekt ist. Die Plankosten entsprechen somit der Grenzproduktivität im Sinne der Produktivitätstheorie (Abschnitt 3.1.1).

[2] Diese Abkürzung ist in Rahmen von EVA nicht üblich und wird hier lediglich zu Vereinfachung der Erläuterung der Kostenmodelle bei BSCW verwendet.

$$\text{CPI} = \frac{\text{BCWP}}{\text{ACWP}}$$

- Schedule Performance Index (SPI): Dieser Index beschreibt die relative Terminabweichung über das Verhältnis von der zum Zeitpunkt erreichten Leistung zu dem geplanten Aufwand. Ein Wert unter 1 beschreibt somit eine Terminverzögerung.

$$\text{SPI} = \frac{\text{BCWP}}{\text{BCWS}}$$

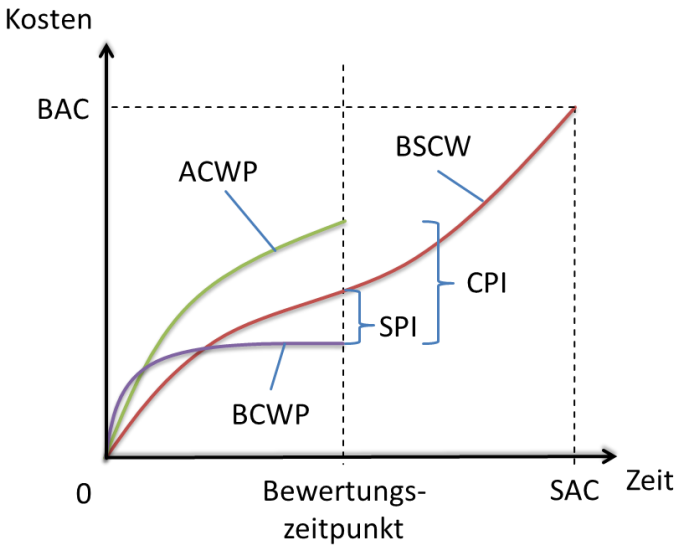


Abbildung 11: Grafische Form der Earned-Value-Analyse

Die Grenzen dieses Verfahrens liegen in der Erfassung des Fertigstellungsgrades für den Earned Value. Es wird keine objektive Grundlage für die Ermittlung dieser Kennzahl anhand des Produktmodells geboten. Interessant ist jedoch die Art der Einbeziehung der Plan- und Sollkosten der Entwicklung in der Bewertung. Im Abschnitt 5.8.3 wird ein an der EVA angelehntes Verfahren entwickelt, deren Abgrenzung zur EVA im Abschnitt 5.9.1.2 detailliert wird.

### 3.4.2 Meilensteintrendanalyse

Insbesondere bei längerfristigen Projekten kann es immer wieder zu Terminveränderungen von Einzelaufgaben kommen. Dies stellt in der Regel kein Problem dar, wenn genügend Puffer im Projektplan einkalkuliert wurde. Daher kann der einmaligen Abweichung vom Projektplan keine große Aussagekraft beigemessen werden (Burghardt 2006, 343). Die wiederholte Korrektur stellt hingegen ein Problem bei der Planung des Projektes dar. Daher ist es notwendig nicht nur Abweichungen vom aktuellen Plan zu betrachten, sondern auch die Änderung des Plans über die Projektlaufzeit zu analysieren. Die Meilensteintrendanalyse stellt ein Werkzeug zur Erkennung von Planungs-

trends anhand der Veränderung bei Meilensteinterminen dar. Sie basiert auf fest definierten Meilensteinterminen, die zum Beispiel die Fertigstellung von Teilprodukten kennzeichnen. Sie lässt sich grafisch sehr übersichtlich wie in Abbildung 12 darstellen.

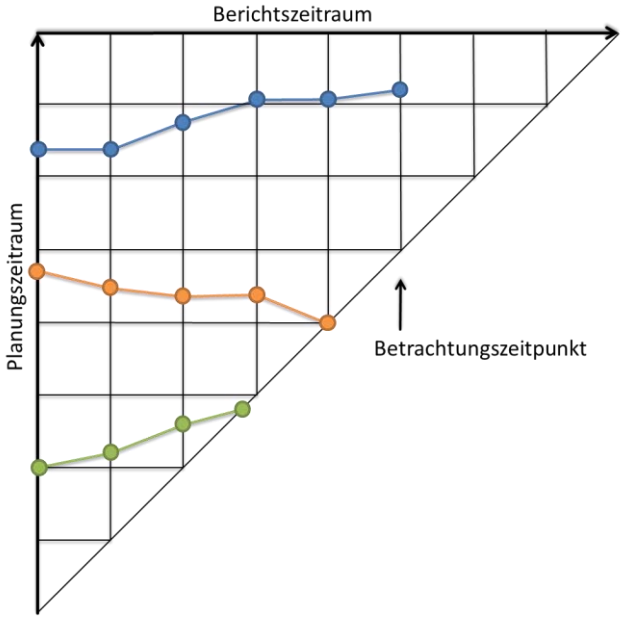


Abbildung 12: Meilensteintrenddiagramm
Quelle: Eigene Abbildung in Anlehnung an (Burghardt 2006, 343)

Auf der waagerechten Achse ist der zeitliche Verlauf der (Neu)-Planungen aufgetragen. Während der Projektlaufzeit werden die geplanten Meilensteintermine von links nach rechts verzeichnet. Die senkrechte Achse bezeichnet hierbei den Termin der geplanten Fertigstellung. Wenn ein Meilensterntermin sich über das gesamte Projekt nicht verändert, so wandert er waagerecht bis zum Ende des Dreiecksrasters, was dessen Fertigstellung bedeutet; Die Planung entspricht dem tatsächlichen Termin. In dem Beispiel wird der obere, blaue Meilenstein immer weiter nach oben verschoben, was eine stetige Korrektur des Meilensteintermins nach hinten bedeutet. Der mittlere, orangene Meilensteintermin wurde hingegen terminlich nach vorne korrigiert und früher als ursprünglich geplant fertiggestellt. An dieser Grafik lässt sich leicht der Trend ablesen, dass der blaue Meilensteintermin voraussichtlich weitere Verzögerungen erleiden wird.

Die Meilensteintrendanalyse bietet einen Überblick über die Terminentwicklung von Meilensteinen und somit ein Mittel um Terminengpässe über den aktuellen Status hinaus zu erkennen. Sie ist aufgrund des geringen Aufwandes leicht durchzuführen und eignet sich als Kommunikationsmittel für Projektbeteiligte. Eine Bewertung der Entwicklung erfolgt aber nur durch Abgleich mit vorherigen Planungswerten. Wie bereits in der Motivation beschrieben, eignet sich diese Vorgehensweise nicht zur Bewertung der Produktivität, da unter anderem die Gestaltung von Puffern sehr unterschiedlich sein kann. Für eine solche Analyse muss eine objektivere Grundlage festgelegt werden.

### 3.4.3 Projektkennzahlenerfassung aus Produktmodellen

Grundlage der vorherigen Methoden sind Kennzahlen wie der Fertigstellungsgrad des Projektes oder der Meilensteine, geplantes und tatsächliches Anfangs- und Enddatum von Vorgängen und entstandene Kosten. In der Regel werden diese Informationen durch die Projektbeteiligten manuell ermittelt oder abgeschätzt und in entsprechende Projektmanagementwerkzeuge eingetragen. Da diese Daten nicht immer aktuell sind und auch subjektiven Einschätzungen unterworfen sind, ist die direkte automatische Erfassung dieser Daten wünschenswert. Als Datenquelle hierfür können Produktmodelle herangezogen werden. Zum Beispiel kann ein Fertigstellungsgrad aus einem Anforderungspartialmodell ermittelt werden, wenn dieses mit einem Partialmodell der Testergebnisse verknüpft ist. Über die Verknüpfung kann ermittelt werden, welche Anforderungen erfolgreich getestet wurden und welche noch nicht erfüllt sind. Durch das Verhältnis von erfüllter und nicht erfüllter Anforderungen kann ein Fertigstellungsgrad definiert werden.

Ein solches Konzept zur Erfassung von Projektdaten auf Basis von Produktmodellen stellt die Arbeit von Jungkunz (Jungkunz 2005) dar. Als Datenquelle werden Produktdatenmanagementsysteme (PDM-Systeme) gewählt, die Metainformationen über Produktmodelle verwalten. Unter der Annahme, dass diese Systeme die Daten in Form des PDM-Austauschformates STEP (Anderl und Trippener 2000) bereitstellen, werden Kennzahlen („Kennlinienelemente“) aus einer in STEP vereinheitlichen Datenbasis ermittelt. Jungkunz beschreibt „Transformationsmuster“, die STEP-Daten auf bestimmte Sätze von Projektkennzahlen für, zum Beispiel, eine Meilensteintrendanalyse abbilden. Hierbei handelt es sich im Wesentlichen um die Realisierung einer Abfragesprache wie SQL für das STEP-Datenmodell EXPRESS. Die Transformationsmuster bestehen aus folgenden Komponenten:

- Präsentierende Elemente: Entspricht einer SELECT-Klausel und definiert die Attribute, die in die Kennzahlen eingehen.
- Skalierende Elemente: Entspricht einer WHERE-Klausel und schränkt die Auswahl der Attribute zum Beispiel auf ein bestimmtes Projekt ein.
- Verdichtungsalgorithmen: Umgangssprachliche Beschreibung von Aggregationen.

Kritisch zu sehen ist, dass Jungkunz weder auf Vorarbeiten im Bereich Datenabfragesprachen zurückgreift, noch die standardisierten Abfrageimplementierungen für STEP in ISO 10303-22 berücksichtigt, was sich deutlich im Formalismus und der Mächtigkeit der Transformationsmuster niederschlägt.

Der im Abschnitt 4.2.1 genauer beschriebene Permeter-Ansatz (Hausmann 2008) implementiert ebenfalls in vergleichbarer Weise eine Abfragesprache für OWL-Modelle. Eine Abfrage wird durch Bausteine definiert, die in Form eines regulären Ausdrucks

kombiniert werden können. Da aus Sicht dieser Arbeit die Modellierung und Aggregation des Produktmodells in Permeter bedeutender ist, wird diese Vorarbeit jedoch unter den verwandten Ansätzen der Strukturanalyse eingeordnet.

Die Ansätze zur Kennzahlenerfassung aus Produktmodellen zeigen auf, dass die Relevanz dieser Daten für die Bewertung von Entwicklungsaktivitäten auch von anderen Autoren erkannt wird. Allerdings wird hierbei der Zusammenhang zur Komplexität oder Produktivität nicht systematisch untersucht. Für die Komplexitätsmessung muss vor der Formulierung von Abfragen an zu bewertende Produktmodelle eine dedizierte Analyse zur Abbildung von Attributen auf Komplexität erfolgen.

## 3.5 Metriken für konzeptuelle Modelle

Konzeptuelle Modelle beschreiben die Domäne eines Produktes in Form von Konzepten und Relationen. Diese Form des Produktmodells wurde zuerst durch die Entity-Relationship-Modelle in der Datenbankentwicklung (Chen 1976) bedeutend. Zuvor beschrieben Produktmodelle vorwiegend nur Eigenschaften des Produktes, zum Beispiel eine konkrete relationale Datenbankstruktur. Diese Produktmodelle sind entsprechend implementierungsorientiert. Wesentliche Bedeutung erhielt diese Modellierung durch die UML-Sprache und die objektorientierte Programmierung im Bereich des Software-Engineerings, welches als konsequente Weiterentwicklung des Ansatzes der ER-Modelle gesehen werden kann (Manso, Genero, und Piattini 2003). Die Abbildung der Konzepte und Relationen einer Domäne in eine UML-Modellierung mit Klassen und Attributen stellt eine wesentlich intuitivere Herangehensweise an die Softwareentwicklung dar. Konzeptuelle Modellierungen sind als wesentlicher Schritt des Entwurfs von Software inzwischen unentbehrlich. Eine wichtige Kategorie konzeptueller Modelle sind Geschäftsprozessmodellierungen, die insbesondere zur Kommunikation mit dem Kunden und der Erfassung von Anforderungen an betriebliche Informationssysteme genutzt werden.

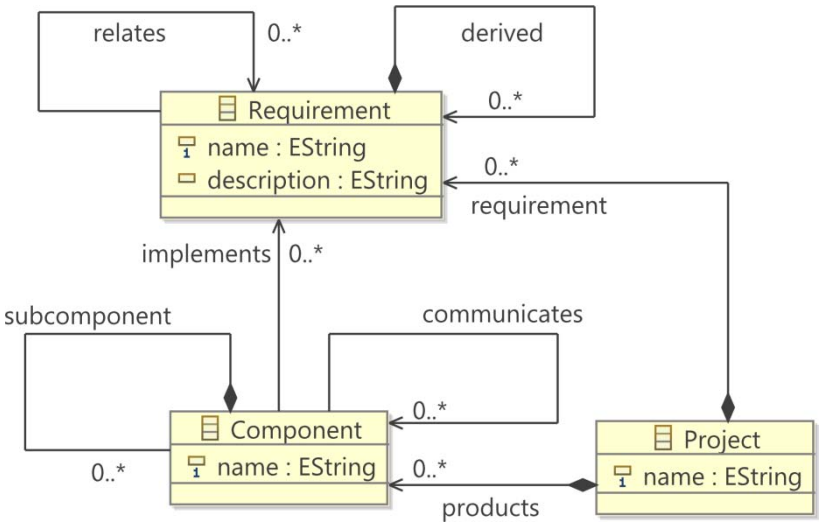


Abbildung 13: Klassendiagramm (EMF) von Komponenten und Anforderungen

Ein Beispiel für ein einfaches konzeptuelles Modell ist Abbildung 13, das wesentliche Konzepte von Elementen einer Produktentwicklung zeigt. Dieses Modell kann für die Abstimmung mit einem Kunden und auch als Vorlage für die Programmierung genutzt werden. (Da es sich um ein EMF-Diagramm (siehe Abschnitt 6.2) handelt, kann aus diesem Modell direkt der Quellcode mit den Methodenrümpfen erzeugt werden.) Wesentliches Merkmal der konzeptuellen Modellierung ist, dass es sich in der Regel um graphenorientierte Abbildungen der Realität handelt. Für die Bewertung von Produktmodellen ergeben sich hierdurch neue Möglichkeiten. Die frühen Software-Komplexitätsmaße wie (McCabe 1976) (weitere siehe Abschnitt 2.1.3.3) orientieren sich an der Syntax der Implementierungssprache wie Zuweisungen oder Variablen. Die Analyse von konzeptuellen Modellen erlaubt hingegen die Quantifizierung auf inhaltlicher Ebene: Wie viele Konzepte werden beschrieben und wie stark sind sie miteinander verknüpft? Auf dieser Ebene können Produkte unabhängig von ihrer konkreten Implementierung miteinander verglichen werden.

Die Analyse und Bewertung von konzeptuellen Modellen ist insbesondere im Bereich des Software Engineering ein beliebter Ansatz. Die Bewertung von konzeptuellen Modellen entwickelte sich von ungenauen Kriterien für „gute“ Modelle zu formalisierten Analyse-Frameworks (Mario Piattini u. a. 2005). Durch Metriken kann die Objektivität der Bewertung in solchen Frameworks sichergestellt werden (Daniel L. Moody 1998). Die geläufigsten Maße werden direkt aus den Eigenschaften des Modells als Graph abgeleitet:

- Anzahl der Konzepte und Klassen
- Verhältnis von Relationen und Klassen
- Anzahl bestimmter Relation, wie zum Beispiel Vererbung oder Aggregation in UML

- Länge von Pfaden im Graphen
- Tiefe von Bäumen bei gerichteten Relationen wie Vererbung

Daneben existiert eine Vielzahl an Vorschlägen für Metriken für konzeptuelle Modelle, die jedoch nur selten empirisch bewertet wurden. Moody stellt eine umfangreiche Liste von Beiträgen dar, von denen nur ungefähr 20% empirisch validiert wurden. Hierdurch sei die Erstellung von konzeptuellen Modellen immer noch mehr eine Kunst als eine Wissenschaft. (D. L Moody 2005)

Strukturelle Eigenschaften wie sie durch Metriken für konzeptuelle Modelle beschrieben werden, sind offensichtlich eine relevante Größe für die Bewertung der Entwicklungsleistung. Eine Integration in den Ansatz dieser Arbeit erscheint daher sinnvoll. Die Herausforderung hierbei ist die Auswahl der relevanten Metriken für die Komplexitätsanalyse. Da der Ansatz disziplinübergreifend gestaltet werden soll, kann keine Festlegung auf bestimmte Strukturen in Modellierungssprachen erfolgen. Die sich aus dieser Forderung ergebenden Unterschiede werden in Abschnitt 5.9.1.4 behandelt.

# 4 Verwandte Ansätze der Strukturanalyse von Produktmodellen

Gegenstand der Komplexitätsbetrachtung sind in dieser Arbeit semantisch gewichtete Beziehungen (siehe Komplexitätsdefinition auf Seite 40). Im Folgenden soll daher zunächst diskutiert werden, wie diese Beziehungen dargestellt werden können. Auf Basis der Darstellungsarten sollen dann existierende Ansätze zur Untersuchung der auf den Beziehungen basierenden Strukturen diskutiert werden. Die Bezüge des in dieser Arbeit entwickelten Ansatzes zu diesen Ansätzen wird in Abschnitt 5.9.2 beschrieben.

Beziehungen oder auch Relationen sind das wesentliche Merkmal von Komplexität in Produktmodellen. Diese Beziehungen können sowohl zwischen Produktkomponenten als auch zu externen Elementen bestehen. Typische Beispiele für solche Relationen sind:

- Entwurfsabhängigkeit
- Unterteilbeziehung
- Implementierung (zwischen Funktion und Produktkomponente)
- Verantwortlichkeit (zwischen Ingenieur und Produktkomponente)

Diese Beziehungen lassen durch lineare Beschreibungen, „Pfeil-Kugel-Diagrammen“ oder auch Matrizen darstellen. Alle Darstellungen lassen sich jeweils als Repräsentation eines Graphen im Sinne der Graphentheorie interpretieren. Graphen sind definiert als ein Tupel $(V, E)$, wobei $V$ eine Menge von Knoten und $E$ eine Menge von Kanten ist. Je nach Eigenschaft der Kanten lassen sich verschiedene Graphtypen beschreiben.

- **Ungerichteter Graph:** $E$ ist eine Untermenge aller 2-elementigen Teilmengen von $V$.
- **Gerichteter Graph:** $E$ ist eine Untermenge des kartesischen Produktes $V \times V$.
- **Graph mit Mehrfachkanten:** Die Untermenge $E$ ist eine Multimenge[3].
- **Hypergraphen:** $E$ ist Untermenge der Potenzmenge[4] von $V$.

Zusätzlich spielen im Rahmen von Produktmodellen Auszeichnungen der Kanten eine wesentliche Rolle:

- **Gewichteter Graph:** Es existiert eine Abbildung $E \rightarrow \mathbb{R}$. Jede Kante ist also mit einem numerischen Wert „beschriftet“.

[3] Eine Multimenge kann dasselbe Element mehrfach enthalten.

[4] Eine Potenzmenge ist die Menge aller Untermengen einer gegebenen Grundmenge (hier die Knoten).

- **Benannter Graph:** Es existiert eine Abbildung $E \rightarrow A$, wobei $A$ eine beliebige Menge bezeichnet. Im Rahmen von Produktmodellen sind $E$ Relationen und $A$ entspricht dem Vokabular der Relationen.

Die Abbildung 14 zeigt die äquivalenten Repräsentationen eines einfachen, gerichteten Graphen als Matrix, lineare Darstellung und Diagramm (von links nach rechts). Die Matrix ist hierbei von vertikaler zu horizontaler Achse zu lesen. Die lineare Darstellung beschreibt die Knoten als Kantenfunktion.

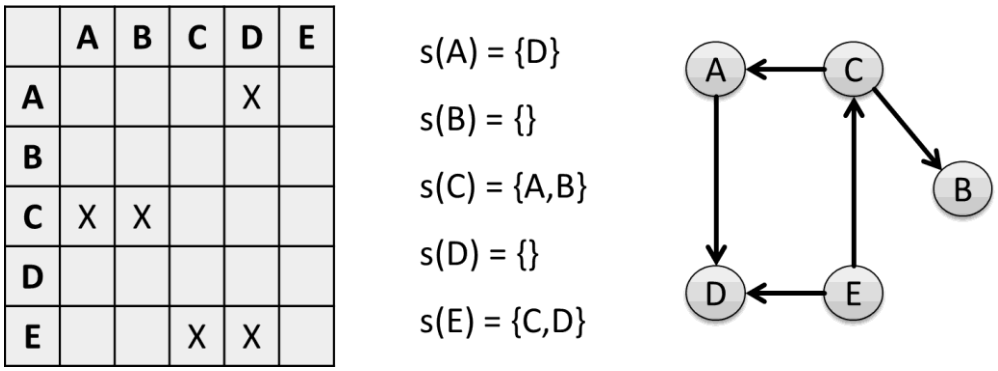


Abbildung 14: Graphrepräsentationen

Die im Folgenden beschriebenen Ansätze sind nach ihrer vorwiegenden Darstellungsform gegliedert. Die Darstellung des Graphen anhand von Knoten und Kanten wie in der Abbildung 14 als Diagramm soll als graphenbasierten Ansatz im engeren Sinne beschrieben werden. Generell sind aber alle Ansätze auch in anderen Repräsentationen durchführbar.

## 4.1 Matrizenbasierte Ansätze

Die Matrizendarstellung kann sehr einfach aus bestehenden Listen erzeugt werden und benötigt keine tiefgreifenden Programmierkenntnisse um sie im Computer darzustellen. Mit Hilfe von Tabellenkalkulationsprogrammen können bereits gerichtete Graphen mit Gewichtung oder Benennung komfortabel erstellt werden. Häufig sind Vorlagen für die Ansätze frei verfügbar. Bei der Verwendung sollten jedoch einige Aspekte beachtet werden.

- Die Anordnung der Knoten entlang der Achsen besitzt in der Regel keine Semantik. Die Zeilen oder Spalten können beliebig getauscht werden. Dies bedeutet, dass äquivalente Graphen sehr unterschiedlich aussehen können. Im Rahmen einer Strukturanalyse wird oftmals versucht eine besser erfassbare Anordnung zu finden. Hierfür existieren Algorithmen für die computerunterstützte als auch manuelle Bearbeitung (Browning 2001).

- Die Interpretation einer Matrix ist manchmal nicht intuitiv möglich. Entscheidende Strukturen wie Schleifen (Feedback Loops) sind nicht sofort ersichtlich und erst als Graph intuitiv erkennbar. (Lindemann, Maurer, und Braun 2008, 90 ff) zeigen hierfür einige Beispiele.
- Matrizen erlauben nicht die Darstellung von Graphen mit Mehrfachkanten. Bei benannten Graphen werden daher als Ersatz häufig mehrere Matrizen benutzt.

Matrizenbasierte Ansätze sind vor allem auf die Bewertung von Relationen bezogen als relevante verwandte Arbeiten aufzufassen. Eine Abgrenzung des im Folgenden entwickelten Ansatzes findet in Abschnitt 5.9.2.1 statt.

### 4.1.1 House-of-Quality

Das „House-of-Quality“ (HOQ) ist ein Werkzeug aus der Quality-Functions-Deployment-Methode (QFD), die 1969 von Yoyi Akao in Japan entwickelt wurde. Die QFD wandelt Benutzerbedürfnisse in Produkteigenschaften und diese in Produktfunktionen um (Akao 1994, 339). Die Methode verwendet eine Vielzahl an Matrizenformen um die für diese Methode charakteristischen Abbildungen zwischen den Perspektiven auf das Produkt darzustellen. *„Grundlegender Ansatz des QFD ist die Verbindung verschiedener Begriffswelten […]“* (Ehrlenspiel 2006, 213). Die Methode umfasst daher eine große Anzahl an Matrizen, Tabellen und Listen, die bedarfsgerecht auf die jeweilige Situation angepasst werden sollen. Die HOQ-Matrix stellt hierbei die erste und bekannteste Matrix dar, welche Kundenanforderungen und technische Merkmale zueinander in Beziehung setzt. Viele Implementierungen beschränken sich auf die Anwendung dieser Matrix.

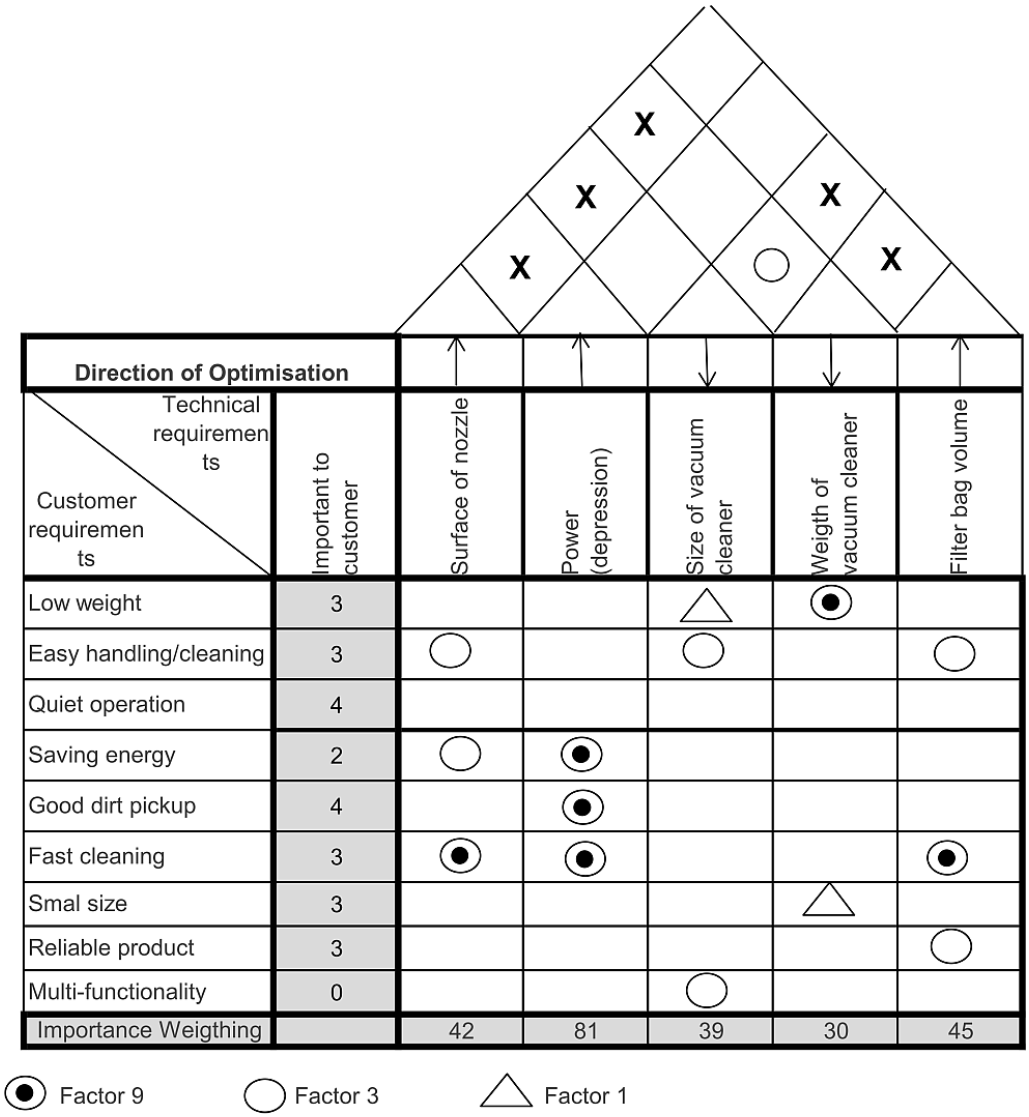


Abbildung 15: House-of-Quality für Staubsauger
Quelle: Eigene Abbildung nach (Eversheim, Breuer, und Grawatsch 2001)

Der Aufbau eines HOQ wird anhand des Beispiels eines Staubsaugers in Abbildung 15 erläutert. Die Anforderungen wie leichtes Gewicht oder gute Schmutzaufnahme werden an der linken Seite auf der vertikalen Achse aufgetragen und erhalten einen Gewichtungsfaktor. Auf der horizontalen Achse werden die technischen Merkmale wie Saugkraft und Gewicht eingetragen. Durch die Matrix in der Mitte werden gewichtete Korrelationen zwischen Anforderungen und technischen Merkmalen beschrieben. Zusätzlich beinhaltet das Diagramm ein „Dach“, welches die technischen Merkmale zueinander in positiver oder negativer Weise korreliert. Das HOQ soll gewährleisten, dass die Eigenschaften des Produktes an den Wünschen und Vorstellungen des Kunden ausgerichtet werden. In ihrer vollständigen Form enthält das HOQ außerdem Korrelationen von Anforderungen zu Anforderungen sowie den Vergleich zu Konkurrenzprodukten.

Das HOQ ist im Wesentlichen eine matrizenorientierte Darstellung zweier gewichteter Graphen. Die innere Matrix beschreibt gerichtete Kanten zwischen zwei disjunkten Knotenmengen (Anforderungen und Eigenschaften). Sie ist im Rahmen der Produktentwicklung als Inter-Domänen-Matrix zu verstehen, die zwei verschiedene Sichten miteinander verknüpft. Das „Dach“ hingegen ist ein nicht-gerichteter[5] Graph über eine Knotenmenge und wird auch als Intra-Domänen-Matrix beschrieben. Sie beschreibt insbesondere Widersprüche in der Konstruktion wie Gewicht des Staubsaugers und Beutelvolumen. Hier sind Parallelen zur Widerspruchsanalyse der TRIZ-Methode (Terninko, Zusman, und Zlotin 1998) zu sehen, die Prinzipien zur Auflösung von Produktparametern in einem matrixorientierten Lösungsbaukasten organisiert. (Eversheim, Breuer, und Grawatsch 2001) beschreiben eine Kombination der Techniken von QFD und TRIZ. Da die Widerspruchsanalyse von TRIZ jedoch nicht der Strukturanalyse dient, wird sie im Rahmen dieser Arbeit nicht weiter behandelt.

Das HOQ ist aufgrund der gewichteten Faktoren für diese Arbeit interessant. Durch die gewichtete Verknüpfung von Anforderungen zu Funktionen erfolgt eine quantitative Bewertung von Elementen eines Produktmodells. Insbesondere kann der Grundgedanke der Modellierung Anregungen für die Gestaltung des Ansatzes bieten. Generell handelt es sich aber um keine Methode zur Produktivitätsanalyse und erfordert subjektive Bewertungen der Produktentwickler.

### 4.1.2 Design Structure Matrix

Die “Design Structure Matrix” oder “Dependency Structure Matrix” (DSM) ist eine matrixorientierte Methode zur Darstellung von Projekten oder Systemen. Der Begriff wurde durch (Steward 1981) bei der matrixbasierten Analyse eines Designsystems geprägt. Beide Achsen der quadratischen Matrix werden in gleicher Reihenfolge mit den Elementen des Systems wie Aufgaben, Komponenten oder Anforderungen beschriftet. Relationen zwischen diesen Elementen werden durch Einträge in dieser Matrix abgebildet, wobei reflexive Relationen nicht erlaubt sind und daher die diagonale Achse ausgegraut ist. Die DSM ist somit in ihrer ursprünglichen Form eine Intra-Domänen-Matrix, die einen gerichteten Graphen repräsentiert. Eine typische DSM für die Komponenten einer Autoklimaanlage und deren Interaktionen ist in Abbildung 16 dargestellt.

[5] Das HOQ gibt eine Optimierungsrichtung für Produkteigenschaften an. Die Beziehung zwischen den Eigenschaften ist jedoch als solche nicht gerichtet, beziehungsweise bidirektional.

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Radiator | 1 | | X | | | | | | | |
| Engine fan | 2 | X | | | | X | | | | |
| Heater core | 3 | | | | | | | | | |
| Heater hoses | 4 | | | | | | | | | |
| Condenser | 5 | | X | | | | X | | X | |
| Compressor | 6 | | | | | X | | | X | X |
| Evaporator case | 7 | | | | | | | | | |
| Evaporator core | 8 | | | | | | X | X | | X |
| Accumulator | 9 | | | | | | | | X | |

Abbildung 16: DSM einer Autoklimaanlage
Quelle: Eigene Abbildung, angepasst und gekürzt nach (Pimmler und Eppinger 1994)

Durch eine DSM können viele verschiedene Sichten auf eine Produktentwicklung wie Prozess, Produktstruktur und Organisation dargestellt werden (Yassine 2004). Die Ziele einer DSM sind ein besseres Verständnis des Systems als auch dessen Optimierung. Einige Anwendungsbeispiele werden in (Lindemann, Maurer, und Braun 2008, 52) dargestellt. In vielen Fällen werden Erweiterungen und Anpassungen der DSM vorgenommen.

- Die Einträge in der Matrix werden mit Prioritäten oder komplexeren Eigenschaften versehen. Dazu werden zusätzliche Auszeichnungen durch Zahlen, Symbole oder Farben vorgenommen. Beispiele hierfür sind Abstufungen von Abhängigkeiten oder Kommunikationsflüssen. In diesem Fall handelt es sich um Graphen mit gewichteten oder benannten Kanten. Durch das Einteilen der Zellen in Feldern können sogar mehrere Relationen dargestellt werden, zum Beispiel in (Pimmler und Eppinger 1994). Diese Form repräsentiert einen Graph mit Mehrfachkanten.
- Die Achsen werden durch verschiedene Domänen belegt und somit zu Inter-Domänen-Modellen. Es werden dadurch vor allem verschiedene Sichten auf das System miteinander verknüpft. Dies weist starke Parallelen zur QFD und eine Abbildung des HOQ als eine Kombination mehrerer DSMs wird sogar durch (Usher, Roy, und Parsaei 1998, 3) dargestellt. Diese Matrizen werden auch als Domain Mapping Matrices (DMMs) als Erweiterung der traditionellen DSM in der Literatur benannt (Danilovic und Börjesson 2001).
- Die Systemelemente werden mit Hilfe von Hierarchien gegliedert. Dies kann zum Beispiel die Zusammensetzung des Systems oder Ober- und Unterfunktionalitäten sein. In der Matrix wird dies als Gliederung auf den Achsen darge-

stellt. Dies entspricht der Ergänzung eines weiteren Intra-Domänen-Graphen mit Baumstruktur.

Die geläufigsten Arten von DSMs können nach Art der Systemelemente in eine Hierarchie eingeordnet werden. In dem Survey von (Browning 2001) wird zwischen statischen und zeitbasierten DSMs unterschieden. Statische DSMs basieren auf Abhängigkeiten zwischen Komponenten oder Personen und zeitbasierte auf sequentielle Anordnung von Aufgaben oder Entwurfsentscheidungen (Parameter decisions). Bei statischen DSMs werden bevorzugt Clusteranalysen verwendet um Projektorganisation oder Systemarchitektur zu strukturieren. Das Ziel ist die Definition von geeigneten Untermengen (Teams oder Baustrukturen), die zueinander möglichst wenige Abhängigkeiten besitzen. Zeitbasierte DSMs werden auf eine geeignete sequentielle Anordnung untersucht um zum Beispiel die Auswirkungen von Wiederholungen in einem Projektplan zu minimieren. Eine weitere Klassifikation kann über die genauere Semantik der Relationen erfolgen. Zum Beispiel können verschiedene Arten von Abhängigkeiten in komponentenbasierten DSMs unterschieden werden.

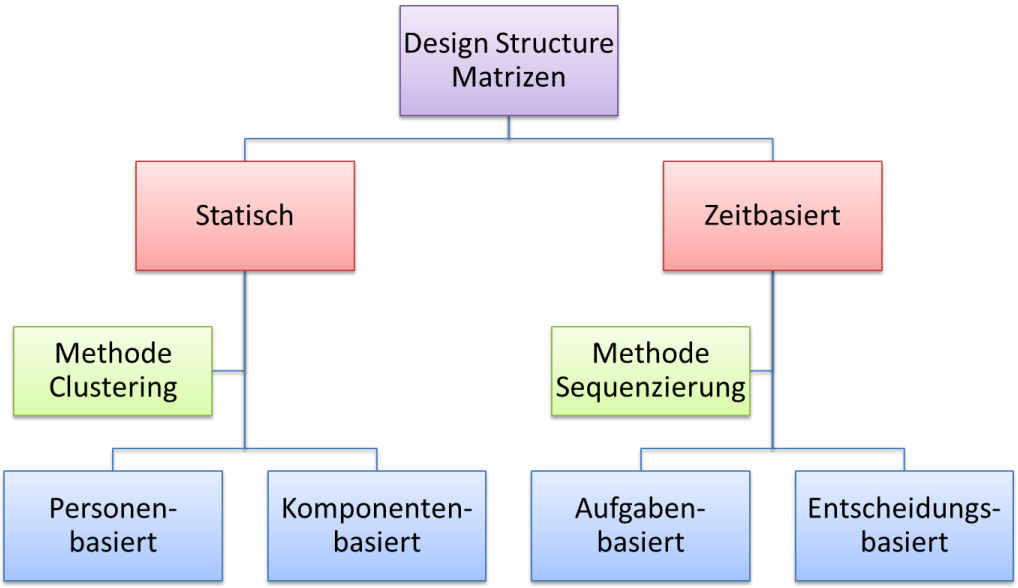


Abbildung 17: Hierarchie der DSMs
Quelle: Eigene Abbildung nach (Browning 2001)

### 4.1.3 Structural Complexity Management

Einen erweiterten Ansatz für die Analyse von Abhängigkeiten innerhalb und zwischen Domänen auf Basis von DSMs und DMMs wird in (Lindemann, Maurer, und Braun 2008) als „Structural complexity management“ dargestellt. Ziel dieses Ansatzes ist die Verbesserung von Systemdesigns sowie das Verständnis der Auswirkungen von Änderungen an bestimmten Komponenten. Ausgehend von einem Entwicklungs- oder An-

passungsproblem eines komplexen Systems werden in fünf Schritten Maßnahmen zur Komplexitätsvermeidung ermittelt.

1. **Systemdefinition:** Das zu analysierende System wird durch eine Menge von Partialmodellen (Domänen) repräsentiert. Abhängigkeiten innerhalb eines Partialmodells werden durch DSM und zwischen Partialmodellen durch DMMs beschrieben. Die Methode sieht unterschiedliche Arten von Abhängigkeiten vor, die gegebenenfalls durch parallele DSMs oder DMMs beschrieben werden. Die erzeugten Matrizen werden durch eine Multi-Domain-Matrix (MDM) geordnet, deren Achsen mit den Partialmodellen belegt sind. Die Zellen der Matrix werden durch die entsprechenden zuvor erstellten Matrizen belegt. Die DSMs sind somit diagonal über die Matrix verteilt und die DMMs befinden sich in den übrigen Zellen. Aufgrund der parallelen Matrizen für unterschiedliche Abhängigkeiten können Matrixzellen durch mehrere DSMs beziehungsweise DMMs belegt sein. Die MDM stellt somit in erster Linie eine Abbildung von Partialmodellpaaren auf eine Menge von Abhängigkeitsgraphen dar.
2. **Informationserhebung:** Dieser Schritt dient der Ermittlung der direkten Abhängigkeiten in den DSMs und DMMs nach der Definition dieser Partialmodelle in der MDM. Hierzu werden Workshops und Interviews durchgeführt. Außerdem wird die automatische Erfassung von Abhängigkeiten miteinbezogen. Ziel ist eine hohe Qualität der Ausgangsdaten für die folgenden Analyseschritte.
3. **Ableitung der indirekten Abhängigkeiten:** Indirekte Abhängigkeiten entstehen durch die Verknüpfung von mehreren Abhängigkeiten. Wenn zum Beispiel in einer DMM über die Partialmodelle Personen und Dokumente definiert ist, dass zwei Personen auf dasselbe Dokument zugreifen, so kann eine weitere Abhängigkeit zwischen beiden Personen abgeleitet werden. Für diesen Schritt werden durch Lindemann unter anderem sechs Ableitungsmuster definiert. Diese können durch Multiplikation der entsprechenden binären Matrizen oder deren transponierten Matrizen berechnet werden.
4. **Strukturanalyse:** Bestimmte Strukturen von Graphen können eine hohe Bedeutung für die Entwicklung haben. Eine solche Struktur sind Zyklen (Feedback-Schleifen), die zum Beispiel im Falle der Konstruktion eines Regelkreises entstehen. Ein anderes Beispiel sind Untermengen von Knoten mit hohem Verknüpfungsgrad (Cluster). Diese Graphmuster werden mit Hilfe der Matrizen erkannt und dienen der Systemanalyse.
5. **Anwendung:** Die Ergebnisse der vorherigen Schritte werden genutzt um das System anzupassen. Hierzu gehört die Vermeidung von bestimmten Graphstrukturen oder die Begrenzung deren negativen Folgen. Eine andere Möglichkeit ist die strukturelle Pareto-Analyse, die quantifizierte Graphstrukturen (zum Bei-

spiel durch die Anzahl beteiligter Zyklen) zur Pareto-Klassifizierung von Systemkomponenten oder Abhängigkeiten nutzt.

Sowohl DSMs als auch Structural Complexity Management als Spezialisierung betonen die Bedeutung bestimmter Strukturen wie Zyklen bei der Bewertung von Produktmodellen. Es liegt nahe, solche Strukturen bei der Gestaltung des Ansatzes dieser Arbeit ebenfalls zu berücksichtigen. Über den Rahmen dieser Ansätze hinaus, muss jedoch dem Benutzer eine Unterstützung bei der Interpretation der Auswirkungen auf den Komplexitätswert gegeben werden. Dieser Zusammenhang kann jedoch je nach Disziplin schwanken und kann in dieser Arbeit nicht auf allgemeiner Ebene beschrieben werden.

## 4.2 Graphbasierte Ansätze

Unter graphbasierten Ansätzen im engeren Sinne sind solche Ansätze zu verstehen, die Knoten und Kanten entweder in einem Diagramm für den Benutzer oder als internes Datenmodell zur computerbasierten Repräsentation verwenden. Diese Ansätze haben gegenüber matrizenbasierten Ansätzen den Vorteil, dass sich mehrere Typen von Beziehungen sehr einfach durch einen einzigen benannten Graphen darstellen lassen, während für matrizenbasierte Ansätze in der Regel pro Beziehungstyp eine Matrix angelegt werden muss. Das House-of-Quality (siehe Abschnitt 4.1.1) kann zum Beispiel als möglichst elegante Darstellung dieser parallelen Matrizen aufgefasst werden, die bei einer graphbasierten Darstellung durch einen Graphen darstellbar wären. Im Falle von dünnbesetzten Matrizen ist die Darstellung als Graph des Weiteren wesentlich kompakter.

Eine wesentliche Herausforderung bei graphbasierten Ansätzen mit vielen Daten ist die Repräsentation für den Benutzer. Wenn zu viele Kanten und Knoten vorliegen, so kann die Übersicht sehr schnell verloren gehen, insbesondere, wenn sich Kanten grafisch überschneiden. Das Themengebiet Graphzeichnen in der Informatik beschäftigt sich mit der benutzerfreundlichen Darstellung solcher Graphen. Ein weiterer Ansatz ist, jeweils nur einen Ausschnitt der gesamten Datenmenge darzustellen, indem in der grafischen Repräsentation bestimmte Knoten oder bestimmte Typen von Beziehungen gefiltert werden.

### 4.2.1 Permeter

Der Ansatz der Software Permeter (Hausmann 2008) basiert auf dem Einsatz semantischer Technologien, die zur graphbasierten Darstellung von Produktmodellen genutzt werden. Ziel des Ansatzes ist ein entwicklungsbegleitendes Controlling auf Basis eines vollständig verknüpften Modells des Entwicklungsprozesses. Dieses Modell wird durch die Transformation und Vereinigung von Daten des Entwicklungsprozesses wie Produktmodelle und Projektpläne erzeugt. Das heißt, es werden zunächst verschiedene

Rohdaten des Entwicklungsprozesses in Teilgraphen, die sogenannten Partialmodelle, überführt. In einem darauffolgenden Schritt werden diese Partialmodelle mit Hilfe von manuellen oder automatischen Werkzeugen miteinander zu einem Gesamtgraphen verknüpft. Hervorzuheben ist hierbei das Werkzeug MatrixBrowser, das dem Benutzer die Verknüpfungen zwischen zwei Modellen anhand einer matrixbasierten Darstellung ermöglicht.

Das Konzept des Permeter-Ansatzes wird in der Abbildung 18 zusammengefasst. Das Vorgehen ist von unten nach oben zu lesen. Die Transformationen der Produktdaten aus den verschiedenen Quellen sind durch die nach oben zeigenden Pfeile dargestellt. Sie erzeugen graphenbasierte Instanzenmodelle (a-box). Deren Metamodelle mit der semantischen Modellierung der Konzepte und Relationen der Partialmodelle sind statisch in den Ontologien (t-Box) hinterlegt. Die Verknüpfung der Instanzenmodelle wird durch die horizontalen Pfeile angedeutet. Die Gesamtheit dieser Modelle (blauer Bereich) stellt das integrierte Modell des Entwicklungsprozesses dar.

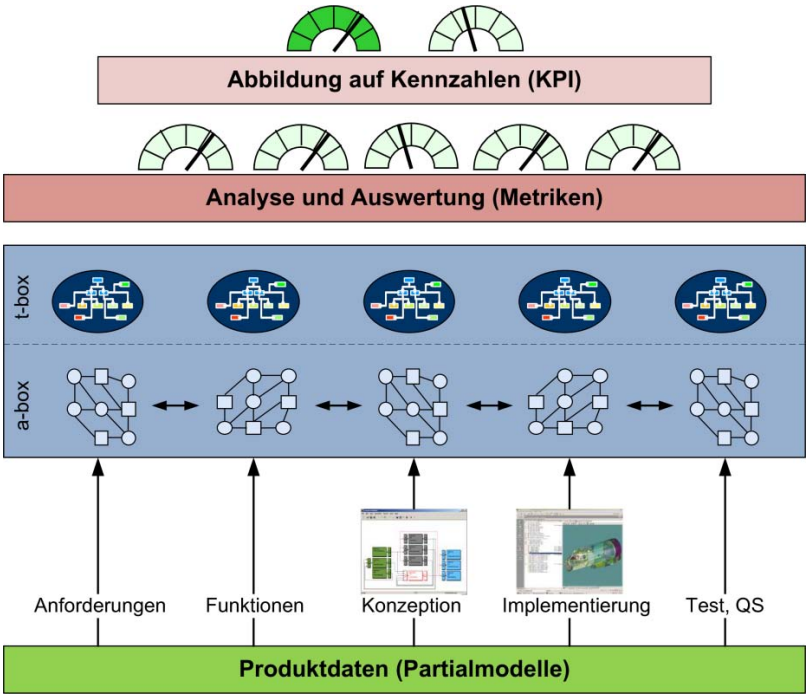


Abbildung 18: Konzeptuelle Übersicht des Permeter-Ansatzes
Quelle: (Hausmann 2008, 98)

Die Analyse und Auswertung des integrierten Modells für das Projektcontrolling wird durch Metriken durchgeführt. Bestandteil des Ansatzes ist die Zusammenstellung dieser Metriken nach einem Bausteinprinzip (siehe auch Abschnitt 3.4.3): Aus atomaren Abfragen an das integrierte Modell und die Kombination mit Operatoren können komplexere Anfragen zusammengestellt werden. Die Metriken können nach dem gleichen Prinzip zu Kennzahlen zusammengefasst werden, die in ein Reporting-System einflie-

ßen. Diese Methode wird an dem Goal-Question-Metrik-Prinzip (Basili u. a. 1994) angelehnt, das die Ableitung von Metriken von Zielen und deren zugehörigen Kennzahlen vorschlägt.

Der Permeter-Ansatz spielt als Vorgängerarbeit eine bedeutende Rolle bei der Gestaltung des Ansatzes dieser Arbeit. Insbesondere ist die Betrachtung der Produktinformationen als Menge von Partialmodellen herauszuheben. In Permeter wird die Bedeutung der partialmodellübergreifenden Relationen betont. Dieser Gedanke sollte in die Analyse der Faktoren für Komplexität einfließen. Im Gegensatz zum Permeter-Ansatz soll die Bildung der „Kennzahl Komplexität“ jedoch nicht auf Basis des Goal-Question-Metrik-Prinzips erfolgen. Die Vorgehensstufe „Question“ beschreibt die Ableitung von zu messenden Eigenschaften auf Basis des festgelegten Ziels der Messung. Dies ist jedoch gerade aufgrund des schwer zu fassenden Charakters der Komplexität, wie in Abschnitt 2.2.1 beschrieben, ohne Hilfsmittel nicht zuverlässig durchführbar. Die Abgrenzung des im Folgenden entwickelten Ansatzes zu Permeter wird in Abschnitt 5.9.2.2 durchgeführt.

# 5 Konzeption eines Ansatzes zur Komplexitätsmessung von Produktmodellen

In diesem Kapitel wird ein Ansatz zur Komplexitätsmessung von Produktmodellen konzeptionell erarbeitet. Die einzelnen Schritte werden mit einem durchgehenden Beispiel eines Geschirrspülers für den Thekenbereich erläutert. Es wird außerdem ein Vorgehensmodell für die Erzeugung eines Komplexitätsmaßes beschrieben. Die Konzeption basiert auf den in Abschnitt 2.2 hergeleiteten Anforderungen für ein Komplexitätsmaß und der daraus abgeleiteten Definition für Komplexität.

Das Konzept ist zunächst implementierungsunabhängig und wird auf graphformaler Ebene entwickelt. Die Implementierung des Ansatzes kann in verschiedenen Modellierungstechnologien erfolgen. Die prototypische Implementierung auf Basis von Eclipse EMF für die Validierung wird in Kapitel 6 beschrieben.

## 5.1 Durchgehendes Beispiel: Gewerbespüler für den Einsatz im Thekenbereich

Das durchgehende Beispiel dient der Erläuterung der einzelnen Schritte des Konzeptes und den Zielen des Ansatzes. Es werden außerdem einige praktische Aspekte des Ansatzes erläutert.

### 5.1.1 Umfeld und Aufgabenstellung des Beispiels

In der Gastronomie sind Spülmaschinen nicht mehr wegzudenken. Der Kunde erwartet im Rahmen der Gastlichkeit nicht nur ein gutes Essen, sondern auch Sauberkeit und Hygiene beim Geschirr. Insbesondere bei Großküchen, zum Beispiel der Mensa an einer Universität, fallen große Mengen Geschirr in kurzen Zeiträumen an. Für diesen Einsatzbereich sind spezielle Maschinen entwickelt worden, die sich von Haushaltgeräten durch Leistung und Handhabung unterscheiden: Ein Reinigungsdurchgang erfolgt innerhalb von wenigen Minuten anstatt über mehr als eine Stunde. Die Geschirrkörbe sind nicht fest in die Maschinen integriert um das parallele Reinigen, Beladen und Entladen von Körben zu ermöglichen und die Beschickung der Maschine erfolgt meistens auf Arbeitstischhöhe um häufiges Bücken zu vermeiden. Eine Nachbehandlung des Spülgutes sollte möglichst gering ausfallen. Einige Beispiele für solche Maschinen aus einem Produktprogramm für Gewerbespüler sind in Abbildung 19 dargestellt.

Abbildung 19: Auszug aus dem Programm für Miele Gewerbegeschirrspüler
Quelle: Präsentation der Firma Miele AG Professional

Diese Maschinen sind für den Einsatz in abgetrennten Küchenbereichen vorgesehen. In manchen mittelständischen Gastronomiebetrieben (Bar, Café, Restaurant) sind jedoch solche Bereiche nicht vorhanden oder für den Kunden einsehbar. Häufig werden anstatt von Spülmaschinen Spülbecken zur manuellen Reinigung benutzt, die oftmals nicht das hohe Maß an Hygiene wie Spülmaschinen erzielen können. Für diesen Bereich sollen Geräte entwickelt werden, die anstatt einer „Industrieoptik“ ein ansprechendes Design für den Einsatz im Thekenbereich besitzen. Die Herausforderung liegt bei der Vereinigung von anspruchsvoller Gestaltung und professioneller Produktivität der Maschine. Die Aufgabe beschränkt sich zunächst nur auf die Entwicklung eines Konzeptes und nicht die vollständige Ausarbeitung.

### 5.1.2 Entwicklung und Produktmodelle des Gewerbespülers

Die entwickelte Produktidee beschreibt einen Gewerbespüler mit Öffnungsmechanismus in vertikaler Richtung, der durch seitliches Einschieben befüllt wird. Hierdurch ist die Maschine besonders platzsparend. Die Entwurfsidee stellt eine Abwandlung des Prinzips des Haubenspülers (linke Seite in Abbildung 19) dar und wird in Abbildung 20 skizziert. Statt des Aufsetzens einer Haube wird der Korb in der Maschine und somit in der Theke versenkt.

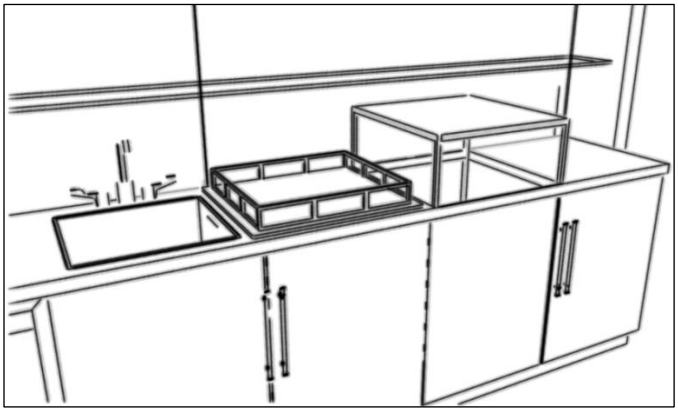

Abbildung 20: Einbau des Gewerbespülers im Thekenbereich

Die Entwicklung erfolgt über mehrere Produktmodelle, die über Relationen miteinander in Beziehung stehen. Modelle sind hierbei alle Entwurfsdokumente, die Aussagen über das zukünftige Produkt treffen.

Am Anfang steht eine Anforderungsliste auf Basis einer Befragung von Kleingastronomien. Es werden Informationen über die Menge des Spülgutes sowie über das Benutzerprofil gesammelt. Zum Beispiel muss eine kurze Einarbeitungszeit reichen, da in kleineren Betrieben das Personal (Studenten, Teilzeitarbeitende) häufig wechselt. Das nächste Modell sind Entwürfe aus einer „Brainwriting“-Phase um verschiedene Konstruktionsideen zu entwickeln. Im Gegensatz zu den anderen Modellen werden diese aus Kreativitätsgründen auf Papier beschrieben. Parallel dazu gestalten Designer einen Entwurf des Maschinenäußeren mit einer speziellen Designsoftware. Diese beiden Entwürfe fließen bei der Ausarbeitung der Idee in das letzte Produktmodell ein. Hierbei handelt es sich um ein 3D-CAD-Modell auf Basis von CATIA V5. Der Entwurfsablauf mit den Produktmodellen ist an Abbildung 21 dargestellt. Die Abbildung ist zwischen den Perspektiven des Technikers und des Designers aufgeteilt.

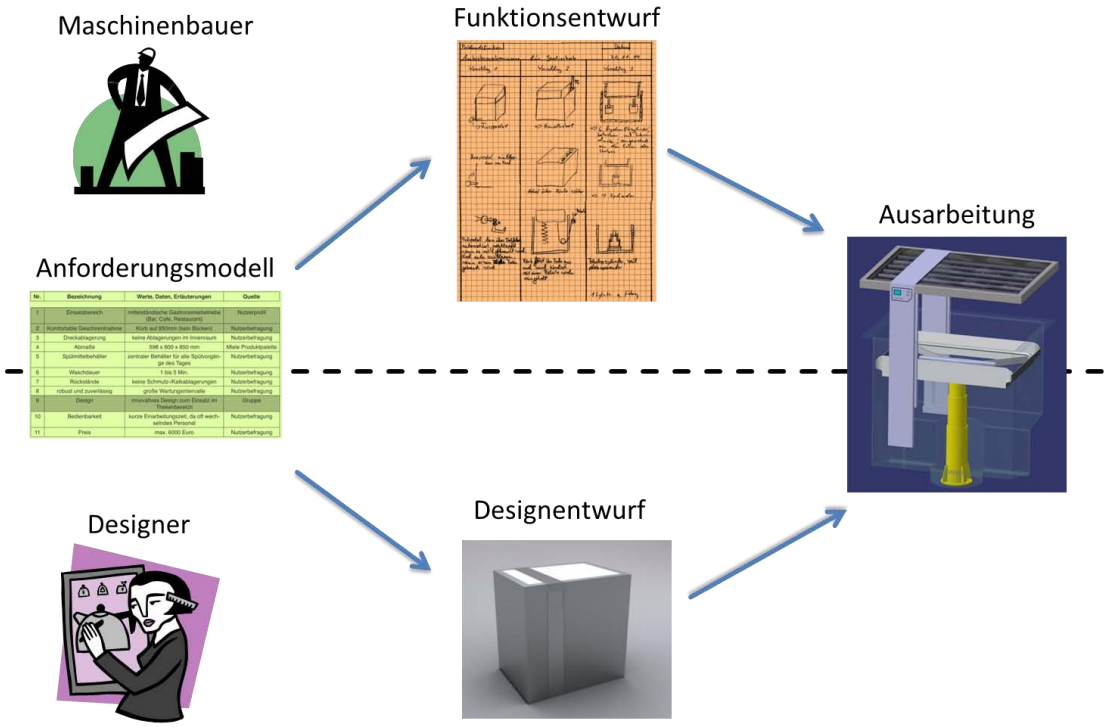


Abbildung 21: Produktmodelle des Gewerbespülers

### 5.1.3 Fragestellung der Komplexitätsmessung

Die Zusammenarbeit von Maschinenbauern und Designern hat ein herausragendes Produktkonzept ergeben. Es wurde jedoch auch festgestellt, dass der Aufwand im Verhältnis zu vergleichbaren Konstruktionsaufgaben wesentlich höher war. So musste die erste Version des CAD-Modells auf Drängen der Designer stark überarbeitet werden um den Anforderungen für ein Design im Thekenbereich zu entsprechen. Aufgrund der guten Ergebnisse wären weitere multidisziplinäre Zusammenarbeiten jedoch trotzdem vielversprechend.

Aus Sicht des Entwicklungscontrollings ist es wichtig zu bewerten, welche Faktoren bei der Zusammenarbeit für den Mehraufwand verantwortlich sind. Dadurch könnte der Mehraufwand bei der Entwicklung des produktionsreifen Entwurfs und bei ähnlichen Entwicklungsprojekten wesentlich besser belegt werden. Die Erkenntnisse wären dann auf vergleichbare Gewerbespüler aus dem Großküchenprogramm anwendbar und der Entwicklungsaufwand für die Thekenvarianten abschätzbar. Des Weiteren will man die Produktivität der Entwickler aus beiden Bereichen besser miteinander vergleichen können um festzustellen, wie effizient die Zusammenarbeit bei ähnlichen Projekten ist. Es sind somit zwei Fragestellungen bezüglich der Komplexität aus Sicht des Entwicklungscontrollings vorhanden:

- Welche Faktoren beeinflussen den Aufwand bei diesen multidisziplinären Entwicklungen und in welchen Umfang tun sie dies?

- Wie kann die Produktivität zwischen normalen und designorientierten Konstruktionsprozessen bei Gewerbespülern verglichen werden?

Die Konzeption des Ansatzes wird zeigen, wie diese exemplarischen Fragestellungen durch den Ansatz zur Komplexitätsmessung beantwortet werden können. Das zu entwickelnde Komplexitätsmaß ist nur für diesen speziellen Anwendungsbereich mit den zuvor beschriebenen Produktmodellen anwendbar. Es wird jedoch keine Einschränkung des Ansatzes bezüglich der Konstruktionsdomänen und Typen von Produktmodellen vorgenommen.

## 5.2 Konzeptuelle Übersicht

Dieser Abschnitt bietet eine Übersicht über das Vorgehensmodell zur Komplexitätsmessung von Produktmodellen. Das Konzept beruht auf den in Tabelle 2 zusammengefassten Anforderungen als auch auf den herausgearbeiteten Definitionen aus Abschnitt 2.2.2, die hier zur Übersicht nochmal wiederholt werden.

- **Komplexität** ist die durch den Aufwand gewichtete Semantik einer Beziehung.
- Die **Komplexität eines Artefaktes** ist die Summe der Komplexitäten der Semantiken seiner Beziehungen.
- **Artefakte** sind Modellelemente, denen eine Komplexität zugeordnet werden kann.

Grundlage des Vorgehensmodells ist insbesondere die Anforderung PA2 „Individuelles Komplexitätsmaß“, das statt eines allgemeinen Komplexitätsmaßes eine Methode zur Erstellung eines individuellen Komplexitätsmaßes fordert. Dies begründet sich zum einem aus der geforderten Domänenunabhängigkeit des Ansatzes (Anforderung KA3): In jeder Domäne besitzen andere Faktoren einen wesentlichen Einfluss auf die Komplexität. Zum anderen folgt dies aus der Definition der Komplexität anhand des Aufwandes: In Abhängigkeit von der Entwicklungsumgebung unterscheidet sich der Aufwand für die Erzeugung desselben Produktmodells. (Vergleiche hierzu Abschnitt 2.1.2.2 zur Quantifizierung von Komplexität als Schwierigkeit der Erzeugung.) Aus diesen Gründen beschreibt das Vorgehensmodell eine Methode zur Erzeugung und Anwendung eines Maßes.

Als Basis für das Komplexitätsmaß wird ein Referenzprodukt gewählt, dessen Komplexität im Sinne der Aufwände für die Komponenten des Produktes bekannt ist. In dem laufenden Beispiel ist dies das Konzept der Spülmaschine für den Thekenbereich. Die Komplexität dieses Produktes ist bekannt, da die Aufwände aus der Entwicklung erfasst wurden. Die Bewertung der Komplexität des Referenzproduktes wird in Abschnitt 5.4 detailliert. Die Gesamtheit aller Modelle, die das Referenzprodukt und dessen Entwicklung beschreiben, wird als Referenzproduktmodell bezeichnet. Das Komplexitätsmaß

wird zunächst als relatives Maß zum Referenzproduktmodell entwickelt. Das heißt der Komplexitätswert beschreibt die Komplexität des gemessenen Produktes im Verhältnis zum Referenzproduktmodell. Die Abbildung 22 verdeutlicht dieses Prinzip anhand des durchgehenden Beispiels; das Maß soll zur Bewertung vergleichbarer Produktmodelle, hier die Ausarbeitung und ein parallel erstelltes Konzept, verwendet werden können.

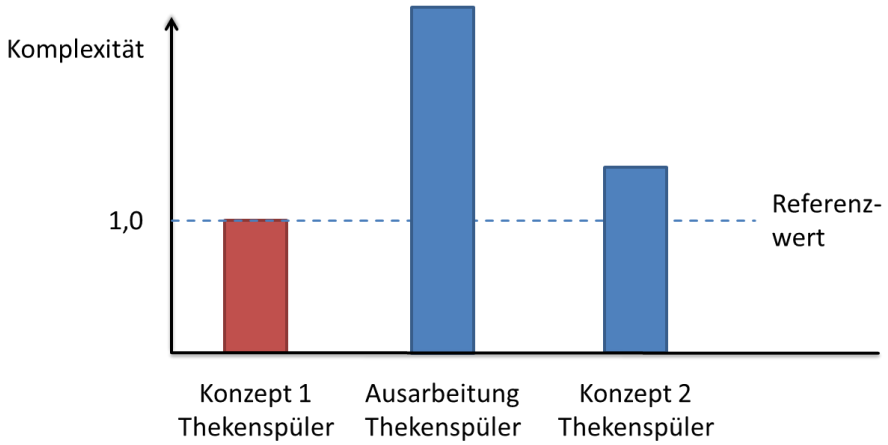


Abbildung 22: Komplexitätsmaß als Verhältniswert zum Referenzprodukt

Um das Komplexitätsverhältnis zweier Produktmodelle zu bestimmen wird in dem Ansatz dieser Arbeit auf der entwickelten Komplexitätsdefinition der Komplexität als die durch den Aufwand gewichtete Semantik einer Beziehung aufgesetzt. Im Falle des Referenzmodells sind die Komplexitäten der Artefakte sowie deren Beziehungen in Form des Produktmodells zueinander bekannt. Zwischen diesen Elementen besteht eine Kausalbeziehung, der mit dem vorgestellten Ansatz quantitativ untersucht wird. Hierzu werden vom Graphen abgeleitete metrische oder nominale Attribute der Artefakte als unabhängige Variablen $X_1, X_2, \dots, X_j$ im Sinne einer statistischen Analyse behandelt. Die Komplexitätswerte der Artefakte werden hingegen als abhängige metrische Variable $Y$ betrachtet. Durch die Definition der Komplexität der Artefakte wird ein Wirkzusammenhang wie folgt impliziert.

$$Y = f(X_1, X_2, \dots, X_j)$$

Eine Verfeinerung dieser Hypothese und eine detaillierte Beschreibung des auf dieser Hypothese basierenden Regressionsmodells erfolgt in Abschnitt 5.5.4. Es liegen somit die notwendigen Voraussetzungen für eine Regressionsanalyse vor, dessen Elemente durch Tabelle 4 (in Anlehnung an (Backhaus u. a. 2008, 53)) zusammengefasst werden. Die Bestimmung der Werte für das Regressionsmodell oder für die Anwendung des Komplexitätsmaßes wird in den folgenden Abschnitten diskutiert.

Tabelle 4: Elemente des Ansatzes als Regressionsanalyse

| **Elemente des Ansatzes** | **Attribute der Artefakte** | **Komplexitätswerte der Artefakte** |
|---|---|---|
| **Bestimmung der Werte** | Ableitung vom Produktgraphen (Abschnitt 5.3) | Erfasste Aufwände (Abschnitt 5.4) |
| **Bedeutung in der Regression** | Unabhängige Variablen, Attribute, Regressor | Abhängige Variable, Komplexität, Regressand |
| **Skalenniveau** | Metrisch oder nominal | Metrisch |
| **Anzahl** | Eins bis sehr viele | Eins |
| **Bezeichner** | $X_1, X_2, \ldots, X_j$ | $Y$ |

Durch die Analyse des Zusammenhanges zwischen den Attributen der Artefakte und deren Komplexitätswert wird die Prognose von Komplexitätswerten auf Basis von Produktmodellen möglich. Dies stellt den Kern des Ansatzes dar und wird in Abschnitt 5.6 beschrieben. Die Regressionsanalyse entspricht somit der Kalibrierung des Komplexitätsmaßes; die Anwendung des quantifizierten Zusammenhanges auf ein anderes Produktmodell zur Prognose des Regressanden entspricht dem Vermessen des Modells. Diese beiden Aspekte des Ansatzes, Kalibrierung und Messung, werden in diesem Kapitel anhand von Vorgehensmodellen beschrieben. Die einzelnen Phasen innerhalb der Modelle werden in den Unterabschnitten dieses Kapitels detailliert und anhand des durchgängigen Beispiels erläutert. Zunächst wird jedoch im Folgenden eine Übersicht zu den beiden Modellen gegeben.

### 5.2.1 Übersicht zur Erstellung des Komplexitätsmaßes

Die Abbildung 23 beschreibt die Schritte zur Erstellung eines Komplexitätsmaßes und die erzeugten und verwendeten Daten der Schritte. Die blauen Elemente stellen Informationen oder Modelle dar, die entweder bereitgestellt oder durch die Schritte der Erstellung des Komplexitätsmaßes erzeugt werden. Die Schritte sind durch rote Elemente dargestellt und ihrer Reihenfolge nach nummeriert. Sie benötigen bestimmte Informationen (eingehende Pfeile) und erzeugen ein Ergebnis für die nachfolgenden Schritte (ausgehende Pfeile). Die vier Hauptphasen sind durch die Hintergrundrahmen voneinander abgegrenzt, greifen jedoch auf Ergebnisse anderer Phasen zurück.

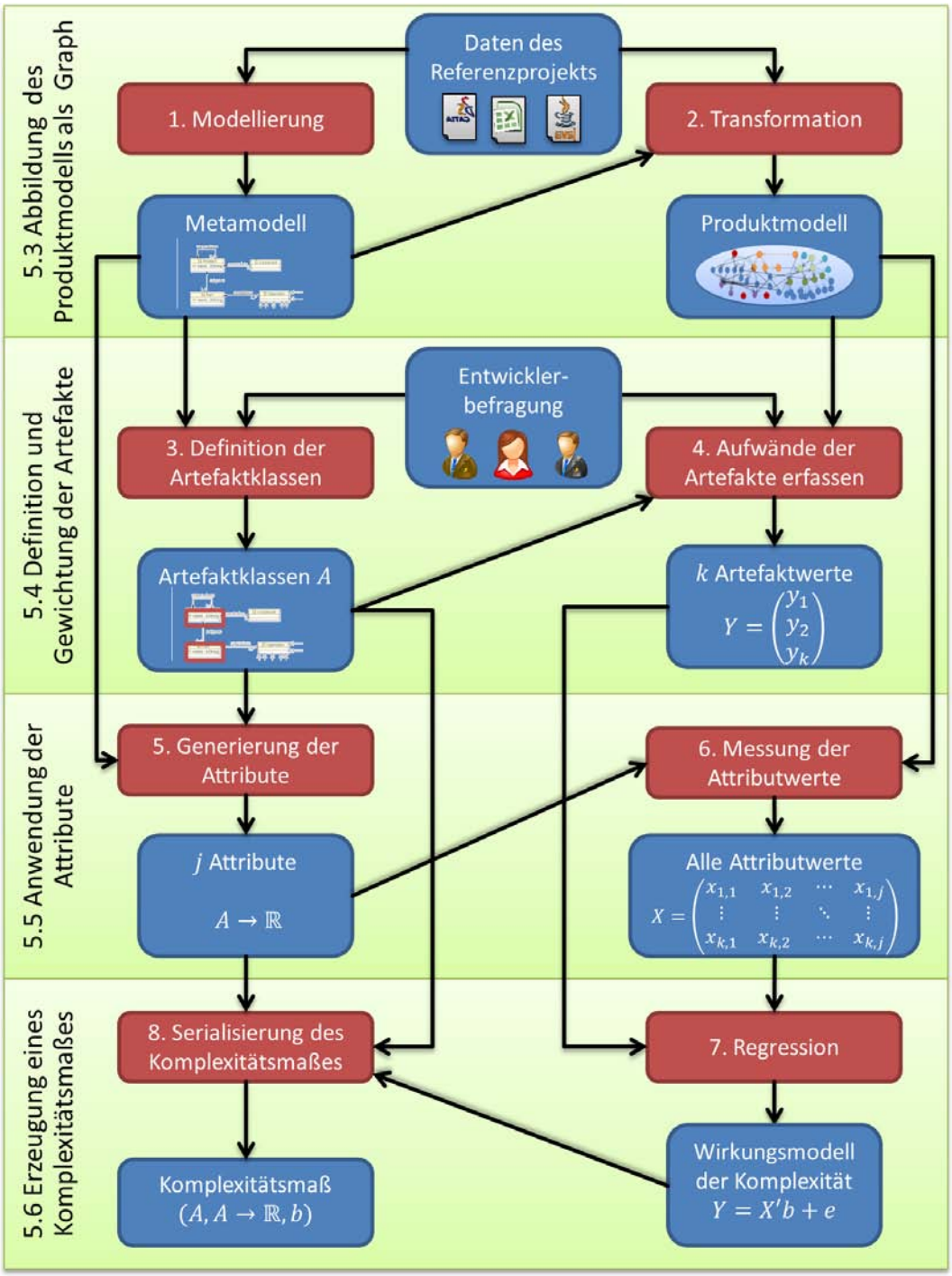


Abbildung 23: Vorgehensmodell der Erstellung eines Komplexitätsmaßes

In der ersten Phase werden die Daten des Referenzprojektes zunächst in ein geeignetes Format für die Komplexitätsanalyse überführt. Hierzu wird für jedes Partialmodell ein Metamodell erstellt (**1. Modellierung**) und anschließend erfolgt auf Basis dieses Modells die Umwandlung der nativen Daten in ein graphbasiertes Format (**2. Transformation**). Diese Phase wird in Abschnitt 5.3 erläutert.

Die zweite Phase definiert, welche Elemente des Referenzproduktmodells Artefakte sind, die eine Komplexität besitzen (**3. Definition der Artefaktklassen**). Diese Kom-

plexität wird anhand von Entwicklerbefragungen über den verbundenen Aufwand erfasst (**4. Aufwände der Artefakte erfassen**). Diese Phase wird in Abschnitt 5.4 erläutert.

In der dritten Phase werden Attribute auf Basis des Metamodells generiert (**5. Generierung der Attribute**), die Eigenschaften von Artefakten im Produktmodell metrisch beschreiben. Die Werte jedes Attributs für jedes Artefakt werden in einer Matrix erfasst (**6. Messung der Attributwerte**). Diese Phase wird in Abschnitt 5.5 erläutert.

Die letzte Phase stellt den Kern des Ansatzes dar. Durch eine Analyse der Daten aus der vorherigen Phase werden geeignete Attribute selektiert und mit diesen eine Regressionsanalyse bezüglich der Auswirkung auf die Komplexität durchgeführt und das Ergebnis in ein Modell abgebildet (**7. Regression**). Ein wichtiges Element ist hierbei die statistische Qualitätsbewertung des Prognosemodells und die inhaltliche Plausibilitätsprüfung durch den Benutzer. Ein plausibles Modell mit den zugehörigen Attributen und Artefaktdefinitionen stellt ein Komplexitätsmaß dar. Dieses kann nach der Transformation eines zu vergleichenden Produktmodells automatisiert angewendet werden (**8. Serialisierung des Komplexitätsmaßes**). Diese Phase wird in Abschnitt 5.6 dargestellt.

### 5.2.2 Übersicht zur Anwendung des Komplexitätsmaßes

Die Abbildung 24 stellt das Vorgehen zur Anwendung eines Komplexitätsmaßes dar. Die Semantik der Elemente entspricht der in Abbildung 23. Im Gegensatz zur Erstellung eines Komplexitätsmaßes umfasst dieses Vorgehensmodell jedoch nur zwei Phasen. Mögliche Analysen, die auf der Messung der Komplexität beruhen, sind durch orangene Elemente dargestellt.

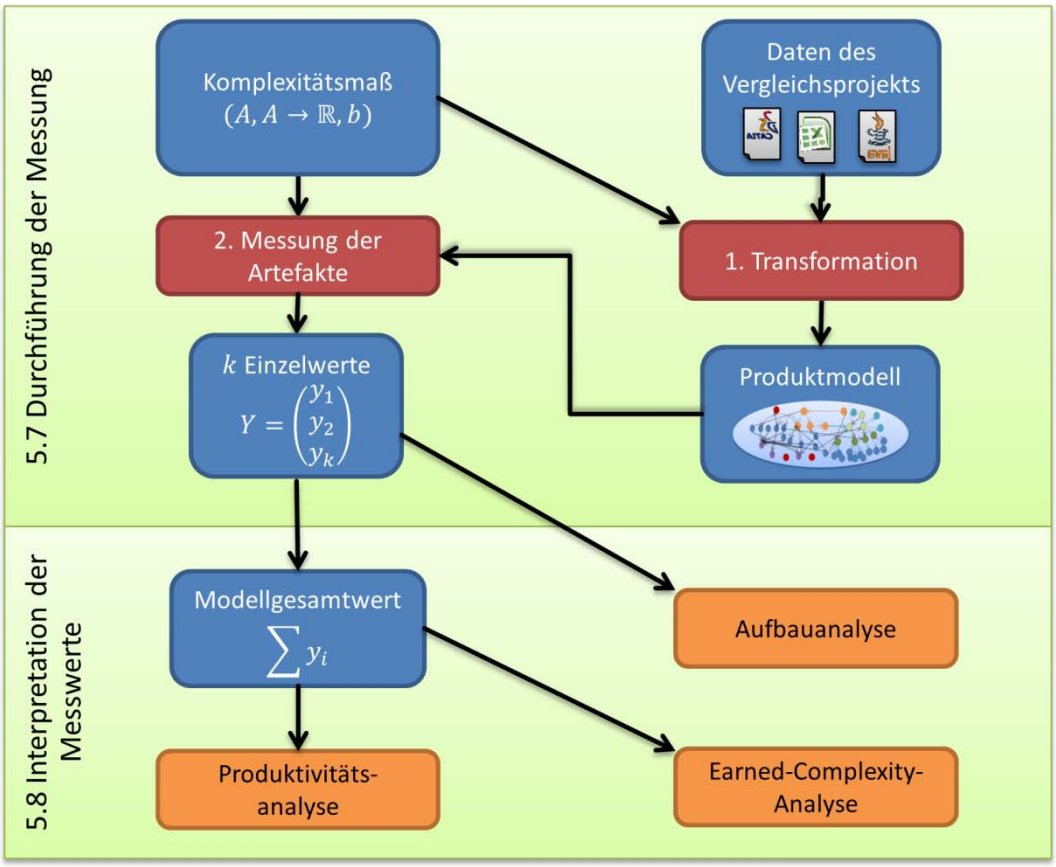


Abbildung 24: Vorgehensmodell der Anwendung eines Komplexitätsmaßes

In der ersten Phase der Anwendung wird die eigentliche Messung auf Grundlage des erstellten Komplexitätsmaßes durchgeführt. Hierzu muss zunächst das zu vergleichende Projekt im ersten Schritt in ein graphbasiertes Produktmodell umgewandelt werden (**1. Transformation**). Auf Grundlage der vereinheitlichten Darstellung können dann schließlich die Werte der Attribute der Artefakte ermittelt und die Komplexität ermittelt werden (**2. Messung der Artefakte**). Diese Phase wird in Abschnitt 5.7 beschrieben.

Die zweite Phase beschreibt mögliche Anwendungsszenarien für das Komplexitätsmaß. Die in Abschnitt 5.8 gezeigten Analysen nehmen Bezug auf die Ausgangsfragestellungen des Ansatzes sowie den Stand der Technik. Eine Abgrenzung zum Stand der Technik findet des Weiteren separat in Abschnitt 5.9 statt.

## 5.3 Abbildung des Produktmodells als Graph

In der Produktentwicklung werden zahlreiche Werkzeuge und Methoden eingesetzt, die Informationen über das zukünftige Produkt oder den Entwicklungsprozess verwenden. Diese Informationen werden in dem vorgestellten Ansatz benötigt, müssen jedoch vor der Verwendung in ein geeignetes graphbasiertes Format überführt werden. Die Grundlagen dieser Abbildung von nativen Datenquellen zu einer geeigneten Form werden in diesem Abschnitt behandelt. Zunächst werden die Anforderungen des Ansatzes an das

Graphenmodell und somit an die zur Implementierung verwendeten Technologien (siehe auch Abschnitt 6.1) präzisiert. Danach wird die Erstellung der Metamodelle für die Produktdaten betrachtet. Anschließend wird die eigentliche Abbildung konkreter Daten aus dem Entwicklungsprozess auf einen Graphen diskutiert, welche anhand von Modellen des durchgängigen Beispiels demonstriert wird. Für jede Datenquelle gibt es zunächst einen separaten Graphen, der Partialmodell genannt wird. Die Aufbereitung der einzelnen Partialmodelle zu einem integrierten Modell wird abschließend diskutiert. Dieses integrierte Modell stellt die Voraussetzung für die darauf folgenden Schritte dar.

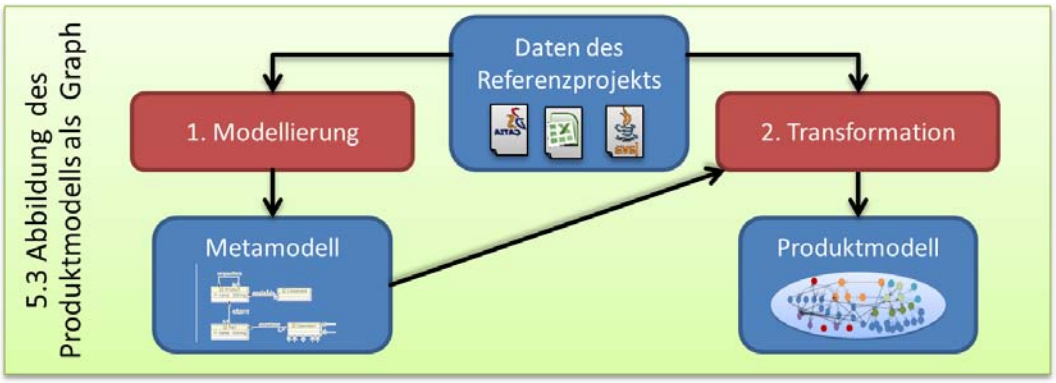


Abbildung 25: Schritte und Modelle der ersten konzeptionellen Phase

### 5.3.1 Eigenschaften des Graphen

Im Kapitel 4 wurden verwandte Ansätze der Strukturanalyse von Produktmodellen vorgestellt, die auf verschiedenen Graphrepräsentationen basieren. Begleitend wurden kurz wichtige Eigenschaften von Graphen vorgestellt. Da der Ansatz implementierungsunabhängig ist und Bezüge zu anderen Ansätzen herstellbar sein sollen, werden in diesem Abschnitt kurz die Eigenschaften des Graphen dieses Ansatzes detailliert.

Die Elemente der Produktentwicklung wie Produktbestandteile, Konzepte, Ideen, Personen und Aufgaben werden durch die Knoten dargestellt. Die Beziehungen zwischen diesen Elementen werden durch gerichtete Kanten repräsentiert. Da zwischen Elementen mehrere Beziehungen bestehen können, handelt es sich um einen gerichteten Graphen mit Mehrfachkanten. Entsprechend der Komplexitätsdefinition können bestimmte Elemente mit einem Komplexitätswert versehen werden. Dieses wird als spezielle Knotenuntermenge mit einer numerischen Abbildung dargestellt. Da die Komplexität eng mit dem Aufwand verknüpft ist, kann dieser Wert nicht negativ sein. Das Produktmodell besteht somit aus folgenden Elementen:

- Ein gerichteter Multigraph $(V, E)$
- Eine Untermenge Artefakte $A$ der Knoten $A \subset V$
- Eine Gewichtungsfunktion $g$ auf den Artefakten $g: A \mapsto \mathbb{R}^+$

Der Ansatz berücksichtigt mehrere Arten von Relationen und beruht daher auf benannten Kanten, wobei alle Kanten auf eine definierte Menge von Relationen $R$ abgebildet werden. Außerdem werden alle Knoten auf eine definierte Menge von Klassen $C$ abgebildet. Die Relationen und Klassen sind in der Regel durch das Metamodell der Domäne definiert und beschreiben unter anderem, welche Relationen zwischen welchen Klassen möglich sind. Es werden somit zwei weitere Funktionen benötigt.

- Eine Relationsfunktion $r: E \mapsto R$
- Eine Klassenfunktion $c: V \mapsto C$

Ein optionales Element der Implementierung stellt die Berücksichtigung von einfachen Knoteneigenschaften dar. Dieses können zum Beispiel Gewicht, Freigabestatus oder Name sein.

### 5.3.2 Definition eines Metamodells

Das Metamodell wird entweder von bestehenden Metamodellen in der Domäne wie zum Beispiel Dateispezifikationen, Austauschstandards oder Ontologien abgeleitet oder in Interviews mit den Benutzern konstruiert. Dies muss nur einmal zur Definition eines Komplexitätsmaßes durchgeführt werden. Welche Sachinhalte das Metamodell beschreibt, hängt von der jeweiligen Entwicklungsumgebung ab. Dies folgt aus der aufwandsbezogenen Komplexitätsdefinition, da unterschiedliche Eigenschaften der Entwicklungsumgebung Unterschiede im benötigten Aufwand zur Entwicklung bedingen. Wären beim Beispiel des Gewerbespülers mit der Entwicklung nur Ingenieure betreut, die bereits weitreichende Erfahrungen mit Designküchen besitzen, so würde die Arbeitsaufteilung keine Bedeutung besitzen und nicht notwendigerweise im Metamodell abgebildet. Andererseits spielen in dem Beispiel offensichtlich die Verknüpfungen mit Designanforderungen eine wichtige Rolle, weswegen diese abgebildet werden. Durch die jeweilige Festlegung der komplexitätsbestimmenden Umwelt werden die Anforderungen Individuelles Komplexitätsmaß und Domänenunabhängigkeit adressiert.

Für die Ausarbeitung wird in dem Beispiel CATIA V5 verwendet um die Geometrie und das Material der Komponenten des Produktes sowie dessen Zusammenbau zu beschreiben. Einzelne Komponenten wie die Bestandteile des Hebezylinders werden durch CATPart-Dateien beschrieben. Der Zusammenbau dieser Komponenten wird über die Definition von Bedingungen in Product-Dateien beschrieben. Product-Dateien können außerdem hierarchisch organisiert werden. Für das dritte Validierungsszenario dieser Arbeit werden außerdem weitere Details der geometrischen Modellierung in CATIA betrachtet und in Abschnitt 7.3.1 beschrieben. Für die Entwicklung des Konzeptes soll an dieser Stelle nur die Struktur der CATPart- und Product-Dateien betrachtet werden.

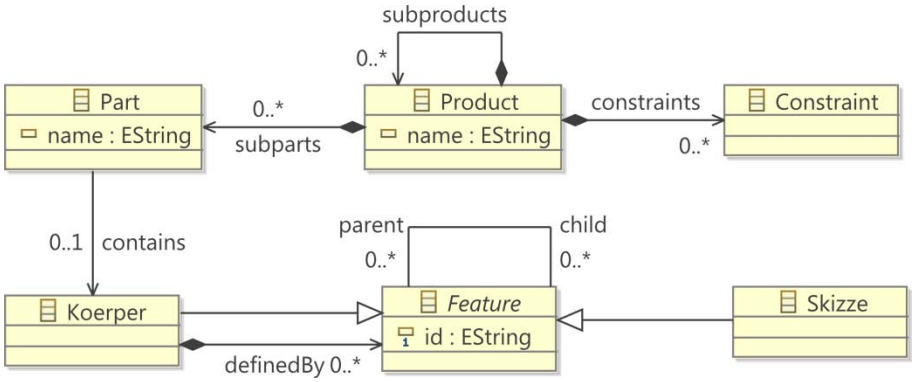


Abbildung 26: Metamodell der CATIA-Daten

Ein Metamodell der CATIA-Dateien ist in UML-Syntax in Abbildung 26 dargestellt. Dieses Modell umfasst natürlich nicht alle Daten, die im nativen Format vorliegen; es liegt also eine verlustbehaftete Abbildung vor. Das Metamodell stellt also eine Reduktion auf die vermutlich wesentlichen Aspekte der Komplexität für diese Produktdaten dar. Sollte bei weiteren Untersuchungen festgestellt werden, dass andere Relationen oder Eigenschaften relevant sind, müssten diese zuerst im Metamodell ergänzt und neue Komplexitätsmaße erstellt werden.

### 5.3.3 Transformation der nativen Daten

Die Daten aus dem Entwicklungsprozess müssen für die Kalibrierung eines Komplexitätsmaßes und die Ausführung einer Messung in Partialmodellgraphen überführt werden, die den zugrundeliegenden Metamodellen der Partialmodelle entsprechen. Diese Transformationen sollten automatisch durchgeführt werden können, um die Anforderung PA1 (Minimierung des Messaufwandes) zu erfüllen. Die Komplexitätskennzahl kann so in automatisierten Reporting-Tools verwendet werden. Allerdings ist auch eine manuell unterstützte Messung in regelmäßigen Abständen denkbar.

Die Transformationen können sich in ihrem Ansatz und den verwendeten Technologien unterscheiden. Eine wichtige Rolle spielen auch die Eigenschaften der Zielsprache, mit der die Graphen beschrieben werden. Einige häufig vorkommende Strategien und Techniken können jedoch identifiziert werden.

- **Parser:** Dies sind Programme, die ein Quellformat in ein Zielformat, hier das Graphenmodell, überführen. Sie können im Rahmen eines Frameworks automatisch angestoßen werden um aus Entwicklungsverzeichnissen Dokumente zu laden und zu transformieren. Parser können für jede mögliche Art von Eingabe programmiert werden.
  In der Regel kann man zur Entwicklung Programmbibliotheken oder Werkzeuge nutzen. Zum Beispiel liegt bei vielen Quellformaten eine Spezifikation in

Form einer formalen Grammatik[6] vor. In diesen Fall können Parsergeneratoren wie ANTLR (Parr und Quong 1995) eingesetzt werden um die Grundstruktur eines Parsers automatisch zu erzeugen.

Viele Quellformate basieren auf einer XML-Syntax. Die Daten können dann durch spezielle XML-Parser-Frameworks verarbeitet werden. Transformationen können in einigen Fällen auch mit XSLT[7] programmiert werden. XSLT-Parser parsen mit Hilfe dieser Spezifikation XML-Dokumente in ein gegebenes Zielformat.

- **Austauschformate:** In einigen Fällen liegt keine Spezifikation des nativen Dateiformates eines Entwicklungswerkzeuges vor. In der Regel kann jedoch aus dem Programm in ein Austauschformat exportiert werden, welches dann in einem zweiten Schritt zum Zielformat transformatiert wird. Zum Beispiel ist das CATPart-Format für die CAD-Modelle des Gewerbespülers proprietär. CATIA verfügt jedoch über Exportfunktionen in zahlreiche Austauschformate. Der Export ist jedoch schwerer zu automatisieren.
- **Stufenweise Reduktion:** Die Ausgangsdaten sind bei vielen Werkzeugen sehr umfangreich. Bei der Konstruktion des Zielgraphen ist es jedoch meistens notwendig das gesamte Quellmodell im Speicher zu halten. Daher kann es manchmal sinnvoll sein, dieses vorher in ein reduziertes temporäres Modell umzuwandeln.

Die Umwandlung in Graphen wird anhand des durchgehenden Beispiels mit den Partialmodellen Anforderungsmodell und Ausarbeitung (CATIA) demonstriert. Im Anschluss wird die Verknüpfung von Partialmodellen diskutiert und inwieweit sich diese Verknüpfung automatisieren lässt.

#### 5.3.3.1 Beispiel: Transformation der Anforderungen

Die Anforderungsliste liegt zunächst als Tabelle vor, die in einem Tabellenkalkulationsprogramm verwaltet wird. Diese Daten können entweder direkt durch eine Programmbibliothek wie Apache POI[8] aus dem Dokument eingelesen werden oder werden zunächst in ein leicht einzulesendes Format wie semikolonseparierte Listen umgewandelt. Die Anforderungen im laufenden Beispiel sind in Tabelle 5 zusammengefasst.

---

[6] Formale Grammatik ist ein Begriff aus der theoretischen Informatik. Durch formale Grammatiken können sogenannte formale Sprachen beschrieben werden. In der Regel sind Dateiformate formale Sprachen, da sie von einem Programm beim Laden geparst werden.

[7] Die Spezifikation von XSLT liegt unter http://www.w3.org/TR/xslt20/#xslt-mime-definition vor.

[8] Projektseite, Dokumentation und Download unter http://poi.apache.org/

Tabelle 5: Anforderungsliste des Gewerbespülers

| **Nr.** | **Bezeichnung** | **Werte, Daten, Erläuterungen** | **Quelle** | **Abhg.** |
|---|---|---|---|---|
| **1** | Einsatzbereich | mittelständische Gastronomiebetriebe | Gruppe | |
| **2** | Komfort. Geschirrentnahme | Korb auf 850mm (kein Bücken) | Nutzer | |
| **3** | Dreckablagerung | keine Ablagerungen im Innenraum | Nutzer | |
| **4** | Abmaße | 598x600x850mm nach Herstellerprogramm | Hersteller | |
| **5** | Spülmittelbehälter | zentraler Behälter für alle Spülvorgänge des Tages | Nutzer | 4 |
| **6** | Waschdauer | 1 bis 5 Min. | Nutzer | 5 |
| **7** | Rückstände | keine Schmutz-/Kalkablagerungen | Nutzer | |
| **8** | robust und zuverlässig | große Wartungsintervalle | Nutzer | 2, 7 |
| **9** | Design | innovatives Design für Einsatz im Thekenbereich | Gruppe | |
| **10** | Bedienbarkeit | kurze Einarbeitungszeit, oft wechselndes Personal | Nutzer | |
| **11** | Preis | max. 6000 Euro | Nutzer | |

Aus jeder Zeile wird im Rahmen der Transformation ein Knoten erstellt. Die Spaltenwerte können entweder als Knoteneigenschaft oder als Kante zu einem anderen Knoten implementiert werden. Zum Beispiel kann die Spalte „Quelle“ sehr gut durch zusätzliche Knoten abgebildet werden, da nur wenige verschiedene Werte vorliegen. Die letzte Zeile gibt Abhängigkeiten zwischen den einzelnen Anforderungen wieder. Solche Informationen werden oftmals in einer separaten Tabelle, wie sie in Abschnitt 4.1 beschrieben wurden, dargestellt. Eine solche Darstellung ist ebenfalls sehr einfach in einen Graphen zu übersetzen. Das Ergebnis der Transformation ist der Graph in Abbildung 27. Der Einfachheit halber sind die Knoten mit der Anforderungsnummer beschriftet und die Bezeichnungen nicht dargestellt. Die unterschiedlichen Arten von Knoten sind durch die Legende erläutert.

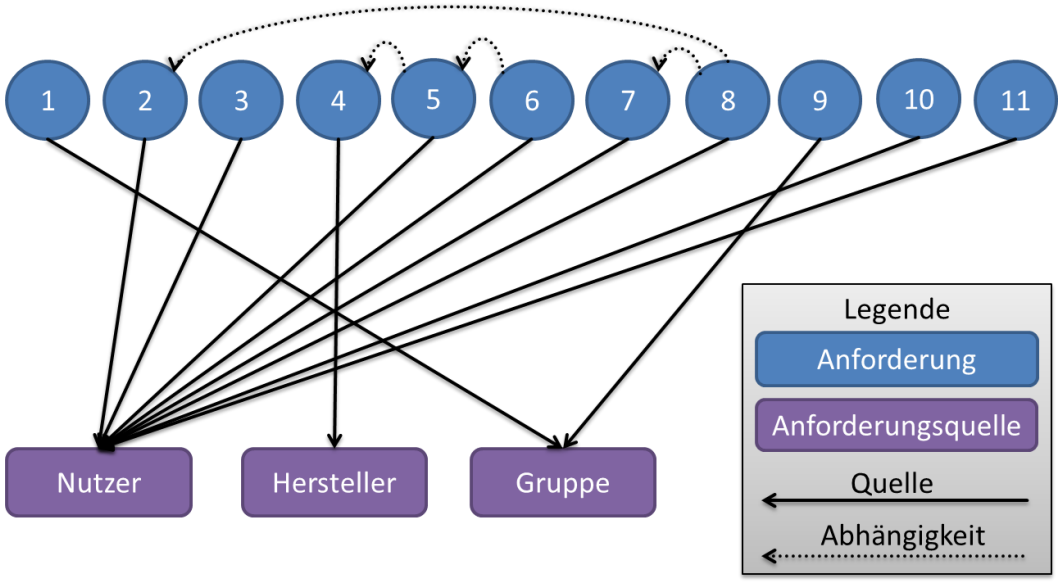


Abbildung 27: Anforderungen des Gewerbespülers als Graph

Anhand des bereits vereinfachten Beispiels für dieses kleine Partialmodell wird deutlich, dass durch die Transformation sehr umfangreiche Graphen entstehen. Bei anderen Partialmodellen können tausende Kanten und Knoten durch die Transformation entstehen. Der Ansatz ist somit von anderen Ansätzen der Strukturanalyse (Kapitel 4) zu unterscheiden, die Graphendiagramme zur Aufbereitung von Informationen für die menschliche Nutzung verwenden. Die Abbildung 27 ist lediglich ein Beispiel für das Datenmodell, das dem Ansatz zu Grunde liegt, und nicht für die Darstellung gegenüber dem Benutzer. Die Daten sollten mit Hilfe einer leistungsfähigen Programmbibliothek verwaltet werden, die die in Abschnitt 5.3.1 beschriebenen Eigenschaften der Graphen unterstützt. In Abschnitt 6.1 werden hierzu einige Alternativen diskutiert.

Eine vereinfachte Ansicht für den Benutzer kann aber zur Kontrolle der Transformation optional implementiert werden. Dabei sollte der Benutzer die Möglichkeit haben Modellaspekte zu verbergen oder sich auf bestimmte Knoten zu konzentrieren. Eine Repräsentation der gesamten Daten ist nur in sehr einfachen Fällen sinnvoll.

#### 5.3.3.2 Beispiel: Transformation des CATIA-Modells

Das CATIA-Modell liegt in einem proprietären Format vor, was die Transformation zunächst erschwert. Wie bereits erwähnt kann hier der Umweg über ein Austauschformat gewählt werden oder über Werkzeuge von Drittanbietern. Alternativ lassen sich in CATIA Skripte programmieren, die die Daten in einem einfach zu verarbeitenden Format exportieren. Für den Ansatz zur Komplexitätsmessung sind nur einige der Daten in den Modellen relevant. Insbesondere sind die geometrischen Details in diesem Beispiel nicht relevant, was bereits im Metamodell des Partialmodells festgelegt wurde. Von Interesse ist die generelle Struktur des Produktes, welche in CATIA durch einen Baum

dargestellt wird, dessen Blätter unter anderem die CATPart-Dateien und Bedingungen sind. Die Abbildung 28 zeigt die Produktstruktur des Gewerbespülers unter anderem mit der CATPart-Datei „Aussen“ zur Definition des Außengehäuses.

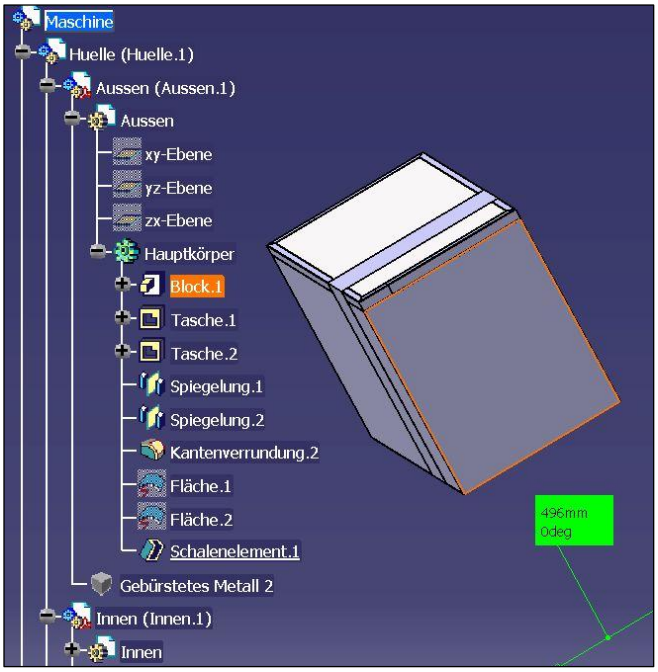


Abbildung 28: Produktstruktur des Gewerbespülers

Die Abbildung in das Graphenmodell basiert auf dem zuvor in Abbildung 26 beschriebenen Metamodell der CATIA-Dateien. Die Product- und CATPart-Dateien bilden die wichtigsten Knoten; alle anderen Knoten dienen vor allem der Beschreibung von Bedingungen. Zunächst werden diese Dateien durch Knoten abgebildet und die Produktstruktur durch Kanten definiert. Danach werden Details der Product- und CATPart-Dateien durch weitere Knoten und Kanten ergänzt. Diese Beschreibungen dienen nur der Charakterisierung der Dateien für diesen Ansatz und reichen nicht mehr aus um die tatsächliche Geometrie nachzuvollziehen.

Die Abbildung 29 gibt einen Ausschnitt aus dem resultierenden Graphenmodell wieder. Die hierarchisch orientierte Darstellung dient nur der Verdeutlichung der Beziehungen zwischen dem nativen Modell und dem Graphenmodell an dieser Stelle und ist nicht in den Daten des Graphenmodells vorhanden.

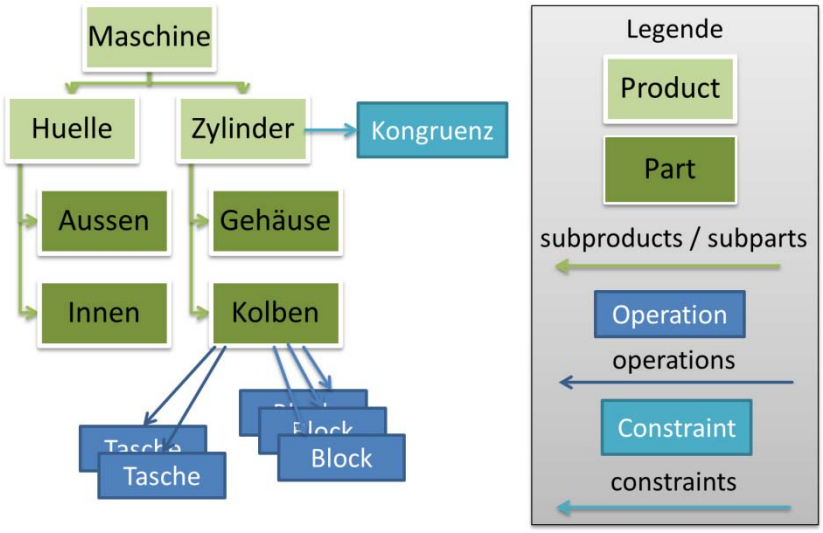


Abbildung 29: Graphenmodell der Catia-Dateien

#### 5.3.3.3 Integration mit partialmodellübergreifenden Kanten

Nach der Transformation der Entwicklungsdaten liegt eine Menge von Partialmodellen vor, die nicht miteinander verknüpft sind. Die bisher erfassten Modelle stellen also „Dateninseln“ dar. Diese Inseln gilt es in diesem Schritt zu verknüpfen. Relationen zwischen verschiedenen Partialmodellen spielen für die Komplexität eine wichtige Rolle. Dies wurde in Abschnitt 2.2.1.2.3 herausgestellt und in der Anforderung PA3 (Partialmodellübergreifende Beschreibung) formalisiert. So lässt sich in dem laufenden Beispiel vermuten, dass die Realisierung einer Anforderung aus dem Anforderungspartialmodell durch einen Knoten des CATIA-Partialmodells wesentlich die Komplexität dieses Knotens beeinflusst. Um dies zu untersuchen, müssen die Kanten zwischen diesen Partialmodellen beschrieben werden. Sind die Verknüpfungen gezogen, so werden die Partialmodelle zu einem einzigen Graphen, dem integrierten Modell der Produktentwicklung, vereinigt. Dieses Vorgehen wird durch Abbildung 30 anhand der Partialmodelle Anforderungen, Funktionen und Ausarbeitung zusammengefasst (vergleiche hierzu Abbildung 21). Das erzeugte Modell stellt die gesamte Entwicklung zu einem definierten Zeitpunkt dar.

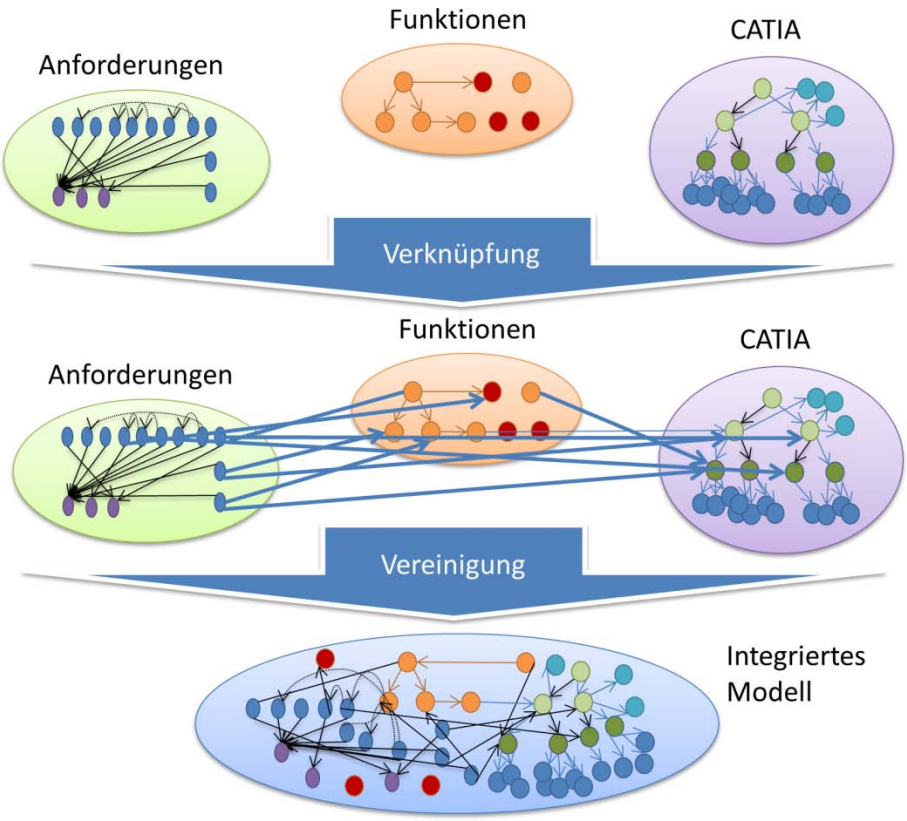


Abbildung 30: Konstruktion des integrierten Modells

Die Verknüpfung der Partialmodelle ist der aufwändigste Schritt und erfordert ein sorgfältiges Vorgehen um eine hohe Qualität der Daten zu erreichen. Nur so kann für die Messerzeugung und Messdurchführung Genauigkeit und Signifikanz gewährleistet werden. Um die Daten zu sammeln bieten sich folgende Optionen an.

- **Experteninterviews und Arbeitssitzungen (Workshops):** Diese Option ist vor allem notwendig, wenn es sich um implizites Wissen handelt. Die Informationen sind in keinem Dokument hinterlegt und müssen erst durch Interviews oder Workshops erhoben werden. In einigen Fällen muss erst die Schwierigkeit überwunden werden, diese Relationen zu beschreiben. Diese Erfassung macht nicht nur im Rahmen dieses Ansatzes der Komplexitätsmessung Sinn, sondern ist generell zu empfehlen um ein besseres Verständnis der Entwicklungsprozesse zu erhalten. (Lindemann, Maurer, und Braun 2008, 79) betonen den Zeitaufwand dieser Maßnahme und empfehlen eine gewissenhafte Vorbereitung und Dokumentation von Arbeitssitzungen. Die Experten stehen nur für kurze Zeit zur Verfügung und sollten jeweils nur die für sie relevanten Aspekte betrachten. Ansonsten kann die Qualität der Ergebnisse durch Ablenkung oder Erschöpfung der Teilnehmer leiden.
  Experteninterviews sind insbesondere geeignet um die Arten von Relationen festzustellen, die zwischen Elementen aus verschiedenen Partialmodellen bestehen können. Einige Relationen sind nicht offensichtlich und werden erst bei in-

tensiverer Diskussion beschrieben. Die Ergebnisse können gut in Form einer Kreuztabelle von Klassen beschrieben werden, in deren Zellen die möglichen Relationen eingetragen werden. Für das laufende Beispiel wird dies anhand der Tabelle 6 demonstriert. Die Richtung der Relationen ist von Y-Achse nach X-Achse zu lesen. Die Relationen `implements` und `depends` sind in diesem Beispiel nicht genauer differenziert, können aber auf Grundlage von Expertenmeinungen definiert werden um Missverständnisse zu vermeiden. In der Tabelle sind außerdem einige Relationen innerhalb von Partialmodellen dargestellt.

Tabelle 6: Relationsarten zwischen Partialmodellen

| Classes | Req. | Function | Part | Product |
|---|---|---|---|---|
| **Requirement** | includes<br>extends | | | |
| **Function** | implements | depends | | |
| **Part** | depends | implements | depends | |
| **Product** | depends | implements | contains | subproduct |

- **Extraktion aus bestehenden Daten:** Oftmals werden bereits Tools eingesetzt, die Verknüpfungen zwischen Partialmodellen verwalten. Ein Beispiel hierfür ist das Anforderungsverwaltungswerkzeug DOORS[9], das die Verfolgung der Implementierung von Anforderungen unterstützt. Insbesondere im Bereich Software Engineering existieren zahlreiche solcher Anforderungsverfolgungswerkzeuge (Traceability tools) und Ansätze. Häufig setzen diese auf die im Abschnitt 4.1 beschriebenen Matrixdarstellungen auf. (Eichinger u. a. 2006) beschreiben eine softwareunterstützte Methode zur Ableitung und Darstellung von Relationen zur Unterstützung des in Abschnitt 4.1.2 vorgestellten Design-Structure-Matrix-Ansatzes.
  Einige Partialmodelle verfügen über unidirektionale Verknüpfungen zu anderen Partialmodellen. Zum Beispiel können Annotationen in einigen Modellen ausgewertet werden, die bestimmte Relationen wie Abhängigkeiten zu externen Elementen informell beschreiben.
- **Halbautomatische Verknüpfung:** Diese Vorgehensweise beruht darauf, dass dem Benutzer automatisiert Vorschläge zur Verknüpfung von Daten vorgelegt werden. Die Vorschläge werden anhand der vorhandenen Partialmodelle durch

[9] http://www-01.ibm.com/software/awdtools/doors/productline/

eine Software erzeugt. Die Permeter-Software (siehe Abschnitt 4.2.1) beinhaltet zum Beispiel eine Komponente zum Vorschlagen von Verknüpfungen auf Basis von Knotenbeschriftungen (Hausmann 2008, 100). Dieses Vorgehen ist empfehlenswert, wenn Strukturen in einem Partialmodell einer späteren Entwicklungsphase eine Abbildung von Strukturen eines früheren Partialmodells darstellen. Dieses ist insbesondere bei einigen Synthesetools wie UML-Java-Generatoren der Fall.

Die Wahl des Vorgehens hängt von den Partialmodellen und von dem vertretbaren Aufwand für die Komplexitätsanalyse ab. Es empfiehlt sich die Verknüpfungen zunächst in separaten Modellen zu verwalten um sie für mögliche Änderungen zu verwalten und erst bei der Vereinigung aller Graphen zum integrierten Modell diese Trennung aufzugeben.

Das Ergebnis der Transformation, Verknüpfung und Vereinigung der Partialmodelle ist ein Abbild des Produktes mit allen aufwandsrelevanten Informationen als einzelner Graph. Die Anzahl der Knoten und Kanten kann bei einigen Produkten sehr umfangreich sein. Eine Programmbibliothek für Graphen, die zur Implementierung des Verfahrens eingesetzt wird, sollte daher ein leistungsfähiges Datenmodell besitzen.

## 5.4 Definition und Gewichtung der Artefakte

Mit der Erstellung des integrierten Produktmodells und des zugehörigen Metamodells liegt eine Datenbasis für die Ableitung der Attribute der Artefakte vor, die im zugrundeliegenden statistischen Modell die unabhängigen Variablen darstellen (siehe S. 77). In dieser Phase werden mit der Festlegung der Komplexitätswerte für die Artefakte des Referenzproduktmodells die abhängigen Variablen definiert. Diese Werte sind über den Aufwand für die Erstellung des Artefaktes definiert, was sich aus den verwendeten Definitionen zur Komplexität ergibt. Dies bedeutet, dass für die Kalibrierung des Komplexitätsmaßes zuvor entstandene Aufwände als Komplexitätswerte für die Artefakte des Referenzmodells bestimmt werden müssen (Schritt 4). Hierfür können verschiedene Quellen verwendet werden, die in diesem Abschnitt mit Vor- und Nachteilen diskutiert werden. In der Regel handelt es sich um eine Befragung der Entwickler der Artefakte. Zuvor muss jedoch festgelegt werden, welche Knoten Artefakte darstellen und mit einem Komplexitätswert bewertet werden können (Schritt 3). Außerdem wird in diesem Abschnitt kurz die Abbildung der erfassten Komplexitätswerte in dem Regressionsmodell diskutiert.

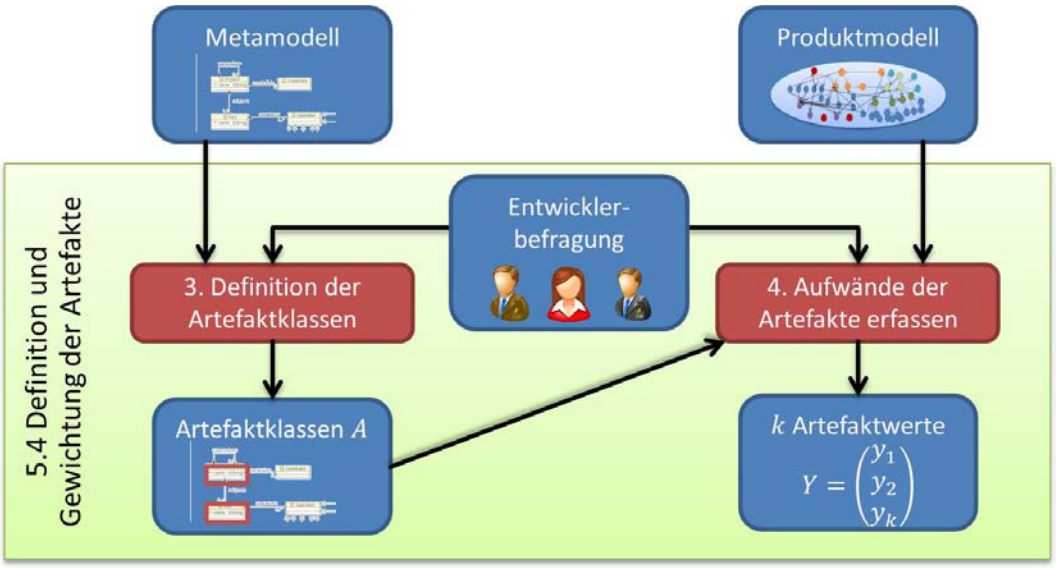


Abbildung 31: Schritte und Modelle der zweiten konzeptionellen Phase

### 5.4.1 Definition der Artefaktklassen

Im Produktmodell sind viele Elemente des Produktes und gegebenenfalls auch der Entwicklungsumgebung repräsentiert. Nicht jedes Element in diesem Modell stellt eine klar unterscheidbare Komponente des Produktes dar und daher muss zwischen Artefakten und anderen Modellelementen unterschieden werden. Ein Artefakt wurde wie folgt definiert: *„Artefakte sind Modellelemente, denen eine Komplexität zugeordnet werden kann.“* Die Entwickler des Referenzproduktes müssen entsprechend definieren, welche Arten von Komponenten durch einen bestimmten Aufwand erst erzeugt wurden. Als Hilfestellung werden einige Kriterien für Artefakte aufgestellt.

- Artefakte entstehen während des Entwicklungsprozesses und die Details und Informationen nehmen im Verlauf tendenziell zu.
- Die Gestalt ist durch die Entwickler im Projekt beeinflussbar.
- Artefakte werden als „Ergebnis“ des Entwicklungsprozesses gesehen. (Dies wird im folgenden Paragraph weiter diskutiert.)
- Artefakte sind eindeutig voneinander zu unterscheiden und besitzen einen einmaligen Identifier. Zum Beispiel wird eine Java-Klasse durch den qualifizierten Klassennamen in einer Laufzeitumgebung identifiziert.
- Wenn mehrere Instanzen von einem Artefakt existieren, so sind diese Instanzen in der Regel keine Artefakte. Wird zum Beispiel eine spezielle Schraube in CAD entwickelt und an vielen Stellen im Produktmodell eingebunden wird, so ist nur die Schraube aber nicht dessen Instanzen als Artefakt anzusehen.
- Sie sind nicht existenziell an andere Elemente gebunden. Zum Beispiel sind einfach Eigenschaften und komplexere Beziehungen keine Artefakte, da sie an die

Existenz des Trägers gebunden sind. Allerdings können Artefakte ohne andere Elemente ihre Funktion verlieren.

- Verknüpfungen zwischen Partialmodellen sind keine Artefakte.

Die Wahl der Artefaktklassen hängt stark davon ab, was als „Ergebnis“ beziehungsweise Output der Entwicklung gesehen wird. Es müssen Zwischenergebnis und Endergebnis unterschieden werden, was eine individuelle Einschätzung darstellt. Zum Beispiel können Anforderungen als Artefakte aufgefasst werden, wenn ihnen selber ein Wert zugeordnet wird. Dies kann im Falle einer konzeptionellen Entwicklung wie im begleitenden Beispiel des Gewerbespülers zutreffen. Wenn Elemente als Zwischenergebnisse definiert werden, so gehen sie nur über ein Artefakt in die Komplexität des Produktmodells ein. Eine Anforderung, die nicht ein Artefakt ist, trägt zum Beispiel zur Komplexität eines Artefakts bei, welches diese Anforderung implementiert.

Artefakte werden auf Ebene des Metamodells durch die Auswahl von Klassen definiert, die im Folgenden als Artefaktklassen bezeichnet werden. Die Instanzen dieser Klassen sind Artefakte. Zur Kalibrierung ist es also nicht notwendig jede entsprechende Instanz als Artefakt zu definieren. In der Regel reicht die Auswahl von ein bis drei Klassen pro Metamodell eines Partialmodells, wobei nicht jedes Partialmodell Artefaktklassen enthalten muss.

Im laufenden Beispiel werden an dieser Stelle der Einfachheit halber nur die CATIA-Dateien als Output aufgefasst. Die Klassen `Product` und `Part` sind entsprechend als Artefaktklassen definiert. Die Definition der Artefaktklassen ist ein Teil eines Komplexitätsmaßes und muss mit diesem abgespeichert werden.

### 5.4.2 Aufwände der Artefakte erfassen

Die Erfassung der Aufwandswerte der Artefakte des Referenzprojektes stellt die Grundlage für die Abschätzung der abhängigen Variable beziehungsweise des Regressanden in der Regressionsanalyse dar. (Die Bedeutung des Regressanden im zugrundeliegenden statistischen Modell wird auf S. 77 erläutert.) Die Qualität der Daten bestimmt maßgeblich die Qualität des resultierenden Komplexitätsmaßes. Die Erhebung muss daher sorgfältig durchgeführt, wobei sich folgende Herausforderungen stellen.

- Durch die unterschiedliche Interpretation der „Komplexität“ kann es bei der Erfassung zu Missverständnissen kommen. Die Daten sind dann unter Umständen nicht vergleichbar und somit wertlos.
- Es muss sichergestellt werden, dass die erfassten Aufwände objektiv sind und nicht durch die Entwickler verfälscht werden.
- Es müssen ausreichend viele Artefakte mit Aufwänden versehen werden um genügend Beobachtungen für die Regression zu erhalten. (Backhaus u. a. 2008, 106) empfehlen die doppelte Anzahl der Variablen in der Regressionsgleichung.

Dies sind in diesem Ansatz die signifikanten Attribute. Da diese jedoch zuerst aus einer viel höheren Anzahl generierter Attribute ausgewählt werden müssen, empfehlt sich eine möglichst hohe Anzahl an Gewichtungen von Artefakten.

Zunächst muss festgelegt werden, welche Form von Aufwand und somit welche Einheit verwendet werden soll. Die Wahl hängt von der Verfügbarkeit der Daten ab und ob diese möglichst objektiv erfassbar sind. Es ist hilfreich eine Aufwandseinheit zu nehmen, die der Entwickler direkt mit den Artefakten verbinden kann wie zum Beispiel der Arbeitsaufwand für ein Artefakt in Stunden. Erfundene Einheiten wie „Aufwandspunkte“ führen zu Missverständnissen. Da das resultierende Komplexitätsmaß die verwendete Aufwandseinheit als Größeneinheit besitzen kann (vgl. Abschnitt 2.2.1.2.4), ist eine vertraute Einheit ebenfalls zu empfehlen. Der Entwickler kann die Messergebnisse so sinnvoller zuordnen. Es kann sogar ganz vermieden werden den Begriff „Komplexität“ zu verwenden, um die Problematik der unterschiedlichen Begrifflichkeiten dieses Begriffes zu umgehen. Die nachfolgende Liste gibt eine Übersicht zu möglichen Aufwandseinheiten und deren Datenquellen.

- **Zeit (zum Beispiel in Arbeitsstunden)**: Als erste Option für die Aufwandseinheit empfiehlt sich der Aufwand der Entwickler in Stunden für ein Artefakt. Dies ist direkt mit dem Artefakt verbunden und es können oftmals bereits vorhandene Daten verwendet werden. Mögliche Datenquellen sind zum Beispiel Aufgabenverwaltungssysteme, Stundenzettel oder andere Zeitabrechnungssysteme. Zeitaufwände können auch im Nachhinein durch Selbsteinschätzung der Entwickler erfasst werden. Der Vorteil besteht vor allem in der intuitiven Verknüpfung mit dem Artefakt und die Möglichkeiten diesen Aufwand mit Werkzeugen objektiv zu erfassen.
- **Geld (zum Beispiel in Euro)**: Wenn im Unternehmen entsprechende Kostendaten vorliegen, können in Verbindung mit den Arbeitsstunden die Kosten eines Artefaktes bestimmt werden. Voraussetzung für die Anwendung dieser Einheit ist, dass die Verbindung mit dem Artefakt eindeutig hergestellt wird. Der Nachteil dieser Einheit ist, dass der Entwickler oftmals nicht mit der Kostenrechnung vertraut ist und er unterschiedliche Werte ansetzen würde als ein Mitarbeiter des internen Rechnungswesens. Entsprechend ergibt sich jedoch auch der Vorteil, dass die Kennzahl besser für das Management und das FuE-Controlling geeignet ist. Da Zeit und Geld stark verwandt sind, ist die Ausrichtung der Komplexitätsmessung somit für die Auswahl entscheidend.
- **Änderungen (zum Beispiel in Anzahl geänderter Zeilen)**: Dokumentverwaltungssysteme für die Entwicklung besitzen oft eine Versionsverwaltung. Anhand der Daten in diesen Systemen kann nachvollzogen werden, wie oft und in welchem Umfang ein Dokument wie eine Quellcodedatei in der Entwicklung abgeändert wurde. Bei Dokumenten mit vielen Änderungen kann von einem er-

höhten Aufwand ausgegangen werden, der zum Beispiel durch zahlreiche Abhängigkeiten bedingt sein kann. Es muss jedoch geprüft werden, ob die Messung der Änderungen zum Beispiel bei Binärdateien objektiv möglich ist und ob die Anzahl der Änderungen nicht maßgeblich durch andere Faktoren wie den jeweiligen Programmierstil beeinflusst wird. Der Vorteil dieser Einheit bei Erfüllung dieser Voraussetzungen ist, dass bestehende objektive Daten verwendet werden können.

- **Existierende Dateimaße**: In manchen Fällen werden in Unternehmen bereits Kennzahlen im Rahmen der Rechnungsstellung oder der Qualitätsmanagements zur Beschreibung des Umfangs eines Dokumentes genutzt. Beispiele sind Quellcode-Metriken wie Lines of Code oder Anzahl der Modellobjekte, welche die Quantität der Komplexität im Sinne der Schwierigkeit der Beschreibung beschreiben. Wenn ein linearer Zusammenhang zwischen dieser Größe und dem tatsächlichen Aufwand durch einen hohen Korrelationskoeffizient belegt werden kann, dann ist auch die Verwendung solcher Kennzahlen möglich. Die Quantifizierung der Komplexität (vgl. S. 19ff.) als Schwierigkeit der Beschreibung und Aufwand der Erzeugung ist in diesen Fällen linear korreliert und somit austauschbar im Rahmen dieses Ansatzes.

Unabhängig von der Art der gewählten Aufwandseinheit und der Datenquelle sollten die ermittelten Werte auf Plausibilität und statistische Ausreißer untersucht werden. Die Erfassung kann durch Tabellenkalkulationswerkzeuge oder andere Werkzeuge unterstützt werden.

In dem laufenden Beispiel werden die CATIA Product- und CATPart-Dateien als Artefakte bestimmt und Zeitaufwände für die Dateien aufgezeichnet. Diese Beispieldaten werden in der Tabelle 7 dargestellt. Diese Werte dienen im Folgenden zur Erläuterung der Regression.

Tabelle 7: Aufwandswerte für Artefakte des Gewerbespülers

| Product/CATPart (Artefakt) | Oberteil | Arbeits-stunden | Product/CATPart (Artefakt) | Oberteil | Arbeits-stunden |
|---|---|---|---|---|---|
| Maschine | | 4,6 | Kolben 1 | Zylinder | 2 |
| Huelle | Maschine | 2,2 | Beweglich | Maschine | 0,7 |
| Aussen | Huelle | 1,6 | Propeller | Beweglich | 1,5 |
| Innen | Huelle | 9,7 | Deckel | Beweglich | 7 |
| Zylinder | Maschine | 2 | Gestell | Beweglich | 2,3 |
| Zylindergehäuse | Zylinder | 5,3 | Kolben 2 | Beweglich | 0,9 |

### 5.4.3 Darstellung im Regressionsmodell

Durch die Definition der Artefaktklassen wird eine Untermenge der Knoten im Produktmodell $A \subset V$ definiert, wie in Abschnitt 5.3.1 beschrieben. Ebenso wurde die Gewichtungsfunktion $g$ auf den Artefakten $g: A \rightarrow \mathbb{R}^+$ durch die Erfassung der Aufwände beschrieben. Die erfassten Werte werden als abhängige Variable beziehungsweise Regressand im Rahmen der Signifikanz- und Regressionsanalyse als Vektor $Y$ dargestellt, dessen $k$ Einträge jeweils einer Beobachtung entsprechen. $Y$ ist der k-Vektor der Beobachtungswerte der abhängigen Variablen.

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix}$$

## 5.5 Generierung und Anwendung der Attribute

Ziel dieser Phase ist die Abbildung der Attribute der Artefakte im Produktmodell durch eine Matrix Als Attribut ist jede metrisch oder nominal erfassbare Eigenschaft der Knoten im graphbasierten Modell aufzufassen. Die Abbildung 32 stellt die Schritte und Modelle dieser Phase dar: Zunächst wird mit Hilfe des zugehörigen Metamodells und der Definition der Artefaktklassen auf Basis generischer Attributmuster eine hohe Anzahl an Attributen konstruiert (Schritt 5). Diese Attribute werden daraufhin auf alle Artefakte im Produktmodell angewendet und eine Matrix der Attributwerte erstellt (Schritt 6). Diese Matrix stellt die Grundlage für die Bildung des Komplexitätsmaßes dar. Im Folgenden werden die beiden Schritte detailliert und die Erstellung der Matrix formalisiert.

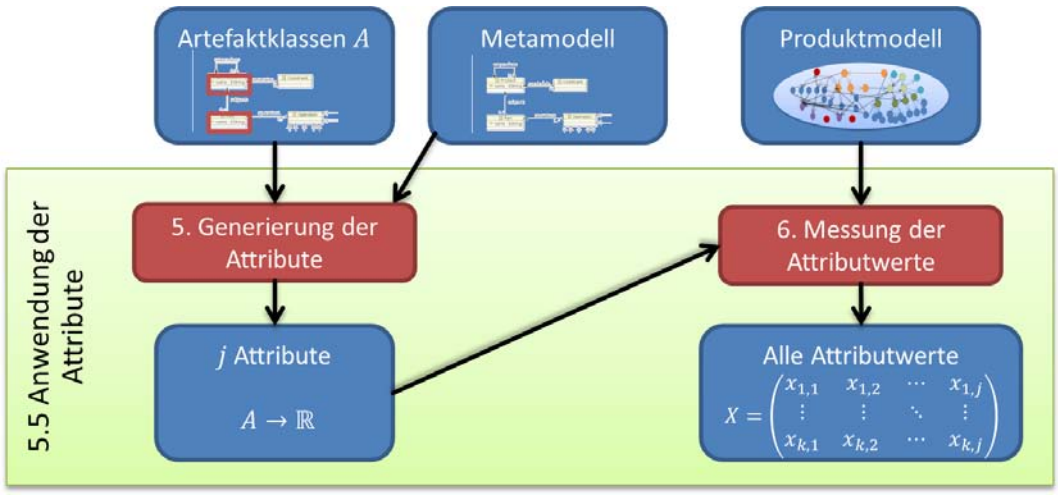


Abbildung 32: Schritte und Modelle der dritten konzeptionellen Phase

### 5.5.1 Generierung der Attribute

Durch Attribute werden Eigenschaften von Artefaktknoten im Produktmodell metrisch oder als boolescher Wert abgebildet. Hierdurch werden sie für die Regressionsanalyse erfassbar und können für die Erzeugung eines Komplexitätsmaßes berücksichtigt werden. Attribute können zum Beispiel die Anzahl ausgehender Kanten (Ausgangsgrad) sein oder mit Berücksichtigung der Graphstruktur die Tiefe eines Knotens in einem Baum. Eine boolesche Eigenschaft ist zum Beispiel die Zugehörigkeit zu einer Klasse. Durch die Einschränkung dieser Muster anhand des benannten Graphen auf bestimmte Kantentypen oder Klassen können viele Varianten erzeugt werden. Die Grundlage hierfür bildet das Metamodell des Produktmodells, das die Definition der Kantentypen und Klassen enthält sowie die Definitionen der Artefaktklassen. Mit Hilfe dieser Informationen wird pro Attributmuster eine hohe Anzahl an Instanzen erzeugt.

Im Anhang werden einige Attributmuster definiert und eine prototypische Implementierung von Attribut und Attributmustern wird in Abschnitt 6.4.2 beschrieben. Es können im Rahmen dieses Ansatzes beliebige weitere Attributmuster implementiert und verwendet werden, die zum Beispiel auch existierende Komplexitätsmaße wie die McCabe-Metrik (McCabe 1976) abbilden können. Attribute können auch auf spezielle Produktmodelle ausgerichtet werden, damit deren spezifische strukturelle Eigenschaften gemessen werden.

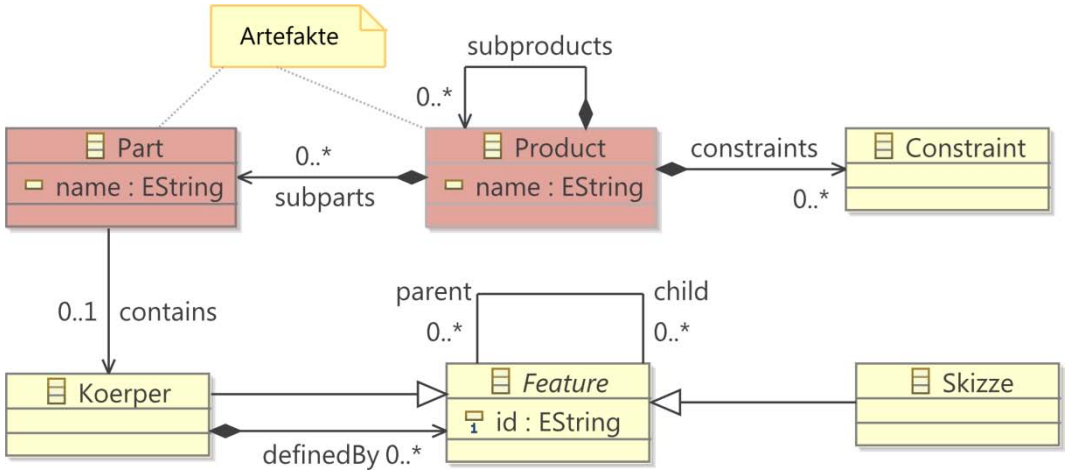


Abbildung 33: Kern des CATIA-Metamodells

Die Abbildung 33 zeigt einen Ausschnitt des CATIA-Metamodells (Abbildung 26) des laufenden Beispiels. Die Klassen `Product` und `Part` sind als Artefaktklassen definiert (in der Abbildung rot hinterlegt). Mit Hilfe der Attributmuster im Anhang können zum Beispiel folgende Attribute definiert werden:

Tabelle 8: Erzeugte Attribute auf Basis des CATIA-Metamodells

| Nr. | Attributmuster | Parameter | Bedeutung |
|---|---|---|---|
| **1** | Reference-Cardinality | - | Die Anzahl an Verknüpfungen kann mit dem Aufwand zur Erstellung der `Product`- oder `Part`-Datei zusammenhängen. |
| **2** | Reference-Cardinality | `constraints` | Bewertet die Anzahl der Constraints, die für eine `Product`- oder `Part`-Datei definiert sind. |
| **3** | Reference-Cardinality | `subparts` | Gibt die Anzahl an `Subparts` wieder. Je mehr Unterteile, desto tendenziell höher der Aufwand. |
| **4** | Reference-Cardinality | `HasOperation` | Gibt die Anzahl an CAD-Operationen an, die in einem `Part`-Dokument vorliegen |
| **5** | Reference-Cardinality | `IsImplementedBy` | Gibt die Anzahl an Anforderungen an, die durch eine `Product`- oder `Part`-Datei implementiert werden. |

Ein Attribut für die Anzahl von Zyklen durch die Relation `subproducts` wird aufgrund der Kompositionsbeziehung nicht gebildet, da `Products` nicht transitiv Unterteil von sich selbst sein können. Dies ist ein Beispiel für die Verwendung spezieller Detailinformationen des Metamodells um die Erzeugung der Attribute einzuschränken. Jedoch ist die Erzeugung von Attributen, die nicht sinnvoll angewendet werden können, nicht kritisch, da diese im späteren Verlauf auf Basis ihrer gemessenen Werte als nicht sinnvolle Attribute (siehe Abschnitt 5.6.1.5) automatisch entfernt werden.

### 5.5.2 Messung der Attributwerte

Nachdem eine Menge Attribute im vorherigen Schritt auf Basis des Metamodells und der Artefaktdefinitionen gebildet wurde, werden diese auf das Produktmodell angewendet. Jedes boolesche Attribut liefert zu jedem Artefakt einen Wert 1 oder 0. Jedes andere Attribut einen reellen Wert. Die Auswertung erfolgt automatisiert durch die Attribute, die unter Angabe der Parameter wie Relationstyp den Wert im Modell am Knoten messen. Zum Beispiel erhält der Attributtyp „Reference Cardinality“ als Parameter eine Relation wie `constraints` übergeben und konstruiert entweder eine entsprechende Instanz (Fabrikmuster) oder wertet den Parameter zur Laufzeit aus.

Da die Anzahl der Attribute durch die automatische Generierung anhand des Metamodells und der Attributmuster, wie im vorherigen Abschnitt beschrieben, sehr hoch sein kann, werden in dem laufenden Beispiel an dieser Stelle daher nur die in Tabelle 8 beschriebenen Attribute betrachtet, um das Beispiel übersichtlich zu halten. Die Abbildung 34 zeigt einen Ausschnitt aus dem Produktmodells des Gewerbespülers. Die Artefakte sind hier Grün für die Instanzen der Klassen `Product` und `Part` hinterlegt und durchnummeriert. Es wird außerdem nur eine Auswahl an Relationen dargestellt.

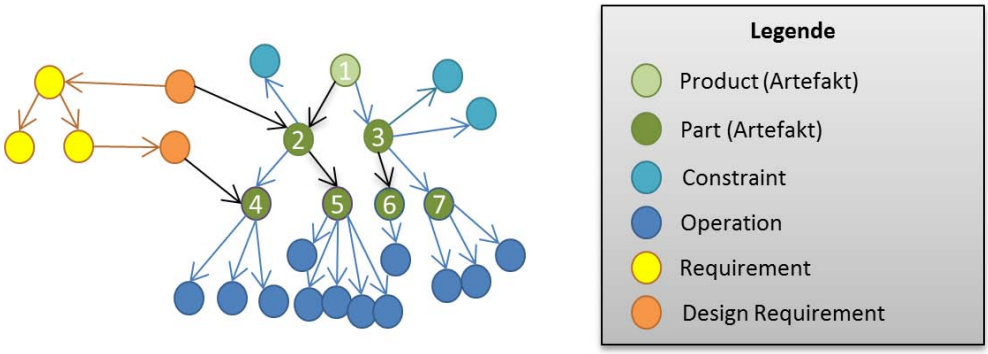


Abbildung 34: Ausschnitt des Produktmodells

Die Messung der Attributwerte wird anhand des Attributs (Reference-Cardinality, `constraints`) demonstriert: Dieser liefert für die Artefakte 1, 4, 5, 6, 7 den Wert 0 zurück, da keine ausgehende Relationen des Typs `constraints` für diese Artefakte vorliegen; für das Artefakt 2 den Wert 1 und für das Artefakt 3 den Wert 2. Diese Werte können in der vierten Spalte der Beobachtungsmatrix *X* abgelesen werden.

Tabelle 9: Attributwerte der Artefakte des Ausschnitts des Gewerbespülers

| | **Attribute** | | | | |
|---|---|---|---|---|---|
| **Artefakte** | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ |
| **1** | 2 | 0 | 0 | 0 | 0 |
| **2** | 3 | 1 | 2 | 0 | 1 |
| **3** | 4 | 0 | 0 | 0 | 0 |
| **4** | 3 | 0 | 0 | 3 | 1 |
| **5** | 5 | 0 | 0 | 5 | 1 |
| **6** | 1 | 0 | 0 | 1 | 0 |
| **7** | 3 | 0 | 0 | 3 | 0 |

Das Ergebnis der Messung aller Artefakte durch alle Attribute wird in Tabelle 9 dargestellt. Auf der vertikalen Achse sind die Nummern der Artefakte wie in Abbildung 34 eingetragen und auf der horizontalen Achse die Attribute wie in Tabelle 8. Für das Attribut 5 (Reference-Cardinality, `IsImplementedBy`) wurden nur die Designanforderungen berücksichtigt.

### 5.5.3 Darstellung im Regressionsmodell

Durch die Generierung der Attribute in Abschnitt 5.5.1 werden $j$ Abbildungen auf der Menge der Artefakte $A$ erzeugt. Die Menge der Artefakte wurde durch die Definition der Artefaktklassen in Abschnitt 5.4 definiert.

$$r_i: A \to \mathbb{R} \text{ mit } i = 0,1, \dots, j$$

Durch Anwendung aller Attribute auf alle Artefakte wird eine Matrix erzeugt, wobei jede Spalte einem Attribut und jede Zeile einem Artefakt entspricht. Der Wert eines Eintrages $x_{n,m}$ ist entsprechend definiert durch:

$$x_{n,m} = r_m(a_n) \text{ mit } a_n \in A, m = 0,1, \dots, j\ , n = 1,2, \dots k$$

Die Einträge in der Matrix sind die Ausprägungen der unabhängigen Variablen im Sinne der Regressionsanalyse. Es liegt somit die Matrix $X$ vor, die eine $(k \times j)$-Matrix der Beobachtungswerte der $j$ Attribute auf Basis der $k$ Artefakte ist:

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k,1} & x_{k,2} & \cdots & x_{k,j} \end{pmatrix}$$

### 5.5.4 Formulierung des Komplexitätsmodells

Durch die Erfassung der Schätzwerte der Entwickler in Abschnitt 5.4 und der Messung der Attributwerte am Produktmodell in diesem Abschnitt liegen die notwendigen Voraussetzungen für die Formulierung des Komplexitätsmodells vor. Mit Hilfe der Matrix $X$ der Attributbeobachtungswerte und dem Vektor der Beobachtungswerte der abhängigen Variablen $Y$ lässt sich das Wirkungsmodell des Ansatzes zunächst allgemein so formulieren.

$$Y = f(X)$$

Die Hypothese des Ansatzes lautet, dass Funktion $f$ linear ist, da die Komplexität als linearer Ausdruck des Aufwandes zur Erstellung des Produktmodells definiert wurde. In Matrixschreibweise wird dies formuliert durch

$$Y = b_0 + Xb + e$$

Wobei die Variablen wie folgt definiert sind (vgl. (Backhaus u. a. 2008, 108)):

- $Y$: Der k-Vektor der Beobachtungswerte der abhängigen Variablen. Dies ist der Aufwandsvektor der Artefakte, der im Abschnitt 5.4 auf Basis der Entwicklerbefragungen erfasst wurde.
- $X$: Die $(k \times j)$-Matrix der Beobachtungswerte der Attribute. Diese Matrix wurde in diesem Abschnitt anhand des Produktmodells erstellt.
- $b_0$: Ein k-Vektor, der als konstantes Glied in der Regressionsgleichung fungiert. Dieser Wert ordnet jedem Artefakt unabhängig von den Beobachtungswerten der Attribute eine „Grundkomplexität“ zu. Dieser Vektor kann in der Untersuchung entfallen, wenn davon ausgegangen wird, dass ein Artefakt ohne jede Information keinen Aufwand und somit keine Komplexität besitzt.
- $b$: Ist ein j-Vektor der Regressionskoeffizienten. Jeder Koeffizient charakterisiert die Bedeutung der durch das entsprechende Attribut dargestellten Knoteneigenschaft. Die Größe des Koeffizienten ist jedoch nicht mit der Bedeutung gleichzusetzen; durch eine andere Skalierung ließe sich der Wert beliebig steigern. Der Vektor $b$ wird innerhalb der Regression ermittelt.
- $e$: Ist der $k$-Vektor der Residualgrößen. Dieser Vektor stellt Abweichungen dar, die mit Hilfe der Attribute nicht erklärt werden können. Bezogen auf die Komplexität können dies Faktoren sein, die sich zum Beispiel gar nicht oder nur sehr umständlich messen lassen und keinen entscheidenden Einfluss auf die Komplexität haben. Dies kann zum Beispiel eine sehr einfache Randbedingung eines Artefaktes sein.

## 5.6 Erzeugung eines Komplexitätsmaßes

In den vorherigen Phasen wurden die notwendigen Daten ermittelt um den Zusammenhang zwischen dem Aufwand und den Eigenschaften von Artefakten quantitativ auszuwerten. In dieser Phase werden nun die vorhandenen Daten analysiert und ein ausführbares Komplexitätsmaß gebildet. Die Modelle und Schritte dieser Phase werden in der Abbildung 35 dargestellt: Den wichtigsten Schritt dieser Phase stellt die Regression (Schritt 7) dar, die zu einem Wirkungsmodell der Komplexität führt. Die Herausforderung dieses Schrittes liegt nicht in der eigentlichen Regression, sondern in der Auswahl der Attribute, die in das Regressionsmodell eingehen. Zunächst wird eine Voranalyse der Attribute durchgeführt um offensichtlich nicht geeignete und sich überschneidende Attribute zu entfernen. Den Kern bildet die Selektion der Attribute, wofür verschiedenen Verfahren diskutiert werden. Wenn in diesem Schritt ein Modell ausreichender Güte erzeugt wurde, so muss schließlich das Modell zusammen mit den zugrundeliegenden Artefaktdefinitionen und Attributen serialisiert werden (Schritt 8). Das serialisierte Maß kann schließlich in den in Abschnitt 5.7 diskutierten Anwendungen eingesetzt werden.

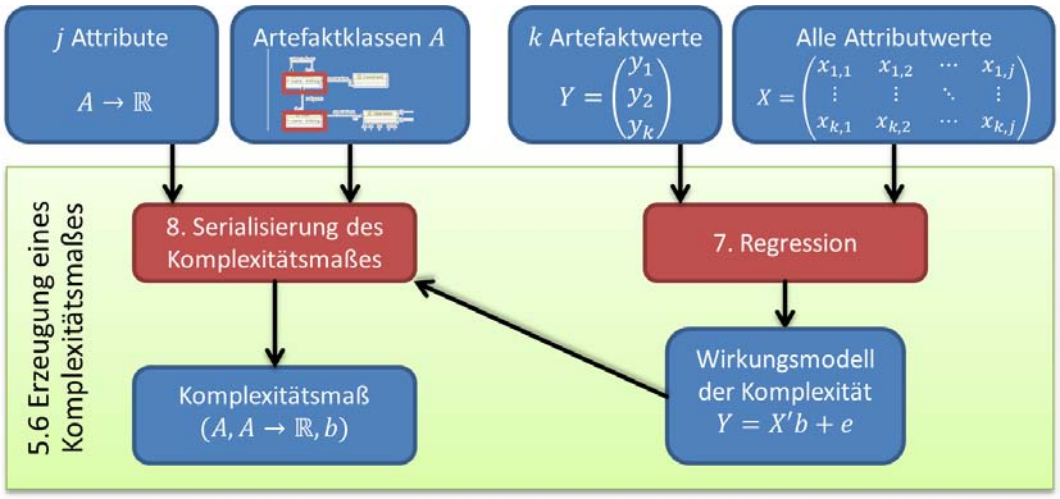


Abbildung 35: Schritte und Modelle der vierten konzeptionellen Phase

Die Analysen in Schritt 7 sollen anhand einer Beispielmatrix $X$ ($r_1$ bis $r_{14}$) demonstriert werden, die auf den Werten aus Tabelle 9 sowie zusätzlichen Attributen basiert. Das Beispiel wird außerdem um eine einspaltige Matrix der Aufwandswerte $Y$ ($y_k$) zu einem vollständigen Regressionsmodell ergänzt. Die Daten der Matrizen sind in der folgenden Tabelle widergegeben.

Tabelle 10: Beispieldaten der Matrizen eines Regressionsmodells

| $k$ | $y_k$ | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ | $r_8$ | $r_9$ | $r_{10}$ | $r_{11}$ | $r_{12}$ | $r_{13}$ | $r_{14}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 2 | 0 | 0 | 0 | 0 | 17 | 0 | 0 | 17 | 1 | 0 | 8,4 | 45 | 0 |
| 2 | 9 | 3 | 1 | 2 | 0 | 1 | 15 | 0 | 0 | 16 | 1 | 0 | 9,2 | 50 | 0 |
| 3 | 25 | 4 | 0 | 0 | 0 | 0 | 22 | 0 | 0 | 22 | 1 | 0 | 34,1 | 47 | 0 |
| 4 | 22 | 3 | 0 | 0 | 3 | 1 | 11 | 0 | 0 | 11 | 1 | 0 | 29,3 | 9 | 0 |
| 5 | 27 | 5 | 0 | 0 | 5 | 1 | 17 | 0 | 0 | 18 | 1 | 4 | 33,7 | 8 | 0 |
| 6 | 6 | 1 | 0 | 0 | 1 | 0 | 4 | 0 | 0 | 4 | 1 | 0 | 5,5 | 53 | 0 |
| 7 | 16 | 3 | 0 | 0 | 3 | 0 | 45 | 0 | 0 | 44 | 1 | 0 | 9,6 | 33 | 0 |
| 8 | 28 | 5 | 0 | 1 | 3 | 0 | 18 | 0 | 1 | 18 | 1 | 0 | 21,4 | 33 | 0 |
| 9 | 17 | 3 | 0 | 0 | 3 | 0 | 19 | 0 | 0 | 22 | 1 | 0 | 18,4 | 17 | 0 |

### 5.6.1 Regression

Die Regressionsanalyse stellt den konzeptuellen Kern der Analyse des Referenzproduktmodells dar, da sie das Modell bestimmt, das die Grundlage des Komplexitätsmaßes bildet. Zunächst wird das Problem der hohen Anzahl an Attributen und somit hohen Anzahl an Variablen für das Regressionsmodell diskutiert. Hierzu werden alternative

Verfahren zur Auswahl der Attribute vorgestellt und als Verfahren des maschinellen Lernens eingeordnet. Danach erfolgen die Regression und die Bewertung des daraus entwickelnden Wirkungsmodells der Komplexität. Im Folgenden wird zunächst das Gebiet des maschinellen Lernens kurz beschrieben.

Maschinelles Lernen ist ein Gebiet der Informatik, das sich mit der Ableitung von allgemeinem Wissen aus einer Menge von Daten durch ein künstliches System beschäftigt. Die Aufgabe des Systems ist es Regeln, Muster oder andere Arten von Zusammenhängen aus den verfügbaren Informationen abzuleiten; das heißt daraus zu lernen. Verfahren des maschinellen Lernens werden im Data Mining auf große Datenbestände in Datenbanken angewendet. Anwendungsbeispiele des Maschinenlernens sind zum Beispiel Warenkorbanalysen, die Kategorisierung von Kunden oder die Schrifterkennung. Die Erstellung eines Komplexitätsmodells kann auch als maschinelles Lernen aufgefasst werden. Die Komplexitätswirkungszusammenhänge werden aus den Daten des Produktmodells und den Aufwänden erlernt.

*"We may not be able to identify the process completely, but we believe we can construct a good and useful approximation. That approximation may not explain everything, but may still be able to account for some part of the data. We believe that though identifying the complete process may not be possible, we can still detect certain patterns or regularities. This is the niche of machine learning. Such patterns may help us understand the process, or we can use those patterns to make predictions."* (Alpaydin 2010, 2)

Bezogen auf den hier vorgestellten Ansatz bedeutet dies, dass man die Entwicklung von Produktmodellen nicht im Ganzen erfassen kann, aber durch das Erkennen von Regelmäßigkeiten ausreichend in einem Komplexitätsmodell approximiert. In diesem Ansatz dient dieses Modell primär der Vorhersage im Rahmen der Produktivitätsbewertung und sekundär zum Verständnis des Prozesses an sich. Die lineare Regression kann als Verfahren des maschinellen Lernens aufgefasst werden. Es handelt sich hierbei um ein sogenanntes überwachtes Verfahren im Sinne des Maschinenlernens. Dem System werden Instanzen und deren bekannte Zielwerte vorgegeben und es leitet hieraus eine allgemeine Abbildung aus der Instanzenmenge in die Zielmenge ab. Im vorgestellten Ansatz werden die Instanzen durch die Artefakte dargestellt und der Zielbereich ist die Komplexität. Dem gegenüber stehen unüberwachte Verfahren, bei denen dem System nur Instanzen übergeben werden. Auf Basis dieser Instanzen wird der Zielbereich abgeleitet, wobei es sich in der Regel um eine Menge von Klassen handelt, und eine Abbildung von Instanzen in den Zielbereich definiert.

#### 5.6.1.1 Anzahl der Attribute und verbundene Probleme

Durch die Generierung der Attribute aus dem Metamodell im 5. Schritt des Verfahrens (Abschnitt 5.5.1) können sehr viele Variable im Sinne der Regression erzeugt werden. Zum Beispiel wurden im Validierungsszenario Permeter (siehe Abschnitt 7.1 ) vor der

Optimierung der Attributgenerierung über 1000 Attribute erzeugt. Da die Betrachtung der Attribute sich hierbei nur auf die Informationen im Metamodell beschränkt hat und keine manuelle inhaltliche Prüfung stattgefunden hat, waren hierunter auch in großer Anzahl Variablen, die keinerlei Bezug zur Komplexität aufwiesen. Dies demonstriert die Probleme für die Skalierbarkeit, die sich aus der automatischen Erzeugung der Attribute ergeben. Bei einer manuellen Untersuchung des Produktmodells würde hingegen zunächst versucht werden, möglichst nicht miteinander korrelierte Variable zu wählen um von vornherein geeignete Daten für die Analyse zu gewinnen. Dies würde auf Basis existierender Erfahrung mit empirischen Untersuchungen und der Kenntnis des Untersuchungsgegenstandes geschehen. Da es sich bei der Komplexitätsanalyse um ein maschinelles Verfahren handelt, ist dies für den vorgestellten Ansatz nicht möglich. Stattdessen werden auf Basis des Metamodells zunächst möglichst viele Variable in Form von Attributen erzeugt und in einem späteren Schritt relevante Variablen ausgewählt.

Für die Bildung des Komplexitätsmaßes müssen daher zunächst folgende Fragen beantwortet werden: Welche Attribute sind für die empirische Komplexität relevant? Wie viele Beobachtungen sind notwendig um verlässliche Modelle zu erstellen? Diese Fragen sind sogar wesentlich schwieriger zu beantworten, als die eigentliche Durchführung der Regression. Sind die relevanten Variablen bekannt, so kann diese durch Statistikprogramme und –Bibliotheken (vgl. Abschnitt 6.4.3) übernommen werden. Durch die Reduktion der Attribute wird in den meisten Fällen erst die Regression ermöglicht, denn es müssen weniger Attribute als Beobachtungen vorliegen. Empfohlen wird, dass die Anzahl der Beobachtungen mindestens doppelt so groß ist wie die Anzahl der Attribute (vgl. (Backhaus u. a. 2008, 106). Wenn es mehrere Artefaktklassen gibt, sollte dieses Verhältnis noch höher liegen, da unter Umständen zwei nicht homogene Untermengen von Beobachtungen vorliegen. Dies ist für die Regression nicht problematisch, erfordert jedoch eine breitere Datenbasis im Sinne von mehr Beobachtungen. Auch aus Gründen der Plausibilitätsprüfung durch den Benutzer sollte die Anzahl der Attribute auf eine überschaubare Menge reduziert werden. Die Herausforderung bei der Reduzierung soll an zwei Verfahren verdeutlicht werden, die zwar intuitiv geeignet erscheinen, aber bei genauerer Betrachtung ausgeschlossen werden müssen.

Das erste Verfahren ist die Auswahl der Attribute auf Grundlage ihres Korrelationskoeffizienten zu dem Beobachtungsvektor. Da davon ausgegangen wird, dass die Attribute einen linearen Einfluss auf die Komplexität besitzen, impliziert dies auch eine höhere Korrelation zwischen Beobachtung und Attribut. Allerdings kann diese Korrelation durch den Einfluss gewichtigerer Faktoren überdeckt sein. Dieses Problem wird durch Abbildung 36 verdeutlicht.

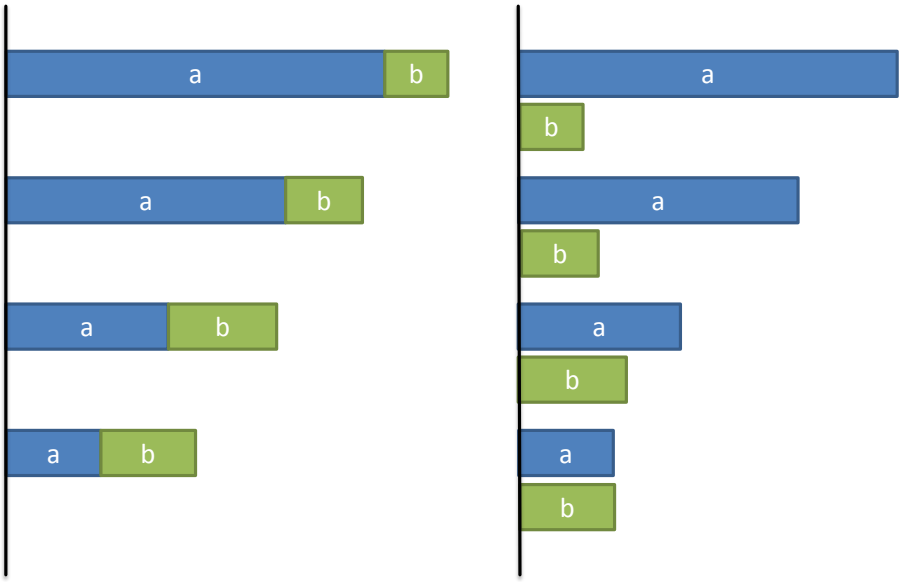


Abbildung 36: Relevante Attribute mit hoher und niedriger Korrelation

In dem dargestellten Beispiel wird die Komplexität durch zwei Attribute a und b vollständig bestimmt. Vergleicht man anhand der zerlegten Teilkomplexitäten die Beobachtungen mit den Attributwerten, so ergibt sich offensichtlich eine hohe Korrelation für das stärker gewichtete Attribut a. Das Attribut b hat jedoch bei den oberen Beobachtungen niedrigerer Werte und bei den unteren Beobachtungen höhere Werte. Eine Messung der Korrelation zwischen Beobachtungen und den Attributwerten von b würde eine sehr niedrige Korrelation ergeben. Allerdings ist gerade das Attribut b wichtig um die Schwankungen der Beobachtungen relativ zum Attribut a zu erklären. Falls a und b zwei von fünfzig Attributen in diesem Schritt wären, so würde durch die Bewertung nach Korrelation Attribut b bereits im Vorfeld verworfen werden.

Eine weitere Möglichkeit besteht darin, eine Regression mit allen möglichen Kombinationen der Attribute durchzuführen um keine Attribute vorzeitig auszuschließen. Hierbei kann die Anzahl der Elemente in den Kombinationen auf eine für die Regression anwendbare Größe festgelegt werden. Dies sei zum Beispiel ein Drittel der $k$ Beobachtungen. Die Anzahl der zu untersuchenden $k$-Teilmengen einer $j$-Attributmenge ergibt sich dann durch die folgende Formel[10]:

$$\text{Anzahl Kombinationen} = \frac{j!}{\left(j - \left\lfloor\frac{k}{3}\right\rfloor\right)! * \left\lfloor\frac{k}{3}\right\rfloor!}$$

[10] Die Herleitung der Formel kann aus Grundlagenbücher der Kombinatorik entnommen werden. Es handelt sich um eine sehr geläufige Formel, da das kombinatorische Problem dem der Gewinnchancen in der Lotterie (Ziehung 6 aus 39) entspricht.

Dieser Wert steigt mit der Anzahl der Attribute stark an. Bei 50 Attributen und 20 Beobachtungen ergibt sich zum Beispiel folgender Wert:

$$\text{Anzahl Kombinationen } = \frac{50!}{\left(50 - \left\lfloor\frac{20}{3}\right\rfloor\right)! * \left\lfloor\frac{20}{3}\right\rfloor!} = \frac{50!}{42! * 6!} = 30.065.204.400$$

Die Auswertung aller Kombinationen ist also nicht immer realistisch möglich, da dieses Verfahren für eine hohe Anzahl an Attributen nicht skaliert. Selbst bei kleineren Metamodellen als im Permeter-Szenario werden zu viele Attribute erzeugt um eine Regression mit jeder in Frage kommenden Kombination durchzuführen.

#### 5.6.1.2 Vorgehen zur Reduktion der Attribute

Im Bereich des Maschinenlernens ist das Problem der Auswahl der Attribute als Dimensionsreduktion bekannt. In vielen Anwendungsdomänen, wie zum Beispiel der Genomanalyse, finden sich Datensätze mit zahlreichen Variablen[11], deren Bezug zum Gegenstand der Untersuchung nicht bekannt ist. Variablen können irrelevant für die Untersuchung sein oder sie sind zu anderen Variablen redundant. Die Ansätze zur Dimensionsreduzierung können in zwei Kategorien aufgeteilt werden (M. A. Hall und Smith 1997): Die Variablenauswahl wählt aus der Menge aller verfügbaren Variablen eine Untermenge aus, die den größten Nutzen für die Analyse hat. Die Variablenverdichtung erzeugt auf Basis der vorhanden Variablen eine neue Menge von Variablen, indem sie diese kombiniert. In beiden Fällen soll die Anzahl der Variablen soweit reduziert werden, dass die Verfahren des maschinellen Lernens in vertretbarer Rechenzeit angewendet werden können. Für die Regressionsanalyse im Rahmen dieser Arbeit wird ein mehrstufiger Ansatz gewählt, der vorwiegend eine Variablenauswahl darstellt. Dies lässt sich wie folgt begründen:

- Der Anteil der irrelevanten Attribute überwiegt stark in der Menge aller theoretisch möglichen Attribute, da keine inhaltliche Prüfung durch einen Menschen stattfindet.
- Das Ergebnis der Komplexitätsanalyse kann bei nicht verdichteten Variablen besser durch den Benutzer interpretiert und der Komplexitätsbeitrag einzelner Attribute wird deutlich.
- Die Serialisierung des Komplexitätsmaßes ist wesentlich kompakter, da keine Verdichtungsfunktionen abgebildet werden müssen und nur wenige Attribute beschrieben werden müssen.

Der Ansatz stellt sicher, dass sich der benötigte Rechenaufwand für die Auswertung des Produktmodells und der Attributwerte erheblich verringert ohne die Qualität der Ergeb-

[11] Im Kontext der Dimensionsreduktion werden Variablen in der englischsprachigen Literatur als Features bezeichnet. Die beiden vorgestellten Hauptansätze werden als Feature (Subset) Selection und Feature Extraction bezeichnet.

nisse wesentlich zu schmälern. Die Stufen dieses Vorgehens sind in Abbildung 37 dargestellt und werden im Folgenden beschrieben.

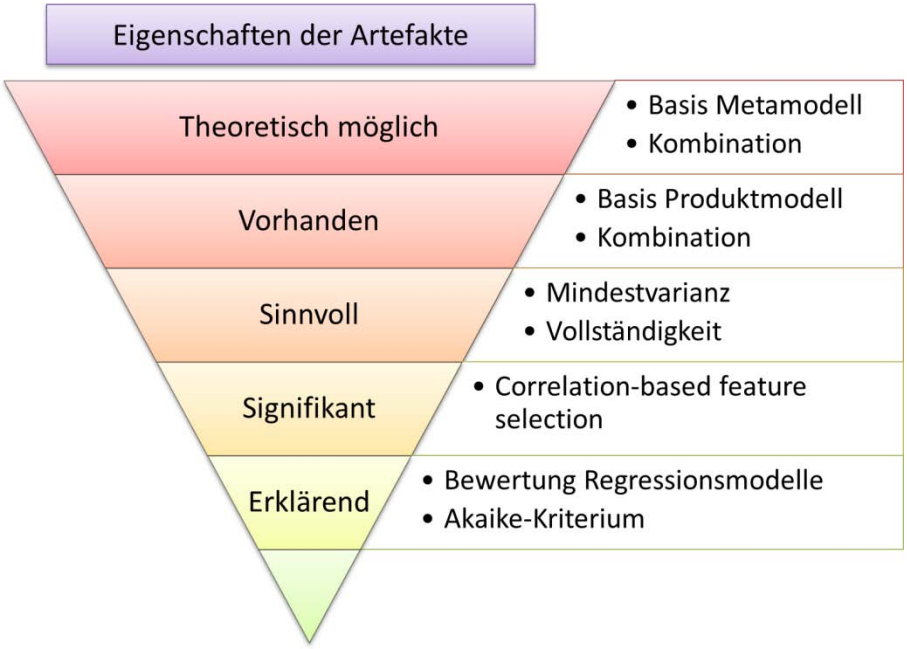


Abbildung 37: Stufen der Attributreduktion

#### 5.6.1.3 Auswahl theoretisch möglicher Attribute

Diese Stufe entspricht der maximal möglichen Ausgangsmenge an Attributen, die auf Basis des Metamodells erzeugt werden können. Die Generierung der Attribute wird in Abschnitt 5.5.1 beschrieben und im Anhang werden die Attributmuster der Validierung zusammengefasst. Die Anzahl der Attribute hängt vom Detailgrad des Metamodells und der Ausdrucksmächtigkeit der Metamodellsprache ab.

#### 5.6.1.4 Auswahl vorhandener Attribute

Diese Stufe umfasst maximal alle im Referenzproduktmodell vorhandenen, messbaren Eigenschaften der Artefakte. Für die Generierung der Attribute wird für jedes Artefakt geprüft, über welche messbaren Eigenschaften es verfügt. Dies geschieht wie in der vorherigen Stufe über Attributmuster, deren Parameter jedoch vom Artefakt anstatt vom Metamodell ermittelt werden. Für das Attributmuster Reference Cardinality zum Beispiel geprüft, über welche ausgehenden Relationen ein Artefakt verfügt. Aus dem in Abbildung 38 dargestellten Teil eines Referenzproduktmodells können zum Beispiel drei Attribute mit den Parameter a, b und c erzeugt werden.

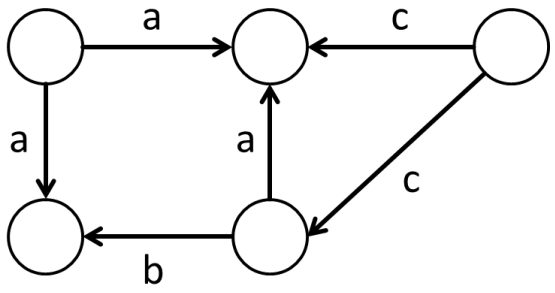


Abbildung 38: Beispielausschnitt des Referenzproduktmodells

Die in diesem Schritt erzeugten Attribute können erheblich reduziert werden, indem geprüft wird, ob eine Eigenschaft mehr als einmal auftritt. Der Einfluss einer einzigartigen Eigenschaft eines Artefakts kann in der Regression nicht bestimmt werden. Daher sollte eine Eigenschaft eine Mindestanzahl an Vorkommen im Referenzproduktmodell besitzen. In dem Beispiel in Abbildung 38 wird bei einer Mindestanzahl von zwei Vorkommen nur das Attribut mit dem Parameter a ausgewählt. Der Parameter c wird nicht ausgewählt, da es sich bei den zwei Kanten dieses Typs nur eine Beobachtung (vom Wert 2) handelt. Da die Komplexitätswerte der Artefakte hierbei nicht relevant sind, handelt es sich bei dieser Selektion um unüberwachtes Lernen.

#### 5.6.1.5 Auswahl sinnvoller Attribute

Das Ziel dieser Stufe ist es festzustellen, welche der automatisch generierten Attribute verwertbare Daten gemessen haben und eine sinnvolle Beobachtung des Referenzproduktmodells darstellen. Hierzu werden relativ einfache statistische Tests durchgeführt, die sowohl ohne die Komplexitätswerte (unüberwachtes Lernen) als auch mit Hilfe der Werte (überwachtes Lernen) durchgeführt werden. Diese Stufe kann als grober Filter für die Variablen beschrieben werden, der offensichtliche Verstöße gegen die Bedingungen an Variablen in einer linearen Regression prüft:

- **Die unabhängigen Variablen beziehungsweise die Attribute beschreiben eine metrische oder boolesche Eigenschaft des Untersuchungsgegenstandes.** Durch die Prüfung der Mindeststreuung werden die Variablen ausgeschlossen, die offensichtlich diese Voraussetzung nicht erfüllen und nicht zur Regressionsanalyse geeignet sind.
- **Die Variablen sind unabhängig voneinander.** Wenn zwei Attribute sehr ähnliche Werte ermitteln, so kann es sein, dass sie im Grunde dieselbe Eigenschaft oder sich inhaltlich überschneidende Eigenschaften messen. Durch die Korrelationsanalyse werden Überschneidungen der Attribute bewertet um das Modell zu vereinfachen.

Durch die Aussortierung der ungültigen Attribute wird die Anzahl der Attribute erheblich reduziert und vereinfacht so die Attributselektion und verbessert die Interpretationsmöglichkeit und Plausibilitätsprüfung durch den Benutzer.

#### 5.6.1.5.1 Vollständigkeit der Daten

Die Erfassung der Attributwerte kann so implementiert werden, dass zwischen einem Messwerte von 0 und einem nicht messbaren Wert unterschieden wird. Hierbei bezieht sich die Nichtmessbarkeit nicht auf Fehler bei der Messung oder auf fehlende Daten sondern auf die Interpretation des Attributs.

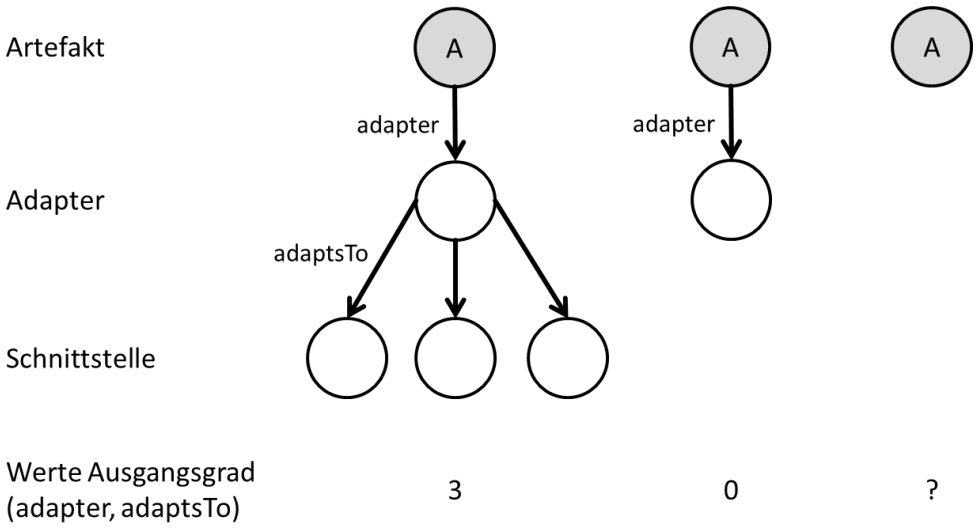


Abbildung 39: Interpretation eines fehlenden Attributs

Die Abbildung 39 verdeutlicht dieses Problem an einem Attribut, dass die Anzahl der berücksichtigten Schnittstellen an einer Unterkomponente eines Artefakts misst. Sofern das Artefakt über eine Adapterkomponente verfügt, so lässt sich der Wert dieses Attributs eindeutig erfassen. Verfügt das Artefakt jedoch über keine Adapterkomponente, so kann dieser Wert als 0 oder als nicht messbar (dargestellt als „?“) interpretiert werden. Wenn diese Interpretation bei der Analyse des Referenzmodells zugelassen wird, müssen die verwendeten Werkzeuge und Datenmodelle mit fehlenden Werten umgehen können.

Bei der Auswahl sinnvoller Attribute kann der Anteil nicht gemessener Werte eines Attributs berücksichtigt werden. Es empfiehlt sich bei vielen fehlenden Werten das Attribut zu verwerfen. Die Mindestanzahl für vorhandene Werte in Abhängigkeit von der Anzahl an Artefakten $k$ kann wie folgt festgelegt werden:

$$MinValues = max\left(9, \sqrt{k}\right)$$

Der Wert der Funktion $max$ ist hierbei das Maximum beider Werte. Die absolute Mindestanzahl 9 ist ein Wert, der sich bei den Analysen in der Validierung als sinnvoll er-

wiesen hat um unbrauchbare Attribute zu entfernen; Bei weniger als 9 vorhandenen Werten konnte kein Zusammenhang zur Komplexität belegt werden. Der relative Wert $\sqrt{k}$ stellt sicher, dass das Attribut nur berücksichtigt wird, wenn ein relevanter Anteil der Artefakte über dieses Attribut verfügt.

#### 5.6.1.5.2 Mindeststreuung

In dieser einfachen Prüfung werden die Vektoren der Messwerte der Attribute einzeln betrachtet und auf ihre Aussagekraft untersucht. In der Regel werden hier bereits viele Attribute verworfen, da die generierten Attribute in vielen Fällen Eigenschaften abbilden, die in der Menge der Artefakte homogen sind, sehr selten vorhanden sind oder gar nicht vorkommen. Diese Beobachtungen sind für die Regression nicht sinnvoll auswertbar. Es werden daher alle metrischen Attribute entfernt, deren Beobachtungswerte zu mehr als 70% identisch sind. Dies umfasst entsprechend alle Attribute, die einen einheitlichen Vektor zurückliefern ($r_7, r_{10}$ und $r_{14}$) und deren empirische Varianz somit 0 ist, sowie alle Attribute, die bis auf wenige Abweichungen einem solchem Attribut entsprechen ($r_2, r_3, r_8$ und $r_{11}$). Aus der Beispielmatrix für diese Phase wird insgesamt die Hälfte der Attribute aus dem Modell gestrichen.

Tabelle 11: Anteil identischer Werte der Vektoren der Attribute in %

| $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ | $r_8$ | $r_9$ | $r_{10}$ | $r_{11}$ | $r_{12}$ | $r_{13}$ | $r_{14}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0 | 0 | 0 | 0 | 17 | 0 | 0 | 17 | 1 | 0 | 8,4 | 45 | 0 |
| 3 | 1 | 2 | 0 | 1 | 15 | 0 | 0 | 16 | 1 | 0 | 9,2 | 50 | 0 |
| 4 | 0 | 0 | 0 | 0 | 22 | 0 | 0 | 22 | 1 | 0 | 34,1 | 47 | 0 |
| 3 | 0 | 0 | 3 | 1 | 11 | 0 | 0 | 11 | 1 | 0 | 29,3 | 9 | 0 |
| 5 | 0 | 0 | 5 | 1 | 17 | 0 | 0 | 18 | 1 | 4 | 33,7 | 8 | 0 |
| 1 | 0 | 0 | 1 | 0 | 4 | 0 | 0 | 4 | 1 | 0 | 5,5 | 53 | 0 |
| 3 | 0 | 0 | 3 | 0 | 45 | 0 | 0 | 44 | 1 | 0 | 9,6 | 33 | 0 |
| 5 | 0 | 1 | 3 | 0 | 18 | 0 | 1 | 18 | 1 | 0 | 21,4 | 33 | 0 |
| 3 | 0 | 0 | 3 | 0 | 19 | 0 | 0 | 22 | 1 | 0 | 18,4 | 17 | 0 |
| 44 OK | 89 | 78 | 44 OK | 67 OK | 22 OK | 100 | 89 | 22 OK | 100 | 89 | 0 OK | 22 OK | 100 |

#### 5.6.1.5.3 Korrelationsanalyse

In diesem Test werden starke Überschneidungen zwischen den Attributen identifiziert. Als Werkzeug hierfür hat sich die Korrelationsmatrix etabliert. Sie bewertet den linearen Zusammenhang jedes Attributpaars anhand ihres (empirischen) Korrelationskoeffi-

zienten. Dies ist ein Wert, der anhand der Vektoren berechnet werden kann und zwischen $-1$ und $+1$ liegt. Werden die Wertpaare zweier korrelierter Vektoren als Koordinaten in einem zweidimensionalen Koordinatensystem interpretiert, so liegen die Wertpaare auf einer steigenden Linie $(+1)$ bzw. einer fallenden Linie $(-1)$. Der Korrelationskoeffizient kann mit Hilfe von Statistikprogrammen und Bibliotheken aus zwei gleichlangen Vektoren berechnet werden. Der Korrelationskoeffizient für zwei Attribute $r_1$ *und* $r_2$ ist definiert wie folgt:

$$Kor(r_1, r_2) = \frac{\sum_{k=1}^{K}(r_1(a_k) - \overline{r}_1) * (r_2(a_k) - \overline{r}_2)}{\sqrt{\sum_{k=1}^{K}(r_1(a_k) - \overline{r}_1)^2 * \sum_{k=1}^{K}(r_2(a_k) - \overline{r}_2)^2}}$$

Für $r_1$ wie folgt und $r_2$ entsprechend:

$r_1(a_k) =$ Attributwert des Attributs $r_1$ für Artefakt $a_k$ (siehe S. 81)

$\overline{r}_1 =$ Mittelwert der Attributwerte des Attributs $r_1$ über alle Artefakte $k$

Diese Berechnung wird für jedes Paar der Attribute durchgeführt und ist symmetrisch, so dass eine Dreieckmatrix der Korrelationswerte entsteht. Für die aus dem ersten Schritt übriggebliebenen Attribute ergibt sich die folgende Matrix.

Tabelle 12: Matrix der Korrelationskoeffizienten der Attribute

| | $r_1$ | $r_4$ | $r_5$ | $r_6$ | $r_9$ | $r_{12}$ | $r_{13}$ |
|---|---|---|---|---|---|---|---|
| $r_1$ | 1,000 | | | | | | |
| $r_4$ | 0,533 | 1,000 | | | | | |
| $r_5$ | 0,256 | 0,277 | 1,000 | | | | |
| $r_6$ | 0,255 | 0,199 | -0,291 | 1,000 | | | |
| $r_9$ | 0,272 | 0,223 | -0,283 | 0,995 | 1,000 | | |
| $r_{12}$ | 0,736 | 0,432 | 0,344 | -0,050 | -0,038 | 1,000 | |
| $r_{13}$ | -0,477 | -0,869 | -0,444 | -0,069 | -0,107 | -0,597 | 1,000 |

Die Attribute $r_6$ und $r_9$ besitzen eine sehr hohe Korrelation und sehr wahrscheinlich sind diese Attribute stark miteinander verwandt. Die Attribute $r_4$ und $r_{13}$ besitzen hingegen eine relativ hohe negative Korrelation. In dieser Stärke kann dies inhaltlich begründet oder auch Zufall sein. Daher werden diese beiden nicht in einem Attribut zusammengefasst.

Die Reduktion der Attribute $r_6$ und $r_9$ zu einer Variable kann auf unterschiedliche Art geschehen. Eine Möglichkeit ist die Aggregation der beiden Attribute zu einer Variab-

len. Dies setzt aber voraus, dass die beiden Attribute sinnvoll standardisiert werden, da stark korrelierte Vektoren nicht ähnliche Durchschnittswerte haben müssen. Ist der Durchschnittswert eines Vektors wesentlich höher als der andere, so würde bei der Addition ein Vektor vom anderen „überdeckt“ werden. Bei der statistischen Standardisierung eines Vektors wird für jeden Wert die Differenz zum Durchschnitt gebildet und dieser durch die Standardabweichung dividiert. Die Vektoren werden dadurch so skaliert, dass ihr Mittelwert gleich Null und die Standardabweichung Eins ist. Standardisierte Vektoren lassen sich statistisch sinnvoll addieren. Für das Komplexitätsmaß bedeutet dies, dass bei einer Messung die beiden ursprünglichen Attribute erfasst und dann mit Hilfe der Standardisierung erst zur relevanten Beobachtung aggregiert werden müssen. Es handelt sich somit um eine Variablenverdichtung. Dies kann die Messung geringfügig verbessern, ist aber recht aufwändig und das Komplexitätsmaß ist für den Benutzer schwieriger zu verstehen.

Eine andere Möglichkeit besteht darin eine der beiden Attribute als Stellvertreter zu behalten und das andere Attribut einfach zu entfernen. Dies kann bedeuten, dass das Komplexitätsmaß geringfügig ungenauer wird. Allerdings sind bei einem Korrelationskoeffizienten der Attribute $r_6$ und $r_9$ von 0,99 keine relevanten Einbußen zu erwarten. Ist die Korrelation geringer aber noch signifikant (~ 0,8 bis 0,95) so kann immer noch ein Attribut als Stellvertreter auswählen werden. Allerdings stellt sich dann die Frage, welches Attribut besser geeignet ist. Hierfür kann man zwei Kriterien heranziehen: Die Beschreibung des Attributs oder dessen Korrelation zu den Aufwandswerten.

Bei der Beschreibung des Attributs kann untersucht werden, wie viele Parameter aus dem Metamodell bei der Erzeugung benötigt wurden beziehungsweise wie speziell die gemessene Eigenschaft ist. Dies kann als statische Methode der Attributmuster auf einer fest definierten Skala implementiert werden. Des Weiteren kann geprüft werden, ob sich eine hohe Korrelation durch die Eigenschaften der Attributmuster erklären lässt. Zum Beispiel besitzen die Attribute (Reference-Cardinality, -) und (Reference-Cardinality, `constraints`) eine Korrelation, da letzterer eine Verfeinerung darstellt. Wie stark diese Relation ist, hängt davon ab, wie viele unterschiedliche Typen von Relationen ein Artefakt durchschnittlich besitzt. Korrelationen entstehen auch, wenn eine Relation die Inverse einer anderen Relation ist. Diese Information kann mit Hilfe einiger Metamodelle wie Ontologien (siehe Abschnitt 6.1) beschrieben und bereits bei der Attributerzeugung berücksichtigt werden. Bei anderen Metamodellen wie Entity-Relationship-Diagrammen können solche Informationen hingegen nicht formuliert werden. Im Folgenden wird davon ausgegangen, dass das Attribut $r_6$ sich als einfacher und somit geeigneter erweist und $r_9$ verworfen wird.

#### 5.6.1.6 Auswahl signifikanter Attribute

Die vorherigen Schritte zur Selektion von Attributen stellen vor allem grobe Filter dar, die durch einfache Methoden implementiert werden. Durch sie wird die Anzahl der Attribute für die folgenden Schritte auf ein handhabbares Maß reduziert. (In den Analysen der Validierung waren dies in den meisten Fällen zwischen 15 und 80 Attribute.) Es wurden sinnvolle und unterscheidbare Attribute identifiziert aber noch nicht deren Aussagekraft für die Komplexität untersucht. Entsprechend handelte es sich vorwiegend um unüberwachte Lernverfahren im Sinne des Maschinenlernens. In diesem und dem folgenden Schritt wird der Zusammenhang der verbliebenden Attribute untersucht und es kommen folglich überwachte Lernverfahren zum Einsatz.

Die Auswahl signifikanter Attribute untersucht, welche Untermenge der Attribute einen erkennbaren Einfluss auf die Komplexität besitzen und somit in die Regression aufgenommen werden sollte. Hierzu werden zwei Verfahren als Kandidaten ausgewählt und anhand des korrigierten Bestimmtheitsmaßes der Regression auf Basis der ausgewählten Attribute miteinander verglichen. Es handelt sich um die Hauptkomponentenanalyse und die korrelationsbasierte Variablenselektion. Beide Verfahren werden durch Statistik-APIs zur Verfügung gestellt und werden an dieser Stelle nur relativ kurz in ihren Eigenschaften beschrieben.

Die Hauptkomponentenanalyse (Principal Component Analysis) (PCA) wurde von Karl Pearson (Pearson 1901) entwickelt und wird als Werkzeug in der explorativen Datenanalyse genutzt. Ihr Hauptanwendungsgebiet ist die Reduktion der Variablen in umfangreichen Daten wie den Attributwerten der Artefakte in dieser Analyse. Grundidee ist die Abbildung von Variablen in eine Menge von Linearkombinationen (Hauptkomponenten) der Variablen. Die Anzahl der Linearkombinationen ist hierbei niedriger als die der ursprünglichen Variablen. Es handelt sich somit um eine Variablenverdichtung. Eine Linearkomponente soll eine Menge inhaltlich verwandter Variablen zusammenfassen. Bezogen auf ein Artefakt kann zum Beispiel eine Linearkombination der Attribute Reference Cardinality und Usage als „Verknüpfungsgrad“ gebildet werden. Die Koeffizienten dieser Kombination werden durch das Verfahren so gewählt, dass das neue Attribut möglichst aussagekräftig im Sinne eines Vorhersagemodells ist.

Die korrelationsbasierte Variablenselektion (Correlation-based Feature Selection) (CFS) ist ein in der Arbeit von Mark Hall (M. A. Hall 1999) entwickeltes Verfahren zur Variablenauswahl. Der Einsatzbereich von Halls Algorithmus ist das Aufbereiten von großen Datensätzen, so dass diese für andere Verfahren des maschinellen Lernens angewendet werden können. Insbesondere die Auswahl von Attributen für Klassifizierungsverfahren wird als wichtiges Einsatzgebiet gesehen und die Validierung der Arbeit orientiert entsprechend an der Verbesserung der Qualität der Ergebnisse eines darauffol-

genden Verfahrens. Die grundlegende Idee ist in der zentralen These der Arbeit zusammengefasst.

*"A good feature subset is one that contains features highly correlated with (predictive of) the class[12], yet uncorrelated with (not predictive of) each other."* (M. A. Hall 1999, 4)

Dieses Kriterium wird in dem Verfahren durch eine Bewertungsfunktion ausgedrückt (siehe (M. A. Hall 1999, 69)), die die Güte einer Auswahl von Variablen bestimmt. Diese Gütefunktion wird mit einem heuristischen Optimierungsverfahren wie dem Greedy-Algorithmus kombiniert um einen Lösungsraum abzusuchen. Da sowohl die Bewertungsfunktion als auch das Optimierungsverfahren sehr schnell sind, können auch sehr große Variablenmengen untersucht werden.

Für die Bewertung wurden die Daten des Validierungsszenarios Permeter verwendet. Die Anzahl der Beobachtungen betrug 44. Die bereits beschriebenen Schritte zur Auswahl sinnvoller Attribute wurden bereits durchgeführt. Die Ergebnisse der Untersuchung sind in Tabelle 13 dargestellt. Die CFS erzielt leicht bessere Werte als die PCA. Beide Verfahren können als geeignet angesehen werden, um Variablen im Komplexitätsmodell auszuwählen. Da die CFS leicht bessere Werte liefert und für den Benutzer verständlichere Zusammenhänge aufzeigt, wird dieses Verfahren für diese Stufe der Variablenselektion ausgewählt.

Tabelle 13: Bewertung von Verfahren zur Variablenreduktion

| **Variablenauswahl** | **Principal Component Analysis** | **Correlation-based Feature Selection** |
|---|---|---|
| **Anzahl Variablen** | 11 | 9 |
| **Bestimmtheitsmaß $R^2$** | 0,99357 | 0,99944 |
| **Korrigiertes Bestimmtheitsmaß $\overline{R^2}$** | 0,99137 | 0,99435 |

Zu den Ergebnissen ist anzumerken, dass die Werte von $R^2$ und $\overline{R^2}$ sehr hoch und nicht bezeichnend für die zu erwartende Qualität des Vorhersagemodells sind. Die Anzahl der Variablen, die einen wesentlichen Beitrag zum Vorhersagemodell leisten, wird tendenziell geringer sein. Das korrigierte Bestimmtheitsmaß „bestraft" die Anzahl der Variablen nicht ausreichend (Fahrmeir, Kneib, und Lang 2008, 161). An dieser Stelle wird

[12] Der Zielwert des maschinellen Verfahrens wie Klassen bei einer Klassifikation. Bei der linearen Regression der Vektor entspricht dies der abhängigen Variablen

dennoch dieses Maß zur Auswahl eines Verfahrens zur Variablenselektion ausgewählt, da ein Überhang an Variablen für die letzte Stufe der Variablenselektion sinnvoll ist.

#### 5.6.1.7 Auswahl erklärender Attribute

Durch die vorherigen Schritte wurden Attribute ermittelt, die anscheinend in ihrer Gesamtheit als Grundlage für ein Komplexitätsmodell genutzt werden können. Halls CFS hat eine geeignete Untermenge von Attributen identifiziert, auf der in diesem letzten Schritt Regressionen durchgeführt werden. Insbesondere wird auch der Beitrag der einzelnen Variablen in einem Komplexitätsmodell betrachtet. Es wird, wie im vorherigen Schritt, ein heuristisches Suchverfahren durchgeführt um eine Auswahl an Attributen zu treffen. Die Bewertung basiert hierbei jedoch auf der Ausführung einer Regression. Das Vorgehen lässt sich wie folgt skizzieren:

- Wähle eine Untermenge der Attribute als Anfangslösung aus.
- Führe eine Regression auf der aktuellen Lösungsmenge aus.
- Wende eine Bewertungsfunktion auf das Ergebnis der Regression an
- Entscheide mit Hilfe des Suchverfahrens auf Basis der Bewertung, ob und wie eine andere Untermenge der Attribute als neue Lösungsmenge ausgewählt wird. Wird eine neue Untermenge ausgewählt, so gehe zu 2.
- Erstelle ein Komplexitätsmodell aus der Lösungsmenge

Als Anfangslösung wird die Menge aller Attribute ausgewählt. Die Regression kann durch eine Statistik-API durchgeführt werden. Wichtig ist vor allem die Auswahl einer geeigneten Bewertungsfunktion, die eine Modellqualität abbildet, die Eigenschaften des Modells wie Genauigkeit, Verlässlichkeit und Größe berücksichtigt. Diese Qualität wird auch als Informationskriterium bezeichnet und wird unter anderem durch folgende Bewertungsfunktionen implementiert. Die Funktionen werden an dieser Stelle nicht im Detail vorgestellt, da die Herleitung und die Erläuterung der statistischen Grundlagen umfangreich sind. Außerdem sind Implementierungen dieser Kennzahlen bereits in vielen Statistik-APIs verfügbar. Daher sollen nur kurz die wesentlichen praktischen Eigenschaften erläutert werden:

- Korrigiertes Bestimmtheitsmaß: Diese Maß wurde bereits im vorherigen Schritt als Bewertungsfunktion genutzt, um ein Verfahren zur Variablenselektion auszuwählen. Es ist eher als historischer Vorläufer moderner Informationskriterien zu sehen. Für die Auswahl der signifikanten Attribute ist diese Bewertungsfunktion jedoch nicht geeignet, da es tendenziell zu viele Attribute akzeptiert.
- Bayesianisches Informationskriterium (BIC) (Schwarz 1978): Dieses Maß wird auf Basis des Bayesschen Wahrscheinlichkeitsbegriffs hergeleitet und besitzt einen wesentlich größeren „Strafterm“ für die Anzahl der Attribute als das korrigierte Bestimmtheitsmaß.

- Akaikes Informationskriterium (AIC) (Akaike 1974): Dieses Maß ist mit dem BIC eng verwandt. Es beruht auf der Methode der maximalen Wahrscheinlichkeit (Maximum-Likelihood). In praktischer Hinsicht ist die Bestrafung der Attributanzahl jedoch nicht so ausgeprägt wie beim BIC (Fahrmeir, Kneib, und Lang 2008, 162).

Als Suchverfahren auf Basis dieser Bewertungsfunktionen kann zum Beispiel wieder ein Greedy-Algorithmus benutzt werden oder auch andere Suchalgorithmen wie genetische Algorithmen. Da nur noch einige Attribute vorhanden sind, kann eine relativ breite Suche durch Anpassung der Algorithmen durchgeführt werden. Bei ausreichender Rechenleistung und genügend Zeit ist sogar eine vollständige Durchsuchung des Lösungsraumes denkbar. Die Auswahl der erklärenden Attribute beendet somit die Modellsuche und resultiert in einem Komplexitätsmodell.

#### 5.6.1.8 Bewertung des Komplexitätsmodells

Durch die Auswahl der erklärenden Attribute ist die wesentliche Herausforderung bei der automatischen Analyse des Referenzproduktmodells genommen. Die Regression wurde bei der Attributsuche „nebenbei“ vorgenommen und muss nicht erneut durchgeführt werden. Zu klären ist jedoch, ob das Ergebnis der Analyse ein zuverlässiges Komplexitätsmodell ist. Es kann nicht ausgeschlossen werden, dass auch eine detaillierte Abbildung des Referenzproduktmodells und eine umfassende Analyse zu keinem Komplexitätsmodell geführt haben, das für die Bewertung von anderen Produktmodellen geeignet ist. Diese Möglichkeit erfordert eine „Qualitätskontrolle“ für das Modell.

Um sicher zu stellen, dass das gewonnene Maß auch Komplexität misst, muss überprüft werden, ob zwischen den gefundenen Attributen und der Komplexität tatsächlich ein Zusammenhang besteht. Die Schwankungen der Komplexitätswerte könnten auch durch Einflüsse begründet sein, die nicht im Modell sind. Hierfür bietet sich der F-Test an, der auf der Prüfung einer sogenannten Nullhypothese beruht.

$$Y = b_0 + Xb + e$$

Die obige Formel wiederholt das in Abschnitt 5.5.4 beschriebene Komplexitätsmodell. Die Nullhypothese lautet, dass das Komplexitätsmodell ohne die Attribute (also dem Term $Xb$) die Varianz der Residuen $e$ auch erklärt. Unter der Annahme, dass die Werte der Residuen einer F-Verteilung folgen, kann die Wahrscheinlichkeit berechnet werden, dass sich die unterschiedlichen Varianzen der Modelle (mit und ohne Attribute) lediglich zufällig ergeben. Ist die Wahrscheinlichkeit sehr hoch (>95%), dass die Attribute doch einen Einfluss auf die Varianz haben, so muss die Nullhypothese verworfen werden und man kann von einem Zusammenhang zwischen Attributen und Komplexitätswert ausgehen.

Wenn der Zusammenhang zwischen der Komplexität und den ermittelten Attributen belegt ist, so stellt sich die Frage nach der Genauigkeit der Komplexitätsmessung. Bei einer Regression wird üblicherweise das Bestimmtheitsmaß $R^2$ verwendet um Aussagen zur Anpassung des Modells an die zugrundeliegenden Daten zu treffen. Das Bestimmtheitsmaß ist für lineare Modelle definiert als der Prozentanteil der Streuung von $Y$, hier den Komplexitätswerten, der durch das Modell erklärt wird. Er berechnet sich wie folgt:

$$R^2 = 1 - \frac{Streuung\ der\ Residuen}{Streuung\ der\ Komplexität} = 1 - \frac{\sum_{k=1}^{K} e_k^2}{\sum_{k=1}^{K} (y_k - \bar{y})^2}$$

Die Variablen sind aus der Formulierung des Komplexitätsmodells (siehe Seite 101) übernommen und um folgende Variable ergänzt.

- $e_k$ und $y_k$: Jeweils der k-te Eintrag des Vektors $e$ der Residualgrößen und des Aufwandsvektors $Y$. Die Werte von $e$ hängen von dem Ergebnis der Regression ab.
- $\bar{y}$: Der Mittelwert der beobachteten Komplexitätswerte.

Dieses Maß kann bereits als guter Indikator für die Qualität des resultierenden Komplexitätsmaßes gewertet werden. Allerdings gilt es im Zusammenhang mit der Komplexitätsmessung zu beachten, dass die Messung individueller Artefakte nur von sekundärem Interesse ist. Hauptziel der Messung ist die Messung eines Gesamtwertes eines Produktes und somit der Summe über eine Menge von Artefakten. Hierbei können die gemessenen Werte als Zufallsergebnisse einer normalverteilten Wahrscheinlichkeitsfunktion betrachtet werden, deren Erwartungswert die wahre Komplexität ist. Dieses Modell ist in Abbildung 40 dargestellt mit einer Skala des relativen Messfehlers. (Zur Interpretation negativer Komplexität siehe Abschnitt 5.7.2.)

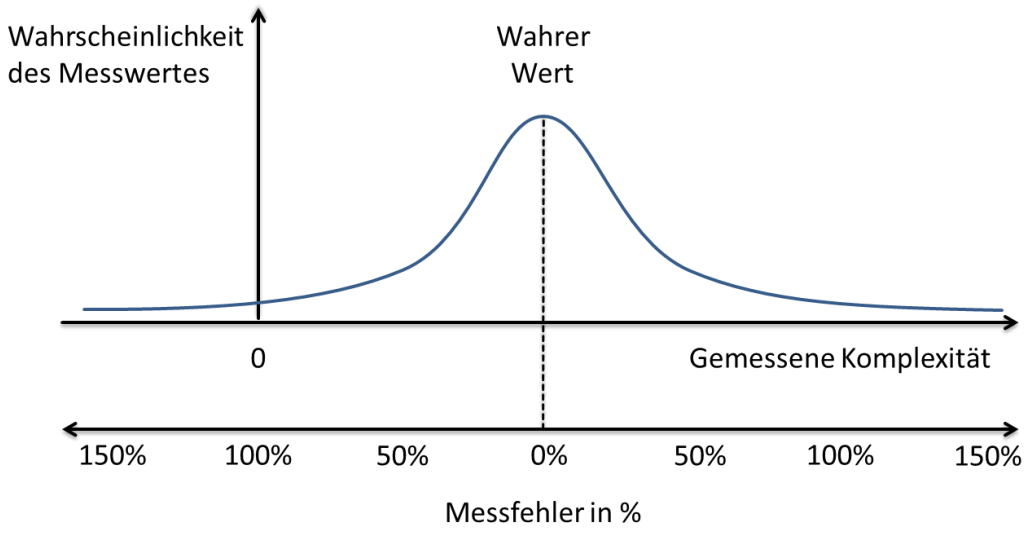


Abbildung 40: Messung der Komplexität als Wahrscheinlichkeitsverteilung

Da der Komplexitätswert als Skalar zum Vergleich von zwei Produktmodellen angewendet wird, interessiert die relative Abweichung. Diese kann bei Einzelmessung relativ hoch sein und wird durch die Addition von zwei oder mehr Messungen in der Regel geringer. Durch die Addition der einzelnen Komplexitäten der Artefakte konvergiert die Messung der Gesamtkomplexität zum Erwartungswert des Komplexitätsmaßes. Für die Anwendung des Komplexitätsmaßes zur Produktivitätsmessung ist somit entscheidend wie viele Artefakte eine Messung umfassen sollte, damit der relative Messfehler mit einer bestimmten Wahrscheinlichkeit gering genug ist. Hierzu wird die Wahrscheinlichkeitsverteilung der addierten Messungen betrachtet. Für die Addition zweier Zufallsgrößen $X_1$ und $X_2$ mit den zwei normalverteilten Dichtefunktionen $f_1(x)$ und $f_2(x)$ gilt allgemein (Kohn 2004, 280):

$$f_1(x) = f_N^{(\mu_1,\sigma_1^2)}(x), f_2(x) = f_N^{(\mu_2,\sigma_2^2)}(x)$$

$$\Longrightarrow f_Y(y) = f_N^{(\mu_1+\mu_2,\sigma_1^2+\sigma_2^2)}(y)$$

Wobei $f_Y(x)$ die Dichtefunktion der Summe darstellt. Sei $f_n(x)$ die Dichtefunktion der $n$-ten Addition einer Zufallsvariablen $X$ mit der normalverteilten Dichtefunktion $f(x)$ so gilt entsprechend:

$$f(x) = f_N^{(\mu,\sigma^2)} \Rightarrow f_n(y) = f_N^{(n\mu,n\sigma^2)}(y)$$

Hieraus lässt sich berechnen, welchen relativen Fehler die addierten Komplexitätsmessungen von Artefakten besitzen, wenn der Erwartungswert $\mu$ und die Standardabweichung $\sigma$ der Messung bekannt ist. Bei Normalverteilungen wird in der Regel mit Tabellen gearbeitet. Aus diesen lässt sich entnehmen, dass im Intervall $\pm 2\sigma$ von $\mu$ ungefähr 95,45% aller Messwerte liegen. Dieser Wert kann übernommen werden um zu be-

schreiben, dass eine Messung mit entsprechender Wahrscheinlichkeit in einem gewünschten Intervall liegen soll. Die relative Abweichung an den Grenzen dieses Intervalls beschreibt die maximale relative Abweichung $x_{rel}$ von 95,45% aller Messungen:

$$x_{rel} = \frac{2\sigma}{\mu}$$

Gesucht ist nun die Anzahl $n$ der Messungen, die durchgeführt werden müssen, um $x_{rel}$ unter einer gewünschten Grenze wie zum Beispiel 0,05 zu bringen. Hierzu werden der Erwartungswert und die Standardabweichung der Summe von $n$ Messungen eingesetzt:

$$x_{rel} = \frac{2\sqrt{n\sigma^2}}{n\mu}$$

$$\Leftrightarrow x_{rel} n\mu = 2\sqrt{n\sigma^2}$$

$$\Rightarrow x_{rel}^2 n^2 \mu^2 = 4n\sigma^2$$

$$\Leftrightarrow n = \frac{4\sigma^2}{x_{rel}^2 \mu^2}$$

Um die notwendigen Parameter $\sigma$ und $\mu$ für ein Komplexitätsmaß zu bestimmen, kann man die zur Regression verwendeten Artefakte als Stichproben und deren Residualgrößen als Messfehler interpretieren. Unter dieser Annahme kann die korrigierte Stichprobenvarianz als Schätzer für die Varianz $\sigma^2$ abgewandelt werden. Die korrigierte Stichprobenvarianz ist für eine Stichprobe $x_1, \ldots, x_n$ definiert als:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Der Term $(x_i - \bar{x})$ als Ausdruck der Abweichung vom Erwartungswert wird hierbei ersetzt durch die Residuen.

$$\sigma^2 \approx s^2 = \frac{1}{n-1} \sum_{i=1}^{n} {e_i}^2$$

Als Erwartungswert wird das arithmetische Mittel der wahren Komplexitätswerte der Artefakte und somit der Werte des Vektors $Y$ verwendet.

$$\mu \approx \frac{1}{k} \sum_{i=1}^{k} y_i$$

Aus diesen angestellten Überlegungen lässt sich eine Bewertung für das Komplexitätsmaß erstellen, welche eine bessere Aussage zur Eignung als Produktivitätskennzahl als $R^2$ ermöglicht, da es die Addition der Einzelkomplexitäten zur Gesamtkomplexität eines

Produktmodells berücksichtigt. Dies sei am Beispiel eines Regressionsmodells mit folgenden Werten demonstriert:

- Der akzeptable relative Fehler wird für den Vergleich von zwei Produktmodellen auf 5% und somit $x_{rel} = 0{,}05$ gesetzt.
- Das Regressionsmodell des Komplexitätsmaßes wird mit $o^2 = 4$ und $\mu = 16$ abgeschätzt.
- Durch Einsetzen wird $n = 25$ berechnet.

Das Komplexitätsmaß ist relativ ungenau bei einzelnen Artefakten und sollte für den Vergleich von Produktmodellen nur auf Produktmodelle mit mindestens 25 Artefakten angewendet werden. Für solche Produktmodelle beträgt mit einer Wahrscheinlichkeit von mindestens 95,45% der Messfehler höchstens 5%. Dies unterliegt der der Komplexitätsmessung zugrundeliegenden Annahme, dass es sich um mit dem Referenzproduktmodell strukturell vergleichbare Produktmodelle handelt. Diese Kennzahl des Regressionsmodells soll im Folgenden als Konvergenzwert des Komplexitätsmaßes bezeichnet werden und stellt ein wichtiges Hilfsmittel zur Einschätzung der Anwendbarkeit dar.

### 5.6.2 Serialisierung des Komplexitätsmaßes

Wenn die Analyse des Referenzproduktmodells beendet ist und ein Regressionsmodell mit ausreichender Qualität erzeugt wurde, muss das Ergebnis dieser Analyse als Komplexitätsmaß gespeichert werden. Zum Komplexitätsmaß gehören alle Informationen, die notwendig sind, um dieses Maß sinnvoll einsetzen zu können. Je nach zugrundeliegendem Datenmodell kann sich die konkrete Ausgestaltung der Daten ändern, es müssen jedoch folgende Informationen abgebildet werden.

- Metainformationen zum Komplexitätsmaß, die das zugrundeliegende Referenzprodukt charakterisieren und somit auch Informationen zur Anwendbarkeit geben.
- Die Einheit des Komplexitätsmaßes, wie sie bei der Erfassung der Aufwände definiert wurde (siehe 5.4.2).
- Die Definition der verwendeten Artefaktklassen entsprechend des verwendeten Datenmodells.
- Die Liste der Attribute $r_1, \dots, r_n$ im Regressionsmodell in Form einer Beschreibung des Attributmusters und der notwendigen Parameter.
- Der Vektor $b$ der Regressionskoeffizienten des Komplexitätsmodells. Dies sind die zentralen Parameter für die Berechnung des Komplexitätswertes.
- Die Kennzahlen $R^2$ und Konvergenzwert des Komplexitätsmaßes.

Das Komplexitätsmodell kann als Datei oder in einer Datenbank abgelegt werden. Beim Laden des Komplexitätsmaßes muss durch die Ausführungsumgebung sichergestellt werden, dass das Maß für das vorliegende Produktmodell geeignet ist und dass die Attribute für die Messungen verfügbar sind.

## 5.7 Durchführung der Messung

Durch die Erstellung eines Komplexitätsmaßes steht ein Werkzeug zur Verfügung um andere Entwicklungsprojekte quantitativ abzuschätzen. Jedoch bestehen auch hier Anforderungen an die Ausgangsdaten und das Vorgehen: Das Referenzprojekt muss einen vergleichbaren Charakter wie das zu vermessende Projekt besitzen. Zum Beispiel sind experimentelle Entwicklungen nicht mit Projekten vergleichbar, die dem Tagesgeschäft zugeordnet werden können. Auch mit einem Komplexitätsmaß lassen sich nicht „Birnen mit Äpfeln“ vergleichen. Sind diese Voraussetzungen gegeben, so lassen sich die ermittelten Werte in weiterführende Analysen einsetzen.

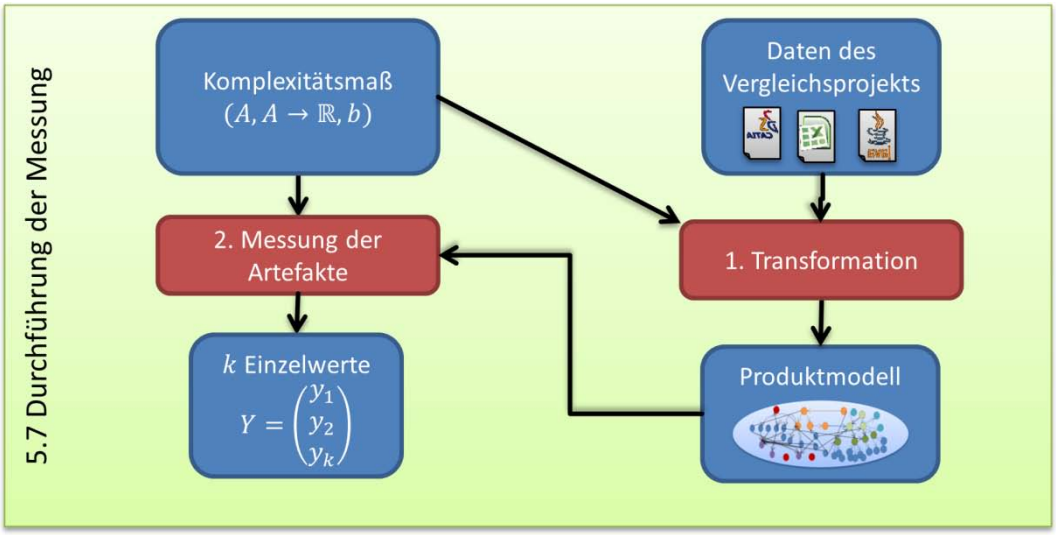


Abbildung 41: Schritte und Modelle der ersten Phase der Anwendung

### 5.7.1 Transformation der nativen Daten

Ausgehend von einem vergleichbaren Projekt muss dieses zunächst wie bei der Erstellung des Komplexitätsmodells in ein graphbasiertes Produktmodell umgewandelt werden. Mögliche Ansätze zur Transformation wurden hierzu in Abschnitt 5.3.3 beschrieben.

Wichtig für die Transformation bei der Messung ist, dass sie auf demselben Mechanismus oder demselben Parser beruht wie das Referenzproduktmodell und mit den gleichen Parametern durchgeführt wird. Nur so kann die Wiederholbarkeit und Plausibilität der Messung sichergestellt werden. Das Komplexitätsmaß kann hierzu Metainformatio-

nen bereitstellen und geht somit als Input neben den Ausgangsdaten des Vergleichsprojektes in diesem Schritt ein. Als Teil der Transformation muss gegebenenfalls wieder eine Integration von Partialmodellen stattfinden.

### 5.7.2 Messung der Artefakte

Für die Messung des transformierten Produktmodells wird zunächst die Menge $A$ der Artefakte identifiziert und anschließend durch die Attributmaße $r_1, \dots, r_n$ vermessen. Die Attributmaße werden durch das Komplexitätsmaß bestimmt. Der Komplexitätswert $c_A(a)$ eines Artefakts $a \in A$ ist dann gegeben durch:

$$c_A: A \to \mathbb{R}, a \mapsto \sum_{i=1}^{n} r_i(a)$$

Es lässt sich nicht ausschließen, dass der Wert $c(a)$ negativ ist. Eine solche negative Komplexität bedeutet, dass das Artefakt eine sehr geringe Komplexität besitzt und das Messergebnis durch natürliche Messschwankungen in den negativen Wertebereich verschoben wurde. Eine negative Komplexität ist nach der Definition von Komplexität (siehe Abschnitt 2.2.2) im Sinne eines negativen Aufwandes nicht sinnvoll begründbar. Sollte der Komplexitätswert wesentlich im negativen Bereich liegen, ist dies ein Indiz für ein unvollständiges Komplexitätsmodell, welches Eigenschaften vernachlässigt, die das mit einer negativen Komplexität bewertete Artefakt beschreiben.

Die Komplexität des Produktmodells als Graph $\mathrm{p}(V, E)$ ergibt sich aus der Summe der Einzelkomplexitäten seiner Artefakte $A \subset V$ .

$$c_P: P \to \mathbb{R}, p \mapsto \sum_{i=1}^{n} c_A(a_i) : a \in A \subset V$$

Dieser Wert beschreibt die Komplexität des Produktmodells in der Einheit, die zur Kalibrierung des Komplexitätsmaßes verwendet wurde. Der Wert entspricht einer Abschätzung, welchen Aufwand das Entwicklungssystem für die Erzeugung des Produktmodells benötigt hätte. In den beiden nächsten Abschnitten werden zwei Ansätze vorgestellt, wie diese Kennzahl zur Bewertung von Entwicklungsprozessen eingesetzt werden kann.

## 5.8 Interpretation der Messwerte

Nachdem die Komplexitätswerte gemessen wurden, müssen diese sinnvoll interpretiert werden. In diesem Abschnitt werden dazu konzeptionelle Grundlagen zur Verwendung eines Komplexitätswertes behandelt. Insbesondere soll gezeigt werden, wie die Integration des Komplexitätsmaßes in bestehende Methoden realisiert werden kann.

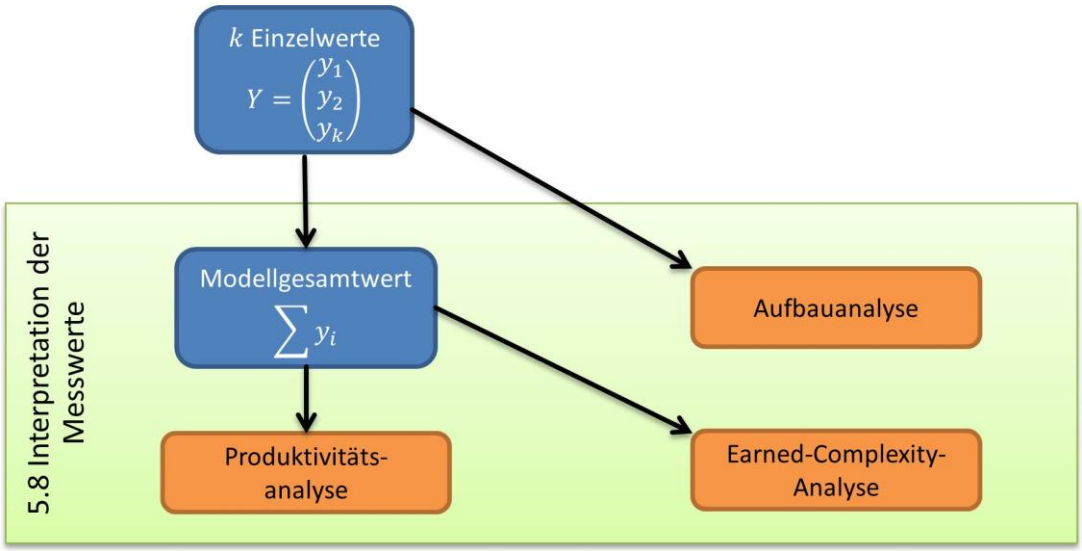


Abbildung 42: Daten und aufbauende Analysen der zweiten Phase der Anwendung

### 5.8.1 Aufbauanalyse

Zwar ist die Messung der Gesamtgröße des Produktmodells das primäre Ziel der Komplexitätsmessung, jedoch lassen sich auch oftmals aus der Verteilung der Einzelkomplexitäten der Artefakte Informationen gewinnen. Der Aufbau eines Produktes lässt sich zum Beispiel in für den Entwicklungsaufwand maßgebende Elemente und Elemente mit geringer Wirkung auf die Gesamtkomplexität aufteilen.

Bei dieser Art von Analyse muss jedoch beachtet werden, dass die Varianz der Messwerte bei Einzelmessungen der Artefakte gegebenenfalls sehr hoch sein kann. In Abschnitt 5.6.1.8 wird dieser Aspekt bezüglich der Qualität eines Komplexitätsmaßes diskutiert. Die beiden im Folgenden beschriebenen Analysen auf Basis eines Modellgesamtwertes sind jedoch wesentlich interessantere Anwendungsmöglichkeiten für das Komplexitätsmaß.

### 5.8.2 Produktivitätsanalyse

In Abschnitt 5.1.3 wurde vor dem Hintergrund der multidisziplinären Entwicklung des Gewerbespülers die Frage entwickelt, wie sich dieser veränderte Designprozess gegenüber der traditionellen Entwicklung bezüglich der Produktivität vergleichen lässt. In der darauf aufbauenden Konzeption wurde mit Bezugnahme auf die in Unterkapitel 2.2 formulierten Anforderungen das Komplexitätsmaß als vergleichendes Maß für Produktmodelle entwickelt. Dadurch kann der Output eines Entwicklungssystems mit Bezugnahme auf das Referenzproduktmodell quantifiziert werden. Wenn der Input aus der Projektkostenrechnung ebenfalls bekannt ist, dann ergibt sich die Produktivität als das Verhältnis vom Output zum Input.

$$Produktivität = \frac{Output}{Input} = \frac{Komplexität\ des\ Produktmodells}{Kosten\ des\ Entwicklungssystems}$$

Wenn außerdem die Einheit des Komplexitätsmaßes, also der verwendeten Aufwandswerte nach Abschnitt 5.4.2, und die Einheit der Kosten übereinstimmen, so kann die Produktivität als Faktor ermittelt werden. Dieser Faktor bezieht sich dann auf die Produktivität des Referenzentwicklungssystems und bietet einen Orientierungswert für die Leistungsfähigkeit des Entwicklungssystems, das das vermessene Produktmodell erzeugt hat. Ob ein Vergleich zwischen dem Referenzentwicklungssystem des Komplexitätsmaßes und dem Entwicklungssystem des vermessenen Produktmodells sinnvoll ist, hängt von einem vergleichbaren Rahmen ab. Handelt es sich bei der Referenzproduktentwicklung um ein länger zurückliegendes Projekt, so müssen bei einem Vergleich Faktoren wie Inflation, veränderte Zusammensetzung von Teams oder technologischer Fortschritt berücksichtigt werden.

Sollen zwei von der Referenzproduktentwicklung verschiedene Entwicklungsprozesse $a$ und $b$ miteinander vergleichen werden, so kann die Einheit des Komplexitätsmaßes aus der Berechnung gestrichen werden. Seien $c_P(a)$ und $c_P(b)$ die gemessene Komplexität der Produktmodelle und $cost(a)$ und $cost(b)$ die erfassten Entwicklungskosten, so ist die relative Produktivität wie folgt definiert:

$$Relative\ Produktivität\ (a,b) = \frac{c_P(a)}{cost(a)} \Big/ \frac{c_P(b)}{cost(b)} = \frac{c_P(a) * cost(b)}{c_P(b) * cost(a)}$$

Neben einer Betrachtung der Produktkomplexität wurde in Abschnitt 5.1.3 ebenfalls die Frage nach dem Einfluss bestimmter Eigenschaften des Produktmodells auf die Komplexität gestellt. Diese Eigenschaften sind im Rahmen des vorgestellten Ansatzes durch die Attribute der Artefakte dargestellt. Durch den Vektor $b$ der Regressionskoeffizienten im Komplexitätsmaß wird der Einfluss der Attribute auf die Gesamtkomplexität beschrieben. Die Werte des Vektors können jedoch nicht direkt miteinander verglichen werden, sondern müssen relativ zur durchschnittlichen Ausprägung des Attributs im Produktmodell gewertet werden. Eine entsprechende Untersuchung der Komplexitätstreiber eines Produktmodells kann Einblicke in die Faktoren geben, die den Aufwand bei der Entwicklung wesentlich beeinflussen. Im Kontext des Gewerbespülers kann zum Beispiel untersucht werden, welche Relationen zwischen den Teilmodellen der multidisziplinären Entwicklung eine höhere Komplexität implizieren. Bei der Interpretation dieser Untersuchung und der Konzeption von Maßnahmen sollte jedoch beachtet werden, dass ein gefundener Zusammenhang nicht zwingend eine Kausalität bedeutet. Wird zum Beispiel eine große Korrelation der Komplexität mit der Anzahl der Kommentare zu einem Artefakt dargestellt, so ist es nicht zielführend aus diesem Grunde die Kommentare zu reduzieren. Vielmehr hängen wahrscheinlich beide Werte von einer dritten Eigenschaft ab, die entscheidend für den Aufwand eines Artefakts ist. Ein Komplexi-

tätsmaß misst gegebenenfalls die „wahre“ Komplexität indirekt über Attribute, die von dieser abhängen. Die Untersuchung des Einflusses der Attribute kann also nur Hinweise auf mögliche Ursachen geben.

### 5.8.3 Earned-Complexity-Analyse

Der Komplexitätswert eines Produktmodells kann im Kontext vieler Entwicklungsmanagementmethoden benutzt werden. Denkbar sind zum Beispiel die Überwachung des Komplexitätswertes von Produktkomponenten und die Verknüpfung mit der Aufgabenplanung. Durch die automatisierbaren Messvorgänge kann der Ansatz außerdem gut in Entwicklungsumgebungen integriert werden. Die Einsatzmöglichkeiten sollen in diesem Abschnitt am Beispiel einer angepassten Earned-Value-Analyse (EVA) demonstriert werden. Die EVA wurde bereits im Rahmen der verwandten Ansätze im Abschnitt 3.4.1 vorgestellt.

Die Einbindung des Komplexitätswertes beruht im Wesentlichen darauf den Parameter „Budgeted Cost of Work Performed (BCWP)“ bzw. „Earned Value“ durch die Komplexität zu ersetzen. Dieses Vorgehen spricht die Problematik an, dass der BCWP schlecht objektiv bewertet werden kann. Der BCWP ergibt sich aus dem Budget und dem prozentualen Fertigstellungsgrad und stellt somit einen relativen Wert dar. Im Gegensatz dazu wird die Komplexität absolut gemessen und die Kennzahlen der Earned-Complexity-Analyse (ECA) erhalten somit eine leicht veränderte Semantik. Bei der ECA wird ein Komplexitätsmaß verwendet, das auf ein Projekt desselben Entwicklungssystems beruht. Somit können die im Verlauf des ECA-Projekts entstehenden Kosten mit der entstanden Komplexität verglichen werden.

In der in Abbildung 43 dargestellten ECA sind die tatsächlichen Kosten, die gemessene Komplexität über den Projektverlauf und der geplante Komplexitätsgewinn aufgetragen. Auf den Achsen sind außerdem der geplante Projektendtermin und die zu diesem Zeitpunkt erwartete Komplexität aufgetragen. Es werden außerdem zwei Kennzahlen angedeutet:

- Die Produktivität definiert sich über das Verhältnis von Earned Complexity und derzeitige Kosten. Dieser Wert zeigt an, wie effizient das Entwicklungssystem mit dem Projekt umgehen kann. Dieser Wert kann insbesondere für die Projektplanung und Projektsteuerung von großer Bedeutung sein.
- Der Leistungswert beschreibt das Verhältnis von erzielter und geplanter Komplexität. Dieser Wert ist nicht zwingend mit dem Sachfortschritt gleichzusetzen, da das Ziel eines Projektes oftmals bezogen auf den Umfang zunimmt.

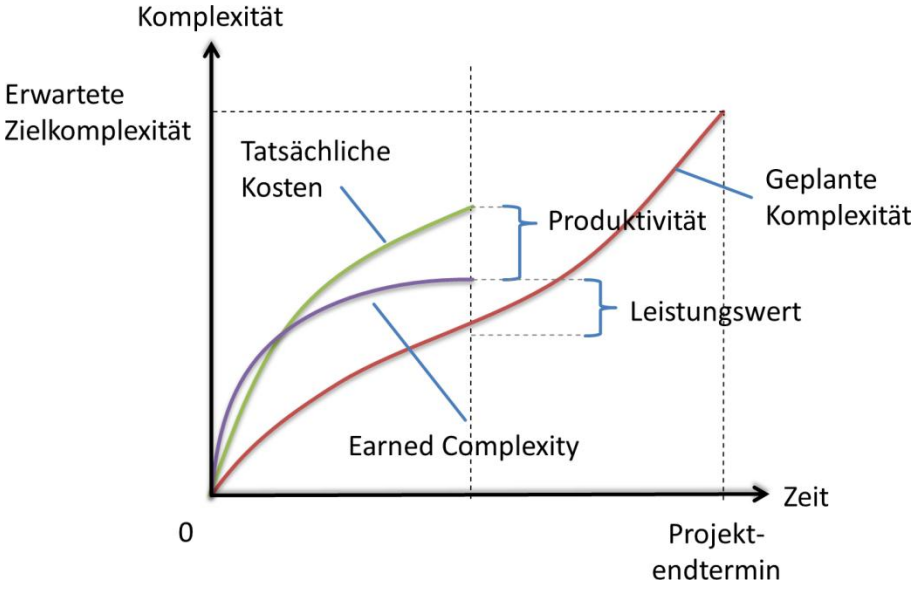


Abbildung 43: Earned-Complexity-Analyse

Im Gegensatz zur EVA muss die Earned Complexity am Projektendtermin nicht der erwarteten Zielkomplexität entsprechen. Hieraus ergeben sich neue Bewertungsmöglichkeiten für ein Projekt. Wenn zum Beispiel am Ende des Projekts die Earned Complexity höher liegt, so wurde in der Entwicklung mehr umgesetzt als geplant. Dies kann sinnvoll sein oder auch durch unnötige Mehrarbeiten (Gold Plating) verursacht werden. Diese Information erlaubt insbesondere auch eine bessere Bewertung von Budgetüberschreitungen. Der Komplexitätswert kann darstellen, ob die Mehrausgaben gerechtfertigt waren oder auf Probleme bei der Entwicklung zurückzuführen sind. Generell stellt die Earned-Complexity-Analyse eine sinnvolle Ergänzung zu vorhandenen Projektmanagementmethoden dar, da sie die häufig vorkommenden Veränderungen des Projektumfangs sinnvoll bewerten kann.

## 5.9 Abgrenzung

Im Rückblick auf den Stand der Technik, der in den Kapiteln 3 und 4 dargestellt wurde, wird in diesem Abschnitt diskutiert, wie sich der in diesem Kapitel dargestellte Ansatz differenziert. Insbesondere werden die konzeptionellen Unterschiede zwischen den Ansätzen und die zugrundeliegenden Gründe herausgearbeitet. Gleichzeitig werden die Einflüsse auf die Arbeit verdeutlicht und somit diese Vorarbeiten gewürdigt. Dieser Abschnitt zeigt auf, dass in diese Arbeit Ideen aus vielen verschiedenen Bereichen eingeflossen sind und für die Ziele dieser Arbeit neu kombiniert wurden.

### 5.9.1 Abgrenzung zu Ansätzen der Leistungsmessung in der Produktentwicklung

Die Messung der Entwicklungsleistung ist ein vielseitig adressiertes Thema in der Literatur. In Kapitel 3 wurden zunächst der Leistungsbegriff aus produktivitätstheoretischer Sicht als auch aus Sicht des FuE-Controllings differenziert. Es wurden außerdem bestehende Ansätze dargestellt, deren Abgrenzung in diesem Abschnitt erfolgt.

#### 5.9.1.1 Abgrenzung zu Ansätzen der Grenzproduktivität

Der erste auf der Theorie der Grenzproduktivität basierende Ansatz ist die Data Envelopment Analysis (DEA), die eine Ableitung der Grenzproduktivität über die Menge der beobachteten Firmen durch ein Optimierungsproblem vorschlägt (siehe Abschnitt 3.1.2). Auf Basis dieser Berechnung kann die technische Effizienz ermittelt werden. Der in dieser Arbeit entwickelte Ansatz kann ebenfalls zum Effizienzvergleich zwischen Unternehmen verwendet werden. Der Ansatz basiert ebenfalls auf einem linearen System wie die Bedingungen des Optimierungsproblems von (Farrell 1957), unterscheidet sich von der DEA jedoch wie folgt.

- Der Ansatz in dieser Arbeit definiert durch die Regression keine Produktivitätsgrenze und trifft daher keine Aussagen über eine technische Effizienz im Sinne von (Farrell 1957) wie DEA. Der Fokus liegt auf Aussagen zur Produktivität.
- Das zugrundeliegende mathematische Modell des Ansatzes dieser Arbeit ist ein lineares Gleichungssystem im Gegensatz zu dem Ungleichungssystem der DEA. Letzteres stellt bereits das Optimierungsproblem dar, während bei der Regressionsanalyse in dieser Arbeit das Optimierungsproblem zur Lösung des überbestimmten Gleichungssystems benutzt wird (Kleinste-Quadrate-Methode).
- Die Bewertung erfolgt in Bezug auf ein einzelnes Unternehmens durch DEA extrinsisch bezüglich anderer Unternehmen, während der Ansatz dieser Arbeit intrinsisch auf den Artefakten der Entwicklung in einem Unternehmen gerichtet ist. Dies hat den Vorteil, dass keine Produktionsdaten konkurrierender Unternehmen notwendig sind. Andere Firmen können jedoch immer noch bezüglich der Produktivität verglichen werden, wenn die produzierten Produktmodelle vorliegen.

Die DEA ist somit von Ziel und Methode her ein unterschiedlicher Ansatz, der eine nichtparametrische Messung der technischen Effizienz ermöglicht.

Der zweite vorgestellte Ansatz war die Stochastic Frontier Analysis (SFA), die eine Produktivitätsgrenze auf Basis einer Cobb-Douglas-Produktionsfunktion definiert (siehe Abschnitt 3.1.3). Die SFA besitzt mehr Gemeinsamkeiten mit dem in dieser Arbeit dargestellten Ansatz als DEA, da es ebenfalls auf einem parametrischen Modell beruht. Die

Parameter müssen in ähnlicher Weise durch eine Menge an Beobachtungen bestimmt werden. Die zentralen Unterschiede sind zum Teil dieselben wie bei der DEA:

- Der Ansatz dieser Arbeit trifft keine Aussagen zur Effizienz (siehe DEA).
- Die Bewertung erfolgt bezüglich der Firma intrinsisch (siehe DEA).
- Das zugrundeliegende Cobb-Douglas-Produktionsmodell bei SFA ist multiplikativ und beschreibt den Output $q$ durch die Inputs $x_i$. Das Modell dieser Arbeit ist linear und modelliert den Aufwand im Sinne der Komplexitätsdefinition von Seite 40 auf Basis von Aussagen zu Artefakten. Betrachtet man diese Aussagen als Output (der Produktentwicklung) und den Aufwand als Input, so lassen sich beide Modelle wie folgt grob vergleichen:

$$\text{SFA: } q = \prod x_i^{a_i} \quad \text{Komplexität: } x = \sum q$$

Es ist offensichtlich, dass beide Ansätze eine unterschiedliche Perspektive einnehmen und trotz Verwendung von parametrischen Modellen nicht vergleichbar sind. DEA und SFA sind nicht geeignet für die Ziele dieser Arbeit, besitzen aber durchaus ähnliche Aspekte wie der vorgestellte Ansatz. Die Beschreibung als verwandte Ansätze an dieser Stelle ist daher nicht als bestehende Alternativen zu verstehen sondern als Abgrenzung bezüglich den Modellen und Zielen der ökonomischen Theorie der Produktivitäts- und Effizienzbewertung.

#### 5.9.1.2 Abgrenzung zu Methoden des Projektmanagements

Aus dem Bereich des Projektmanagements wurden die Earned-Value-Analyse und die Meilensteintrendanalyse vorgestellt. Die EVA bietet eine quantitative Bewertung des Projektfortschritts auf Basis von Plankosten und Projektfortschritt. Hierzu wird jedoch eine quantitative Bewertung des Fertigstellungsgrades von Teilaufgaben (BCWP) benötigt. Im Gegensatz zu den anderen Parametern (Plankosten und Zeit) ist dieser Wert nicht in einfacher Weise objektiv zu ermitteln. Hierfür kann das Komplexitätsmaß verwendet werden, da es Produktmodelle als Entwicklungsleistung quantifizieren kann. Diese Erweiterung der Earned-Value-Methode wurde in Abschnitt 5.8.3 als Earned-Complexity-Methode dargestellt. Der Ansatz dieser Arbeit kann somit als ergänzend zur EVA gewertet werden.

Die Meilensteintrendanalyse stellt im Wesentlichen eine hilfreiche grafische Aufbereitung von historischen Planungsdaten dar. Sie ist daher nicht direkt mit dem Ansatz dieser Arbeit vergleichbar. Komplexitätswerte können jedoch als Grundlage für die Schätzung der Meilensteintermine genutzt werden. Der Ansatz kann somit als komplementär zur Meilensteintrendanalyse betrachte werden. Interessant ist jedoch die Idee eine Trendanalyse auf Komplexitätsmessungen zu beziehen.

#### 5.9.1.3 Einordnung und Abgrenzung zu anderen FuE-Metriken

Im Abschnitt 3.2 wurde das FuE-Controlling beschrieben und eine Strukturierung von FuE-Metriken anhand der Dimensionen des FuE-Controllings nach (Ojanen und Vuola 2006) dargestellt. Diese Dimensionen sollen herangezogen werden, um die in dieser Arbeit vorgestellte Methode im Bereich der FuE-Leistungsbewertung einzuordnen. Zunächst wird der Anwendungsbereich genauer eingegrenzt und im Folgenden erfolgt ein Vergleich zu anderen FuE-Metriken für diesen Bereich.

Die Tabelle 14 wiederholt die Ausprägungen der Dimensionen des FuE-Controllings aus Tabelle 3. Die Anwendungen für die vorgestellte Methode sind unterstrichen. Hervorzuheben ist, dass eine Komplexitätsmetrik nur bei Produktentwicklungen oder –Verbesserungen sinnvoll ist. Dies bedeutet, dass hierdurch keine Alternative zu Metriken wie zum Beispiel Anzahl der Veröffentlichungen oder Anzahl Patente geboten wird, die bei diesen FuE-Typen verwendet werden. Aus der Messgrundlage Produktmodell ergibt sich, dass es nur entwicklungsbegleitend oder am fertigen Produkt angewendet werden kann.

Tabelle 14: Anwendungsbereich des Komplexitätsmaßes im FuE-Controlling
Quelle: Angepasst nach (Ojanen und Vuola 2006)

| **Perspektive** | **Messzweck** | **Messlevel** | **FuE-Typ** | **Prozessphase** |
|---|---|---|---|---|
| Kunde | Begründung für Forschung | Industrie | Grundlagen-forschung | Input (Anforderungen) |
| Intern | Benchmarking | Firma | Kompetenz-aufbau | Entwicklungs-begleitend |
| Shareholder | Zuordnung von Ressourcen | Abteilung | Angewandte Forschung | Output (Fertiges Produkt) |
| Andere Stakeholder | Entwicklung von Verbesserungs-programmen | Projekt | Produkt-entwicklung | Rendite (durch Verkäufe etc.) |
| Forschung | Motivation und Belohnung | Individuen | Produkt-verbesserungen | |

In Abschnitt 3.2 wurden außerdem einige typische FuE-Metriken beschrieben. Der in dieser Arbeit vorgestellte Ansatz ist mit den einfacheren zählbasierten Metriken nicht vergleichbar. Strukturbasierte Varianten der Abhängigkeits- und Produktstrukturbewertung können hingegen als Spezialfall aufgefasst werden, die nur ein bestimmtes Attribut betrachten. Insgesamt ist ein Mangel an geeigneten Output-Maßgrößen für das FuE-

Controlling in dem durch Tabelle 14 skizzierten Anwendungsbereich erkennbar. Der Komplexitätsansatz ist daher als eine geeignete Alternative zu sehen.

#### 5.9.1.4 Abgrenzung zu Metriken für konzeptuelle Modelle

In Abschnitt 3.5 wurden Metriken für konzeptuelle Modelle diskutiert, die insbesondere im Bereich der modellgetriebenen Softwarenentwicklung bekannt sind. Die Erstellung von Modellmaßen auf Basis einer graphbasierten Darstellung eines Produktmodells haben die Ansätze im Bereich der konzeptuellen Modelle mit dem hier vorgestellten Ansatz gemein. Außerdem wird jeweils zur Definition von Knotenmetriken auf ein zugrundeliegendes Metamodell zurückgegriffen. Dies bedeutet, dass eine sehr nahe Verwandtschaft bezüglich des Messvorganges vorliegt.

Das wesentliche Unterscheidungsmerkmal zur existierenden Literatur liegt in der Herleitung des Modellmaßes. Die den Metriken für konzeptionelle Modelle zugrundeliegenden Hypothesen formulieren die meisten Autoren auf Basis ihrer Erfahrungen und somit auch stillschweigend auf Basis der jeweiligen Umgebung und den angenommen Prämissen. Wie bereits in der Beschreibung der konzeptuellen Metriken erwähnt, erfolgt leider in vielen Fällen keine empirische Validierung. Der hier vorgestellte Ansatz basiert hingegen auf der automatischen Hypothesenerzeugung auf Basis empirischer Daten. Generell können beide Vorgehensweisen zu ähnlichen Ergebnissen führen, wobei das in diesem Kapitel dargestellte Vorgehensmodell jedoch keine statistischen Kenntnisse voraussetzt. Somit kann das Vorgehensmodell und die Unterstützung durch eine Implementierung als computerbasierte Unterstützung der Suche nach konzeptuellen Metriken interpretiert werden. Der Fokus liegt jedoch auf umgebungsspezifische anstatt auf allgemeingültige Metriken.

### 5.9.2 Abgrenzung zu Ansätzen der Strukturanalyse

Der in dieser Arbeit vorgestellte Ansatz beruht auf der Analyse der Beziehungen zwischen Artefakten in einem graphbasierten Produktmodell. Ein wesentlicher Aspekt dieser Analyse ist die Bewertung von Strukturen wie zum Beispiel Produktaufbau und Abhängigkeiten. In einigen Methoden der Produktentwicklung werden ebenfalls ähnliche Analysen durchgeführt, jedoch mit zum Teil sehr unterschiedlichen Zielen.

#### 5.9.2.1 Abgrenzung und Bezüge zu matrizenbasierten Ansätzen

Aus dem Bereich des Engineerings und der Produktentwicklung stammen Ansätze zur Darstellung von Relationen im Produktmodell auf Basis einer Matrix. Der erste vorgestellte Ansatz war das House-of-Quality als Teil der Quality-Functions-Deployment-Methode (QFD) (siehe Abschnitt 4.1). Der in diesem Kapitel vorgestellte Ansatz basiert auf einer Modellierung, die von QFD stark inspiriert ist. Relationen zwischen Artefakten aus verschiedenen Partialmodellen der Produktentwicklung werden externalisiert.

Die Beschreibung ist jedoch vor allem an der vorher definierten Syntax der Matrizen und Listen orientiert und semantisch nicht so detailliert (zum Beispiel „mittlerer Zusammenhang“) und es werden keine expliziten Domänenmodelle benutzt. Eine detaillierte Abgrenzung bezüglich der Produktmodellmodellierung findet in der Vorarbeit (Hausmann 2008, 31) statt, dessen Grundprinzipien in der Modellierung für diesen Ansatz übernommen wurden.

Der wesentliche Unterschied ergibt sich aus der Zielsetzung des Verfahrens. QFD dient der Vermittlung der Kundenforderungen *„über die unterschiedlichen Begriffswelten hinweg bis zur Fertigungs-, Montag- und Qualitätskontrollplanung“* (Ehrlenspiel 2006, 215). Aus diesem Grund liegt der Schwerpunkt auf Inter- und nicht Intra-Domänen-Matrizen und bestimmten Standardrelationen. Die Methode ist daher eher mit der Anforderungsverfolgung (Requirements Traceability) aus der Softwareentwicklung und dem Systems Engineering vergleichbar. Dies wird auch durch einen Vergleich zu einer Definition der Anforderungsverfolgung ersichtlich. Die Vorwärts- und Rückwärtsrichtungen sind in QFD durch das System der Matrizen repräsentiert.

*„Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forwards and backwards direction […].”* (Gotel und Finkelstein 1994)

Bei der Komplexitätsbewertung spielen Relationen, die in der Anforderungsverfolgung wichtig sind, ebenfalls eine Rolle. Die Verknüpfung zu priorisierten Anforderungen kann ein wesentlicher Einflussfaktor sein. Dies wird jedoch in dem Ansatz dieser Arbeit nicht a priori angenommen, sondern wird wie für alle anderen Relationen und Strukturen geprüft, die einen Einfluss auf die Komplexität besitzen könnten. Insbesondere spielen Intra-Domänen-Strukturen eine wesentlich größere Rolle. Die Informationen, die durch das HOQ und andere Matrizen der QFD repräsentiert werden, können somit als ein Teilaspekt der Komplexitätsmessung aufgefasst werden.

Adjazenzmatrizen werden ebenfalls im Rahmen der Design Structure Matrices (DSM) eingesetzt (siehe Abschnitt 4.1.2). Auf DSMs basierende Ansätze unterscheiden somit vor allem durch die Art der Darstellung von dem hier vorgestellten Ansatz. Viele der Einschränkungen der Matrixmodellierung der nativen DSMs finden sich nicht in der allgemeineren graphbasierten Darstellung. Die auf dem EMF-Framework (siehe Abschnitt 6.2) basierende prototypische Implementierung zeigt diese Vorteile deutlich. Der Grundgedanke einer Darstellung von Produktmodellen als Graphen ist jedoch gleich und so lassen sich viele Parallelen erkennen, wenn von der Modellierungssyntax abstrahiert wird. So werden zum Beispiel Systemkomponenten durch Relationen in Beziehung gesetzt, was im Falle von DMMs auch partialmodellübergreifend geschieht. Die MDM von (Lindemann, Maurer, und Braun 2008) stellt in diesem Rahmen ein Metamodell über Partialmodelle dar. Die Abgrenzung wird durch die folgende Tabelle zu-

sammengefasst und um Aspekte der Modellierung mit ECore-Modellen im Prototypen aus Kapitel 6 ergänzt.

Tabelle 15: Abgrenzung des Ansatzes zum Structural Complexity Management

| | **Structural Complexity Management** | **Komplexitätsmessung von Produktmodellen** |
|---|---|---|
| **Graph-Darstellung** | Matrizen (DSM und DMM) und Metamatrix (MDM) | ECore (EReference als Kante, EObject als Knoten) |
| **Kantenart** | Gerichtet | Gerichtet |
| **Mehrfach-kanten** | Ja, durch mehrere Matrizen pro MDM-Zelle | Ja, in ECore bereits gegeben |
| **Kanten-gewichte** | Nein | Ja, als zu bestimmender Komplexitätsbeitrag von Kanten |
| **Komplexität** | Eigenschaft des Systems welches nach Aspekten unterschieden werden kann (Lindemann, Maurer, und Braun 2008, 29) | Eigenschaft von Artefakten und mittelbar über die Summe der Artefakte eine Eigenschaft des Systems |
| **Primäres Ziel** | Einsichten in das System zur Designverbesserung | Komplexitätsmaße für das Entwicklungscontrolling |
| **Bezeichnung Teilmodelle** | Domänen | Partialmodelle |
| **Struktur-erkennung** | Manuell und automatisch durch matrixbasierte Algorithmen | Automatisch durch Funktionswert der Attribute für Artefakte |
| **Struktur-bewertung** | Manuell | Automatisch durch Korrelationswerte |

Eine Inspiration für die Berücksichtigung von strukturellen Mustern in Graphen stellt jedoch die Strukturanalyse im Structural-Complexity-Management-Ansatz dar. Die Quantifizierung dieser Strukturen im Rahmen einer Pareto-Analyse stellt einen verwandten Ansatz zu den Strukturattributen aus Abschnitt 5.5.1 dar. Die Graphstrukturen, die in den Anhängen von (S. M. Maurer 2007) und (Lindemann, Maurer, und Braun 2008) aufgelistet werden, dienten zum Teil als Vorlage für die Implementierung von Attributmustern. Die Analysemethode von Lindemann u.a. beinhaltet jedoch keine Quantifizierung von Komplexität (und ist daher in Kapitel 2 nicht aufgeführt). Sie bietet

daher keine automatisierbaren Kennzahlen für das Entwicklungscontrolling und eignet sich nicht für die Ziele der Produktivitätsbewertung in dieser Arbeit.

#### 5.9.2.2 Abgrenzung zum Permeter-Ansatz

Die in dieser Arbeit vorgestellte Methode übernimmt einige Aspekte des Permeter-Ansatzes. Die Erstellung des Produktmodells für die Kalibrierung des Komplexitätsmodells und für die Messung basiert auf der Integration der Partialmodelle, wie sie durch den Permeter-Ansatz beschrieben wird. Jedoch wird auf die Verwendung semantischer Technologien zur prototypischen Implementierung verzichtet, welche ebenfalls für den vorgestellten Ansatz geeignet wären. Eine ausführliche Diskussion der verwendbaren Technologien und die Begründung der Auswahl für die prototypische Implementierung findet in Abschnitt 6.1 statt.

Während die Erstellung der Datenbasis starke Bezüge aufweist, unterscheiden sich die Analysemethoden der beiden Ansätze jedoch wesentlich. Die Ausrichtung des Permeter-Ansatzes ist von der Art der implementierbaren Metriken universeller gestaltet: Es können durch das Bausteinprinzip beliebige hierarchisch aufgebaute Metriken konstruiert werden. Allerdings muss der Benutzer wissen, wie sich die zu messende Eigenschaft im Produktmodell manifestiert. Es handelt sich entsprechend um eine statische Datenanalyse. Der Schwerpunkt des vorgestellten Ansatz besteht hingehen auf der explorativen Datenanalyse, die zur Entdeckung von Zusammenhängen in Datensätzen geeignet ist. Im konkreten Fall der Komplexitätsanalyse bezieht sich dies auf diejenigen Attribute und Strukturen, die die Komplexität in der Produktentwicklung bestimmen. Daher kommen im Gegensatz zum Permeter-Ansatz Verfahren des Data-Mining und der Statistik zur Anwendung. Die Unterschiede lassen sich auch durch den unterschiedlichen Zielfokus erklären. Die Komplexitätsanalyse dient primär der strategischen Produktivitätsbewertung und nur sekundär dem operativen Projektcontrolling. Der Schwerpunkt des Permeter-Ansatzes ist hingegen im Projektmanagement und dem operativen Projektcontrolling zu sehen. Die Unterschiede der beiden Ansätze sind in der folgenden Tabelle 16 zusammengefasst.

Tabelle 16: Unterschiede zwischen dem Permeter-Ansatz und der Komplexitätsmessung

| | **Permeter** | **Komplexitätsmessung von Produktmodellen** |
|---|---|---|
| **Graphdarstellung** | Semantische Graphen in Form von OWL-Modellen | ECore (EReference als Kante, EObject als Knoten) |
| **Primäres Ziel** | Operatives Projektcontrolling | Strategische Produktivitätsbewertung |
| **Gemessene Eigenschaften** | Beliebige durch hierarchische Metriken beschreibbare | Komplexität |
| **Analyse** | Statisch | Explorativ |
| **Strukturerkennung** | Keine | Durch entsprechende strukturelle Attribute |

# 6 Prototypische Implementierung

Die Konzeption des Ansatzes im Kapitel 5 stellt den Kern dieser Arbeit dar. Für die Validierung wird eine prototypische Implementierung vorgenommen, die in diesem Kapitel vorgestellt wird. Der Prototyp erfüllt für die Ziele dieser Arbeit folgende Aufgaben:

- Es wird gezeigt, dass der Ansatz prinzipiell implementierbar ist.
- Eine Bewertung der Skalierbarkeit kann vorgenommen werden.
- Die praktische Anwendbarkeit des Ansatzes wird demonstriert.
- Der Prototyp wird für die Umsetzung der Szenarien der Validierung benötigt. Eine manuelle Validierung ist aufgrund des hohen Rechenaufwands nicht praktikabel und sinnvoll.
- Es werden Hinweise für die Implementierung des Ansatzes auch in anderen Umgebungen und auf Basis von anderen Technologien gegeben. Der Ansatz dieser Arbeit ist nicht technologiegebunden.

Der Aufbau dieses Kapitels gliedert sich nach den notwenigen Komponenten für die Erstellung und Ausführung eines Komplexitätsmaßes. Er besitzt hierdurch einen leicht verschiedenen Aufbau als das konzeptionelle Kapitel. Jedes Unterkapitel besitzt jedoch gut erkennbare Bezüge zu bestimmten Abschnitten der Konzeption, wie in Tabelle 17 dargestellt.

Tabelle 17: Zuordnung der Abschnitte von Implementierung und Konzeption

| **Unterkapitel Implementierung** | **Abschnitte Konzeption** |
|---|---|
| 6.1 Auswahl der Basistechnologien | 5.3.1 Eigenschaften des Graphen |
| 6.2 Modellierung der Produktmodelle | 5.3.2 Definition eines Metamodells |
| 6.3 Erfassung der Referenzentwicklung | 5.3.3 Transformation der nativen Daten |
| | 5.4 Definition und Gewichtung der Artefakte |
| 6.4 Erzeugung des Komplexitätsmaßes | 5.5 Generierung und Anwendung der Attribute |
| | 5.6.1 Regression |
| 6.5 Anwendung des Maßes auf Produktmodelle | 5.7 Durchführung der Messung |

## 6.1 Auswahl der Basistechnologien

Die Basistechnologien beschreiben die Implementierungsgrundlage des Prototyps und beeinflussen maßgeblich dessen Leistungsfähigkeit. Der wichtigste Aspekt bei der Auswahl ist die Abbildbarkeit von Produktmodellen als gerichtete Graphen und der effiziente Umgang mit den Modellen. Auf die Entwicklung eines dedizierten Produktmodellframeworks sollte verzichtet werden, da bereits sehr effiziente Modellierungs-Frameworks zur Verfügung stehen. Bei der Auswahl einer geeigneten Technologie kann grob zwischen graphenorientierten und modellierungsorientierten Frameworks unterschieden werden.

- Graphenorientierte Frameworks sind näher an der graphformalen Ebene und unterstützen oftmals auch eine sehr große Anzahl an Knoten. Einige Vertreter fokussieren den Visualisierungsaspekt, während andere große Datenmengen in Form einer Graphendatenbank effizient ablegen und verwalten können. Beispiele dieser Kategorie sind JUNG[13] und Jena[14].
- Modellierungsorientierte Frameworks ermöglichen die Definition eines Metamodells und darauf basierend die Erstellung eines Instanzenmodells. Der Schwerpunkt bei der Modellierung liegt auf der Ausdrucksfähigkeit der erstellbaren Metamodelle. Insbesondere die modellgetrieben Softwareentwicklung ist ein wichtiges Anwendungsszenario. Vertreter dieser Kategorie sind viele UML-Werkzeuge wie Open Modeling Framework[15] oder Poseidon for UML[16] und allgemeinere Frameworks wie EMF[17].

Im Rahmen dieser Arbeit findet kein allgemeiner Vergleich dieser Frameworks statt, da dies den Umfang im Rahmen des Implementierungskapitels stark übersteigen würde. Stellvertretend wird ein Vergleich zwischen Jena und EMF gezogen. Beide Technologien wurden vom Autor als Implementierungsgrundlage evaluiert und sind prinzipiell geeignet.

Jena ist ein Framework für die semantische Modellierung und unterstützt über Profile die Sprachen RDF (Resource Description Framework) und OWL. Im Rahmen dieser Sprachen können Metamodelle in Form von Klassen- und Relationsdefinitionen erstellt werden, welche wiederum RDF- beziehungsweise OWL-Modelle darstellen. Prinzipiell gibt es keine Trennung zwischen einem Metamodell und einem Instanzenmodell. Diese Trennung wird jedoch in der Praxis häufig durch die Aufteilung in zwei Teilgraphen

[13] http://jung.sourceforge.net/
[14] http://openjena.org/
[15] http://www.sdml.info/projects/omf/
[16] http://www.gentleware.com/new-poseidon-for-uml-8-0.html
[17] http://www.eclipse.org/modeling/emf/

durchgeführt. Jena besitzt keine integrierte Visualisierung und stellt eine Graphendatenbank bereit, die über die Abfragesprache SPARQL angesprochen werden kann.

EMF (Eclipse Modeling Framework) ist ein Projekt der Eclipse Community und somit eng mit der Entwicklungsumgebung Eclipse verknüpft. Die Modellierungssprache von EMF ist ECore und eng verwandt mit dem UML-Standard MOF. Entsprechend sind die Basiskonzepte der Sprache Klassen und Interfaces. EReferences sind den Relationen in Klassendiagrammen von UML gleichzusetzen; EAttributes bezeichnen Klassenfelder. ECore ist somit weiter vom Graphformalismus entfernt, aber kann immer noch als Graph im Sinne des Ansatzes dieser Arbeit betrachtet werden. Aus ECore-Modellen kann durch einen Quellcodegenerator eine Java-API erzeugt werden, mit dem ein Instanzenmodell abgebildet werden kann. EMF besitzt außerdem einen grafischen Editor für ECore-Modelle. Instanzenmodelle können sowohl als Dateien als auch in dedizierten Datenbanken abgelegt werden.

Für die prototypische Implementierung wurde schließlich EMF ausgewählt, da es umfangreiche Werkzeuge zur Verfügung stellt und die Implementierung im Vergleich zu Jena weniger Einarbeitung benötigt. Die Tabelle 18 unten zeigt eine Gegenüberstellung von Konzepten in beiden Technologien, die demonstriert, dass sie prinzipiell austauschbar sind.

Tabelle 18: Vergleich der Konzepte in Jena und Eclipse EMF

| | **Jena** | **Eclipse EMF** |
|---|---|---|
| **Graphensprache** | RDF | ECore |
| **Metamodell** | RDF-Schema | ECore-Modell |
| **Instanzenmodell** | RDF-Graph | EMF-Modell |
| **Programmatischer Zugriff** | Generische API | Statische API<br>(generische API optional) |
| **Serialisierung** | RDF/XML-Datei | XMI-Datei |
| **Konzept** | RDF-Schema-Klasse | EClass |
| **Instanz** | RDF-Knoten | EObject |
| **Relation** | RDF-Property (RDF-Knoten) | EReference |
| **Attribut** | RDF-Property (Literal) | EAttribute |

## 6.2 Modellierung der Produktmodelle

Die Transformation der Produktmodelle wurde auf konzeptioneller Ebene in Abschnitt 5.3.3 besprochen. In diesem Kapitel soll dies praktisch auf Basis des EMF-Frameworks diskutiert werden. Für die Abbildung der Produktmodelle als EMF-Modelle ist zunächst eine Modellierung der Konzepte der Domänen in ECore notwendig. Da die EMF-Modellierung insbesondere im Bereich des Systems Engineering eingesetzt wird, existieren zum Teil bereits Modelle. In anderen Fällen können die entsprechenden ECore-Modelle aus Referenzmodellen oder aus Spezifikationen abgeleitet werden. Liegt kein brauchbares Modell vor, so müssen die Konzepte als neues ECore-Modell erfasst werden.

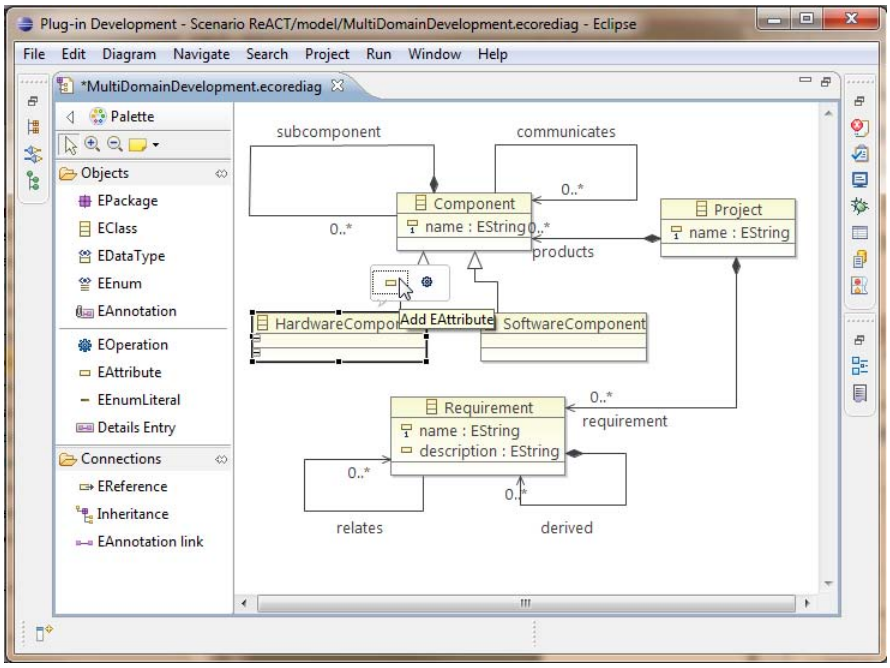


Abbildung 44: Eclipse ECore Editor mit Entwicklungsmodell

EMF bietet für die Bearbeitung von ECore-Modellen einen grafischen Editor an, der in den Prototyp integriert wird. In der Abbildung 44 ist der Editor mit einem einfachen Modell einer Hard- und Softwareentwicklung abgebildet. Die Klasse `Component` besitzt in diesem Beispiel zwei Unterklassen `HardwareComponent` und `SoftwareComponent`. Außerdem werden Anforderungen an das Produkt über `Requirement` definiert. Die Festlegung, welche dieser Klasse eine Artefaktklasse darstellt, geschieht im Modellierungsschritt noch nicht. Mit Hilfe des Editors wird eine ECore-Datei erstellt, aus welcher schließlich eine API für die weiteren Schritte zur Abbildung des Referenzmodells erzeugt wird.

## 6.3 Erfassung der Referenzentwicklung

Nach der Modellierung der Konzepte muss ein Referenzproduktmodell erfasst werden, auf dessen Basis das Komplexitätsmaß erzeugt wird. Dieselbe Erfassungsmethode ist außerdem später bei der Anwendung des Komplexitätsmaßes notwendig. Durch die Verwendung von Eclipse EMF im Prototyp kann auf viele kompatible Technologien zugrückgegriffen werden, die die Implementierung einer Lösung zur Erfassung eines Produktmodells vereinfachen oder sogar vollständig übernehmen.

### 6.3.1 Erstellung oder Transformation des Produktmodells

Aus dem zugrundeliegenden ECore-Modell kann neben der API durch EMF ein einfacher Editor generiert werden. Der Editor ist baumartig aufgebaut und verwendet als Grundlage für die Struktur EReferences, die als Aggregation definiert sind. (In Abbildung 44 sind diese mit ausgefüllter Raute am Ende der beinhaltenen Klasse dargestellt.) Außerdem lassen sich die Werte der EReferences und EAttributes über ein Eigenschaftenfenster (Properties View) festlegen.

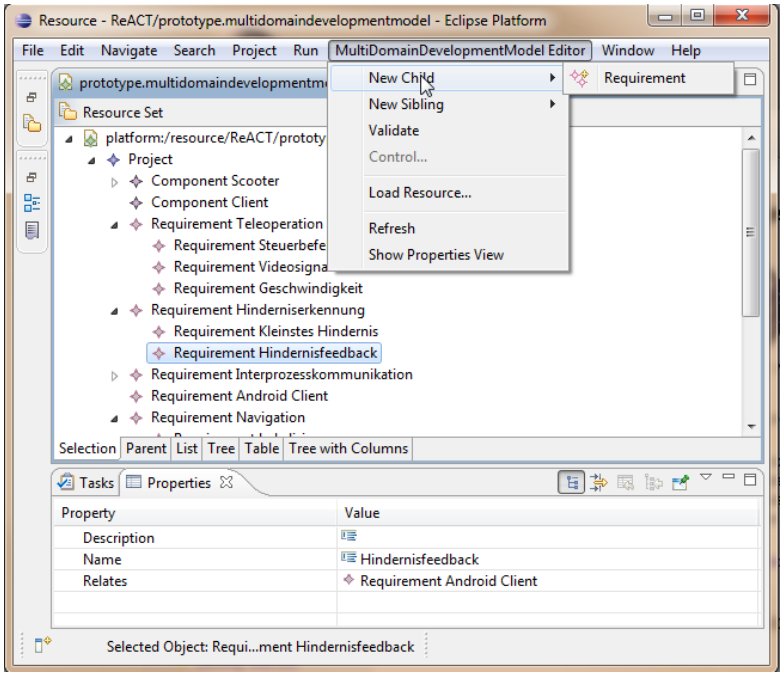


Abbildung 45: EMF-Modelleditor

Die Abbildung 45 zeigt einen Modelleditor, der auf Basis des ECore-Modells aus Abbildung 44 erzeugt wurde. Als Unterknoten der Anforderung Hindernisfeedback kann nur eine weitere Anforderung definiert werden, da im ECore-Modell die Klasse `Requirement` nur eine Aggregationsbeziehung (EReference `derived` in Abbildung 44) zu sich selbst besitzt.

Der einfache EMF-Editor bietet eine schnelle und flexible Möglichkeit zur Beschreibung von einfachen Produktmodellen. Es können somit manuell auch Modelle erfasst werden, die nur in informeller Form, wie zum Beispiel als Textdokument oder als Skizze auf dem Whiteboard, vorliegen. Bei größeren Modellen mit mehr als 1000 Elementen oder sehr detaillierten Modellen kann dieses Vorgehen jedoch unübersichtlich werden. Außerdem müssen die Produktmodelle jedes Mal manuell erfasst oder überarbeitet werden, was einer vollautomatisierten Ausführung des späteren Komplexitätsmaßes widerspricht. In der Regel liegen Produktmodelle bereits in formeller Form als Ergebnis des Entwicklungsprozesses vor. Diese Modelle können durch die Integration von Parsern in EMF-Modelle überführt werden.

Da EMF bereits eine hohe Verbreitung besitzt, können einige Parser aus Open-Source-Projekten übernommen werden. Zum Beispiel bietet das Eclipse-Projekt MoDisco[18] einen Parser für Java-Quellcode. Im Rahmen der EMF-basierten Projekte XText[19] und EMFText[20] können domänenspezifische Sprachen entworfen werden, die auf einem E-Core-Modell basieren. Hierdurch werden Abbildungen zwischen nativen Sprachen und einem EMF-Modell definiert, die im Rahmen der Erfassung eines Produktmodells als Parser eingesetzt werden können. So stellt EMFText auf der Projektseite bereits viele Syntaxabbildungen, zum Beispiel für Latex, OWL2 oder Petrinetze, zur Verfügung.

### 6.3.2 Gewichtungsassistent

Nach der Erfassung des Produktreferenzmodells in Form eines EMF-Modells muss definiert werden, welche Objekte in dem Modell Artefakte darstellen und somit einem Aufwand zuzuordnen sind. Da EMF über ein Klassenmodell verfügt, ist es sinnvoll auf diesem aufzusetzen und bestimmte Klassen als Artefaktklassen auszuzeichnen. Der Prototyp verwendet hierzu einen Assistenten, der die Definition der Artefaktklassen unterstützt. Ein Assistent oder Wizard in Eclipse ist ein Dialogfenster, das aus mehreren Seiten besteht und Vorgänge wie das Erstellen eines neuen Projektes unterstützt, bei denen zunächst bestimmte Informationen abgefragt werden müssen.

[18] http://eclipse.org/MoDisco/

[19] http://www.eclipse.org/Xtext/

[20] http://www.emftext.org/

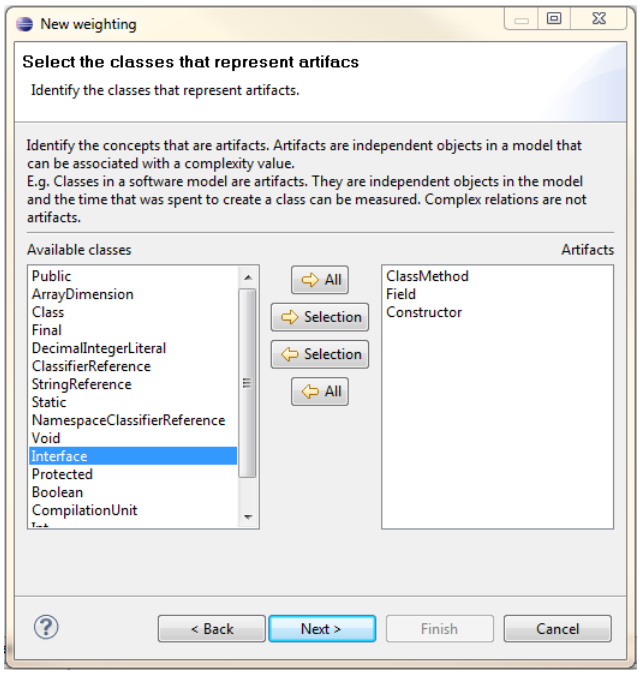


Abbildung 46: Assistentenseite zur Definition der Artefaktklassen

Der Assistent des Prototyps besitzt drei Seiten, die nacheinander alle Informationen sammeln um das Modell zur Gewichtung der Artefakte vorzubereiten.

1. Auf der ersten Seite kann der Benutzer EMF-Modelle in Form von XMI-Dateien auswählen. Wählt der Benutzer mehrere Dateien aus, so handelt es sich hierbei um die Darstellung der Partialmodelle des Produktmodells. Diese werden in ein Gesamtmodell überführt, wie konzeptuell in Abschnitt 5.3.3.3 beschrieben.
2. Die zweite Seite dient der Definition der Artefaktklassen und ist in Abbildung 46 dargestellt. Der Benutzer kann die Klassen der EMF-Modelle aus dem vorherigen Schritt in einer mengenorientierten Ansicht auswählen. Es ist möglich, beliebig viele Klassen als Artefaktklassen zu definieren.
3. Im letzten Schritt muss der Benutzer die Einheit festlegen, die zur Gewichtung der Artefakte verwendet wird und somit auch die Einheit des Komplexitätsmaßes letztlich darstellt. Außerdem kann eine Gesamtkomplexität festgelegt werden, die auf alle Instanzen der Artefaktklassen gleichmäßig aufgeteilt wird. Dies sind jedoch lediglich die initialen Werte für die Gewichtungen der Artefakte.

Nach Durchlauf aller Seiten erzeugt der Assistent eine neue Gewichtungsdatei, die sowohl das vereinigte Referenzproduktmodell, die Definition der Artefaktklassen als auch die initiale Gewichtung der Artefaktinstanzen umfasst. Diese Daten werden entsprechend als Gewichtungsdatei (`.weighting`) bezeichnet und müssen im Folgenden bearbeitet werden, bevor ein Komplexitätsmaß erzeugt werden kann.

### 6.3.3 Gewichtungseditor

Durch den Gewichtungsassistenten wurde eine Gewichtungsdatei erstellt, die die Dateiendung `.weighting` besitzt. Dieser Endung ist im Prototyp ein Editor zugeordnet, der diese Dateien bearbeitet. Dieser Gewichtungseditor vereinfacht das Eintragen der einzelnen Komplexitätswerte der Artefakte des Referenzproduktmodells und kann auch durch Benutzer verwendet werden, die keine Erfahrung in statistischer Datenerfassung besitzen. Prinzipiell kann die Gewichtungsdatei jedoch auch durch einen Texteditor bearbeitet werden, da es sich hierbei um eine XML-Datei handelt.

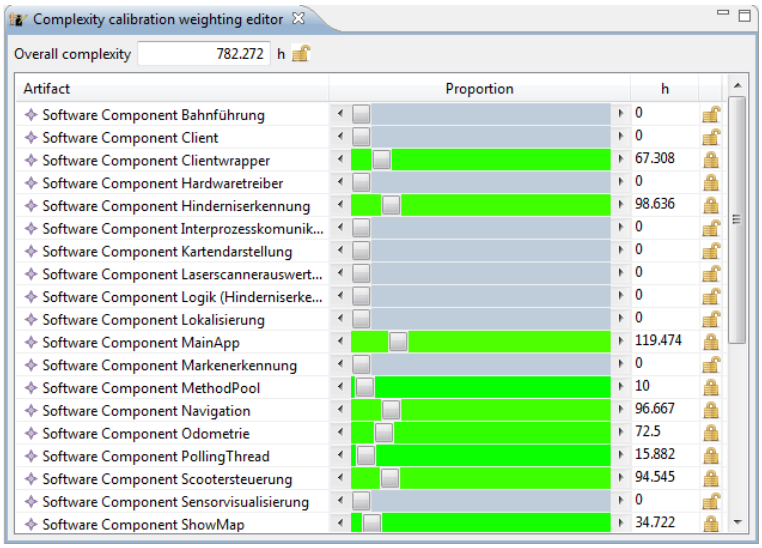


Abbildung 47: Gewichtungseditor

Der Editor ist in Abbildung 47 dargestellt und in zwei Bereiche aufgeteilt: Eine Kopfzeile fasst den Gesamtkomplexitätswert des Referenzproduktmodells zusammen und zeigt die Einheit der Komplexität. Unterhalb der Kopfzeile können die Einzelwerte der Artefakte bearbeitet werden, wozu ein Schieberegler und ein Eingabefeld zu Verfügung stehen. Die Schieberegler werden bei einer größeren Anzahl von Artefakten aus Übersichts- und Performancegründen nicht angezeigt.

Neben den Einzelwerten und dem Gesamtwert wird ein Schloss angezeigt, durch die sich jeder Wert beliebig fixieren lässt. Wenn zum Beispiel der Gesamtwert fixiert ist und ein Einzelwert erhöht wird, so werden automatisch andere Einzelwerte reduziert, sofern sie nicht auch fixiert sind. Der Editor stellt somit die Konsistenz des Modells sicher. Falls sich ein Wert nicht ohne Inkonsistenz ändern lässt, so lässt der Editor diese Änderung nicht zu. Die Motivation hinter diesem Mechanismus sind die unterschiedlichen Datenquellen, die sowohl relative als auch feste Werte umfassen können.

## 6.4 Erzeugung des Komplexitätsmaßes

Durch die Erstellung und Bearbeitung einer oder mehrerer Gewichtungsdateien liegen alle notwendigen Daten für die Erzeugung eines Komplexitätsmaßes durch den Prototyp vor. Diese Daten müssen nun im Folgenden analysiert und in ein Modell der Komplexität umgewandelt werden, was in den Abschnitten 5.5 und 5.6 bereits konzeptionell beschrieben wurde. Dieser Vorgang verläuft zum großen Teil automatisiert und wird durch einen weiteren Assistenten unterstützt, der im Folgenden dargestellt wird. Die Implementierung der Attributerzeugung und der eigentlichen Regression wird in separaten Abschnitten detailliert.

### 6.4.1 Assistent zur Erstellung des Komplexitätsmaßes

Die Erzeugung des Komplexitätsmaßes soll keine statistischen Vorkenntnisse erfordern. Um dies zu realisieren, wird der Benutzer durch einen Assistenten durch den Prozess der Analyse und Modellbildung geführt. Der Übersicht halber sind nicht alle Seiten abgebildet. Der Assistent besteht aus folgenden fünf Seiten:

1. **Erfassung der Gewichtungsdateien:** Auf dieser Seite werden ein oder mehrere Gewichtungsdateien ausgewählt. Liegen mehrere Dateien für verschiedene Partialmodelle vor, so werden diese Modelle zunächst vereinigt.
2. **Auswahl der Attributmuster:** Fortgeschrittene Benutzer können auswählen, welche Attributmuster bei der Analyse des Referenzproduktmodells verwendet werden. Die im Prototyp zur Verfügung stehenden Attributmuster sind im Anhang aufgelistet. In der Standardeinstellung werden alle verfügbaren Attributmuster einbezogen.
3. **Regressionseinstellungen:** Auf dieser Seite werden Informationen zum Referenzmodell angezeigt und der Benutzer kann auswählen, welche Einheit das Komplexitätsmaß verwendet werden soll. Es ist möglich eine neue Einheit mit einem Gesamtwert für das Referenzmodell zu definieren. Zum Beispiel kann der Wert des Referenzmodells auf 1 gesetzt werden und der Einheit einen abstrakten Namen gegeben werden. Wenn der Benutzer von dieser Seite auf die nächste Seite wechselt, so werden die Generierung der Attribute und die Messung der Attributwerte vorgenommen. Um den Fortschritt nachvollziehbar darzustellen, werden ein Fortschrittsbalken sowie eine Statusmeldung angezeigt, was in Abbildung 45 dargestellt ist. Die Implementierung dieses Vorgangs wird in Abschnitt 6.4.2 detailliert.

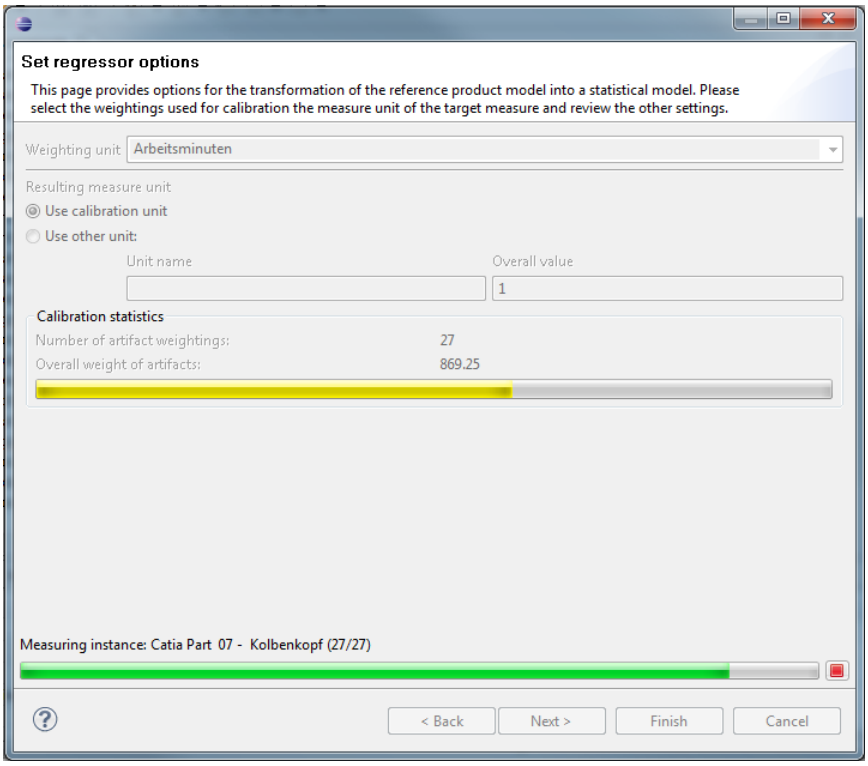


Abbildung 48: Darstellung der Erzeugung und Messung der Attribute

4. **Auswahl des Komplexitätsmodells:** Auf dieser Seite kann der Benutzer interaktiv das Komplexitätsmodell festlegen, indem er Attribute entfernt oder hinzufügt. Die Auswahl der Attribute beschränkt sich hierbei der Übersicht halber auf die Auswahl signifikanter Attribute, wie sie in Abschnitt 5.6.1.6 beschrieben wird. Eine Vorauswahl wird auf Basis der Algorithmen getroffen, die in Abschnitt 5.6.1.7 beschrieben werden. Der Benutzer hat also die Möglichkeit die Ergebnisse der automatischen Analysen inhaltlich zu prüfen und gegebenenfalls anzupassen. Zur Unterstützung werden die in Abschnitt 5.6.1.8 beschriebenen Kennzahlen angezeigt.
5. **Serialisierung des Komplexitätsmaßes:** Diese Seite schließt die Erstellung des Komplexitätsmaßes ab. Neben der Festlegung eines Dateinamens für das Komplexitätsmaß kann außerdem eine Beschreibung angegeben werden, die das Referenzmodell charakterisiert und den Anwendungsbereich des Komplexitätsmaßes eingrenzt.

Bei Abschluss des Assistenten wird ein Komplexitätsmaß in Form einer `.cmeasure`-Datei erstellt. Hiermit ist die Erzeugung des Komplexitätsmaßes abgeschlossen.

### 6.4.2 Erzeugung und Messung der Attribute

Die Erzeugung der Attribute basiert auf der Auswahl von Attributmustern durch den Benutzer im 2. Schritt des Assistenten zur Erstellung des Komplexitätsmaßes. Die At-

tributmuster werden durch Klassen implementiert, die nach dem Programmiermuster Fabrik gestaltet sind; ihre Aufgabe besteht in der Erzeugung der benötigten Attributinstanzen. Sie werden durch Unterklassen der Klasse `RegressorFactory` implementiert. Jeweils eine Singleton-Instanz dieser Klassen erzeugt durch die Methode `createInstances()` auf Basis des Referenzproduktmodells die Attribute in Form von Instanzen der jeweiligen Implementierung des Interfaces `Regressor`. Die Bezeichnung „Regressor“ geht auf die Bedeutung der Attribute als Variablen der Regression zurück (vgl. Tabelle 4).

Das Verhältnis von `Regressor` und `RegressorFactory` wird in Abbildung 49 dargestellt. Im oberen Teil des Klassendiagramms sind das Interface `Regressor` sowie dessen Subinterfaces dargestellt. Die jeweiligen Fabrikklassen sind von der abstrakten Basisklasse `RegressorFactory` abgeleitet. Die in der Abbildung gezeigten `RegressorFactory`-Implementierungen stellen nur eine Auswahl dar. Eine Auflistung der Attributmuster des Prototyps erfolgt im Anhang.

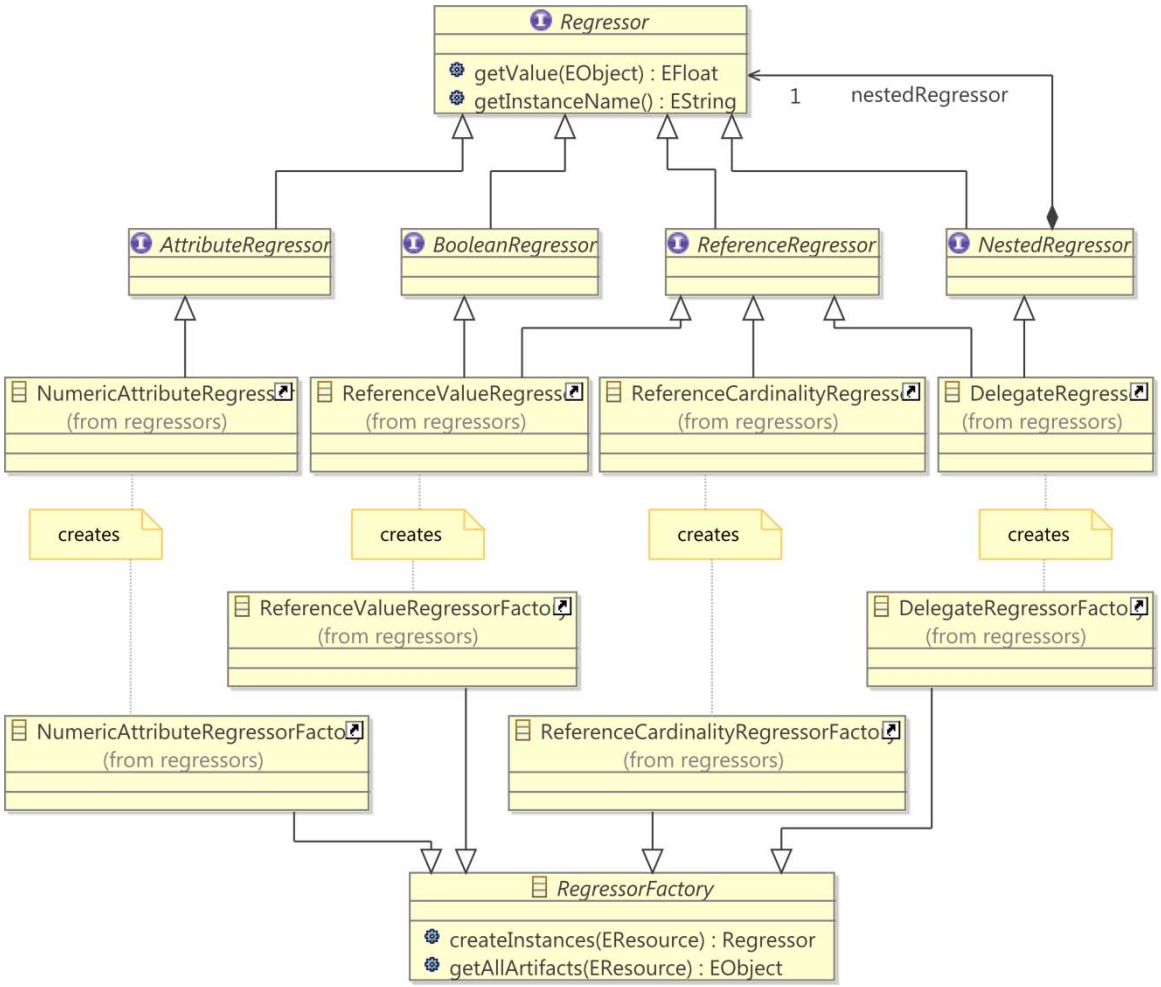


Abbildung 49: Klassenarchitektur für Attribute und Attributmuster

Das Referenzproduktmodell wird auf jede `RegressorFactory` angewendet und die erzeugten `Regressor`en in einer Liste gesammelt. Nach der 3. Seite des Assistenten wird schließlich jeder `Regressor` auf jedes Artefakt angewendet, indem der Methode `Regressor.getValue()` Artefakte in Form von `EObject`s übergeben werden. Hierdurch entsteht eine rechnerinterne Matrix von Attributwerten, die die Implementierung der Matrix $X$ des Regressionsmodell aus Abschnitt 5.5.4 darstellt.

### 6.4.3 Regressionsimplementierung

Für die statistische Auswertung der gesammelten Daten wird auf das Data-Mining-Framework Weka in der Version 3.7.3 zurückgegriffen. Weka[21] ist eine Forschungssoftware der Machine Learning Group an der Universität Waikato. Die Software verfügt über Methoden zur Aufbereitung, Analyse und Darstellung großer Datenmengen und wird in (M. Hall u. a. 2009) diskutiert. Sie kann sowohl über eine Benutzeroberfläche angesteuert werden als auch in Programme als Bibliothek eingebettet werden. Im Prototyp wird die Software in Form eines Plug-Ins eingebunden.

Für die Auswahl signifikanter Attribute werden im Prototyp sowohl existierende Attributfilter der Software Weka benutzt, als auch zusätzliche Filter implementiert. Wie in Abschnitt 6.4.1 beschrieben, werden die ausgewählten Attribute dem Benutzer auf der vierten Seite des Assistenten zur Auswahl präsentiert und eine Vorauswahl erklärender Attribute vorgenommen. Neue Regressionsmodelle auf Basis einer veränderten Auswahl der Attribute können ohne relevante Verzögerung berechnet werden. Die Bewertung dieser Modelle erfolgt ebenso sofort. Die Kennzahl $R^2$ wird durch Weka berechnet, während der Konvergenzwert des Komplexitätsmaßes (siehe Seite 120) durch eine Implementierung im Prototyp ergänzt wird.

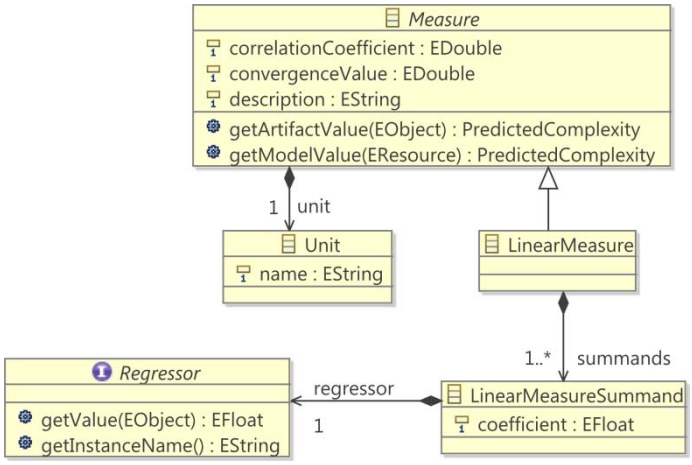


Abbildung 50: Klassenarchitektur für Serialisierung des Komplexitätsmaßes

[21] Projektseite: http://www.cs.waikato.ac.nz/~ml/weka/

Die Speicherung des Komplexitätsmaßes als Datei erfolgt über die XMI-Schnittstelle von EMF. In dem serialisierten Modell werden die in Abschnitt 5.6.2 beschriebenen Informationen eines Komplexitätsmaßes durch ein EMF-Modell beschrieben, das auf den in Abbildung 50 dargestellten Klassen basiert: Das Wurzelelement des Maßes ist eine Instanz der Klasse `LinearMeasure`, die von der abstrakten Oberklasse `Measure` ableitet. Diese verweist auf eine bestimmte Einheit (Instanz der Klasse `Unit`) und besteht aus ein oder mehreren Summanden, die die Attribute des Komplexitätsmaßes mit deren Koeffizienten darstellen. Jedem Summand ist die entsprechende Instanz eines Regressors zugeordnet.

## 6.5 Anwendung des Maßes auf Produktmodelle

Die Anwendung des Komplexitätsmaßes wurde konzeptionell in Abschnitt 5.7 behandelt. Die Implementierung im Prototyp stellt nur eine sehr einfache Art dar, wie das Komplexitätsmaß ermittelt werden kann. Diese Methode wird im Rahmen der Validierung eingesetzt, um manuell die ermittelten Kennzahlen zu prüfen. Sinnvoll in einem größeren Anwendungskontext ist insbesondere die Einbettung in einem Controlling-Framework. Die prototypische Implementierung kann in dieser Hinsicht ohne wesentliche Änderungen in automatisierte Umgebungen integriert werden.

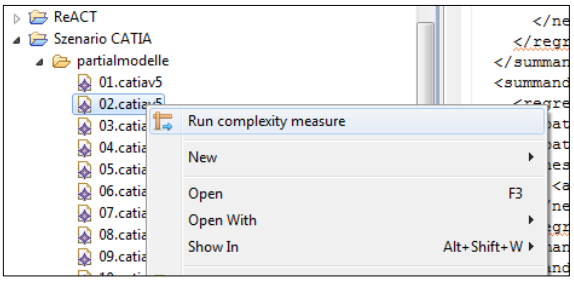


Abbildung 51: Anwendung des Komplexitätsmaßes über ein Kontextmenü

Das Komplexitätsmaß liegt als Datei vor und kann direkt zur Messung eingesetzt werden. Zur Berechnung eines Komplexitätswertes muss das Maß auf ein oder mehrere EMF-Modelle angewendet werden. Im Prototyp erfolgt dies über ein Kontextmenü, wie in Abbildung 51 dargestellt. Für jeden Summanden (vergleiche Abbildung 50) wird der Wert über den zugehörigen `Regressor` gemessen. Der Wert des Komplexitätsmaßes ergibt sich aus der Summe aller `Regressor`en über alle Artefakte im zu vermessenden Produktmodell.

# 7 Validierung des Ansatzes

Der in dieser Arbeit vorgestellte Ansatz zur Komplexitätsmessung von Produktmodellen ist für ein breites Anwendungsszenario ausgelegt. Er soll für Produktmodelle aus verschiedenen Domänen eingesetzt werden und mit anderen Methoden des Entwicklungsmanagements kombinierbar sein. Daher bedarf der Ansatz einer umfassenden Validierung, die in diesem Kapitel durchgeführt wird. Zur Validierung werden auch Vergleiche zu bestehenden Messansätzen und anderen Methoden gezogen, um den Mehrwert des Ansatzes, nach Möglichkeit auch quantitativ, darzulegen.

Die Validierung ist in drei Fallbeispiele aufgegliedert. Diese Szenarien sind voneinander unabhängig und besitzen jeweils einen anderen Validierungsfokus. Das erste Fallbeispiel zeigt anhand der Untersuchung einer Software die generelle Machbarkeit und die Qualität der Messungen in einem Umfeld mit einer Modellart. Dieses Beispiel wurde als Eingangsszenario gewählt, da es ein relativ homogenes Produktmodell mit einem einfachen Projektrahmen darstellt. Im darauf folgenden Fallbeispiel werden die Produktmodelle eines autonomen, experimentellen Fahrzeuges untersucht. An der Entwicklung dieses komplexeren Produktes sind verschiedene Domänen beteiligt und der Schwerpunkt der Validierung liegt auf der Einsetzbarkeit in einer Multi-Domänen-Entwicklung. Im letzten Fallbeispiel wird die Methode auf CAD-Modelle angewendet, die sich wesentlich von den Produktmodellen der anderen Beispiele unterscheiden. Hierdurch wird die Übertragbar des Ansatzes auf andere Disziplinen in den Vordergrund gestellt. Die drei Szenarien und deren fokussierten Fragestellungen sind in Tabelle 19 dargestellt.

Tabelle 19: Übersicht der Szenarien und Fragestellung der Validierung

| **Name** | **Domänen** | **Primäre Fragestellungen** |
|---|---|---|
| **Permeter** | Software | Machbarkeit, Skalierbarkeit und Genauigkeit |
| **ReACT** | Software, Sensorik, Hardware | Multi-Domänen-Anwendbarkeit und Flexibilität der prototypischen Implementierung |
| **CATIA** | Maschinen- und Anlagenbau, Fahrzeuge | Domänenunabhängigkeit und Flexibilität der prototypischen Implementierung |

## 7.1 Fallbeispiel: Permeter-Software

Dieses Fallbeispiel umfasst die Anwendung des vorgestellten Ansatzes auf eine existierende Software. Hierzu wird der Quellcode der Software in ein EMF-Modell überführt

und eine Komplexitätsanalyse durchgeführt, die sowohl Komplexitätsfaktoren identifiziert als auch ein Komplexitätsmaß erzeugt. Dieses Szenario kann als relativ einfach angesehen werden, da bewusst nur eine Modellart untersucht wird. Das heißt, die in Abschnitt 5.3.3.3 beschriebene Integration von mehreren Partialmodellen wird in diesem Fall nicht durchgeführt. Validierungskriterien sind insbesondere die generelle Machbarkeit des Ansatzes inklusive der Skalierbarkeit für große Modelle und die Genauigkeit der Messung. Für diese Aufgabe bietet dieses Szenario folgende Vorteile:

- Durch das relativ homogene Produktmodell können Gründe für Ausreißer und Messabweichungen einfacher identifiziert werden. Es sind außerdem keine weiteren Schritte zur Aufbereitung der Daten neben der Transformation in ein EMF-Modell zu tätigen.
- Die Komplexitätsuntersuchung von Quellcode ist ein bereits in der Software-Engineering-Literatur intensiv erforschtes Thema. Die wichtigsten Grundlagen dieser Ansätze wurden bereits in Kapitel 2 vorgestellt und insbesondere der Abschnitt 2.1.3.3 umfasst die wichtigsten Software-Komplexitätsmaße. Da einige Implementierungen von Quellcode-Metriken verfügbar sind, besteht dadurch eine Vergleichsmöglichkeit des Ansatzes zum Stand der Technik im Bereich Software.
- Es sind automatisierte Transformationen des Quellcodes verfügbar, welche diesen in ein sehr detailliertes EMF-Modell abbilden können. Dadurch ist ein sehr umfangreiches Modell verfügbar, das zur Prüfung der Skalierbarkeit geeignet ist.
- Der Autor des Ansatzes ist mit der untersuchten Software vertraut und kann die Zwischenergebnisse der Komplexitätsanalyse inhaltlich auswerten. Insbesondere können die Aussagen zu den Komplexitätsfaktoren interpretiert werden.

Insgesamt kann dieses Szenario somit als „idealer Einstieg“ in die Validierung angesehen werden, da es vertraut, überschaubar und insbesondere vergleichbar ist. Es können allerdings nicht alle Anforderungen geprüft werden, die für diese Arbeit in Abschnitt 2.2 formuliert wurden. Insbesondere die domänenunabhängige und anpassbare Messung und deren Anwendung muss durch die anderen Szenarien bestätigt werden.

### 7.1.1 Hintergrund

Unter Permeter ist die Gesamtheit an Software zu verstehen, die im Rahmen des Permeter-Ansatzes (siehe Abschnitt 4.2.1) an der Abteilung Business Engineering entwickelt wurde. Nach Abschluss der Arbeiten an der Version 1.2, die als Grundlage zur prototypischen Evaluierung der Arbeiten von (Hausmann 2008) und (Strickmann 2008) diente, wurde die Entwicklung einer neuen Hauptversion begonnen. Diese sollte vor allem die

Defizite der ersten Hauptversion beheben und die Möglichkeiten der semantischen Modellierung von Produktmodellen ausreizen. Hierzu gehörten folgende Fähigkeiten:

- Aufbau eines hierarchischen Produktmodells, das zur Laufzeit erweitert werden kann. Hierarchisch bedeutet in Zusammenhang mit den in Permeter verwendeten OWL-Modellen, dass einzelne Modellelemente andere OWL-Modelle repräsentieren können.
- Die Eigenschaften aller Modelle wurden mit einem RDF-Metamodell beschrieben und alle Modelle in einer RDF-Datenbank verwaltet.
- Ein Partialmodell konnte zur Laufzeit durch mehrere Sichten beschrieben werden. Das heißt ein Modellelement konnte gleichzeitig Instanz verschiedener Klassen aus verschiedenen Sichten sein. Diese Fähigkeit wird durch die Abbildung 52 skizziert: Einige RDF-Knoten (unterste Ebene) sind als Instanz der Klasse `Part` sowohl Elemente der Produktstruktur (mittlere Ebene) als auch Instanz der Klasse `Implementation` im Anforderungsmodell (obere Ebene).

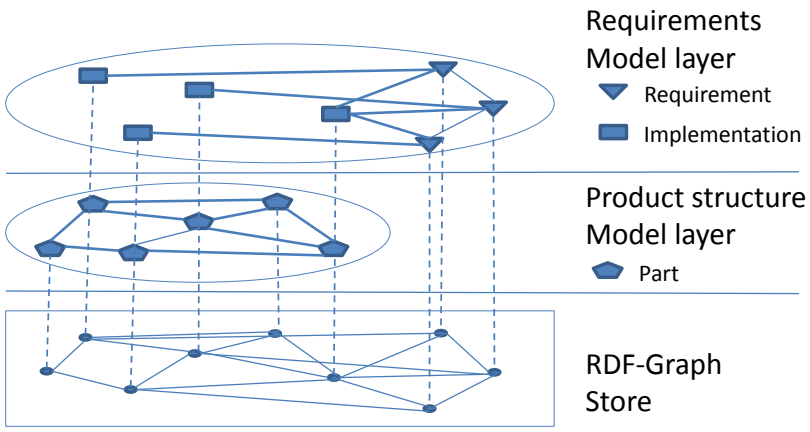


Abbildung 52: Geplante Architektur für Modellsichten in Permeter 2.0

Die auf semantischen Technologien beruhende Variante der Permeter-Software wurde jedoch vor der Fertigstellung zugunsten des Eclipse Modeling Frameworks aufgegeben, welches eine größere Verbreitung und ein wesentlich bessere Werkzeugunterstützung besitzt. Zu den ausführlichen Gründen für die Auswahl der EMF-Technologie siehe Abschnitt 6.1. Für das Validierungsszenario wird der letzte Stand dieses Entwicklungszweiges verwendet.

Die Software ist durch das Eclipse-Framework in Plug-Ins aufgeteilt, die die Funktionalitäten der Software sowie deren Abhängigkeiten untereinander kapseln. Die Abhängigkeiten und Schnittstellen zum Eclipse-Framework werden durch Konfigurationsdateien beschrieben, die im Rahmen dieser Validierung nicht ausgewertet werden. Diese Relationen werden ebenfalls durch den Java-Quellcode ausgedrückt, welcher als Basis für die Untersuchung verwendet wird. Ziel der Analyse ist es, ein Komplexitätsmodell der

Java-Implementierung abzuleiten, welches zur Messung von ähnlicher Software geeignet ist. Außerdem sollen die Wirkzusammenhänge zwischen Eigenschaften der Klassen und dem Aufwand quantitativ untersucht werden.

### 7.1.2 Anwendung des Vorgehensmodells

Die Validierung orientiert sich an dem in Kapitel 5 vorgestellten Vorgehensmodell für die Erstellung eines Komplexitätsmaßes. Der Abschnitt zur Regression stellt in diesem Fallbeispiel die wichtigsten Ergebnisse der Untersuchung dar.

#### 7.1.2.1 Modellierung

Gegenstand der Untersuchung ist das Java-Produktmodell der Software. Bestandteile des Produktmodells sind somit Klassen, Interfaces, Methoden und andere Elemente eines Java-Programmes. Hierzu sind bereits einige Metamodelle in EMF frei verfügbar, die jeweils auf Werkzeugen mit einem anderen Fokus beruhen. Diese Metamodelle sind zum Teil sehr detailliert und daher gut für die Prüfung der Skalierbarkeit des Ansatzes geeignet. Für dieses Fallbeispiel wurden zwei Metamodelle ausgewählt, die auf den Werkzeugen JaMoPP und MoDisco beruhen, die im Folgenden kurz vorgestellt werden. Die Metamodelle können als Plug-Ins ohne zusätzliche Konfiguration in die Testumgebung der prototypischen Implementierung eingebettet werden.

JaMoPP[22] (Java Model Parser and Printer) ist ein Eclipse-basiertes Werkzeug um Java-Quellcode in EMF-Modelle zu parsen und wieder Java-Quellcode aus EMF-Modellen zu generieren. Ziel ist die Verbindung von Modellierung und Programmierung und der Einsatz beliebiger EMF-Werkzeuge zur Entwicklungsunterstützung. JaMoPP unterstützt die gesamte Java5 Syntax und besitzt daher ein sehr umfangreiches Metamodell mit 233 Klassen und 104 Relationen.

[22] http://jamopp.inf.tu-dresden.de

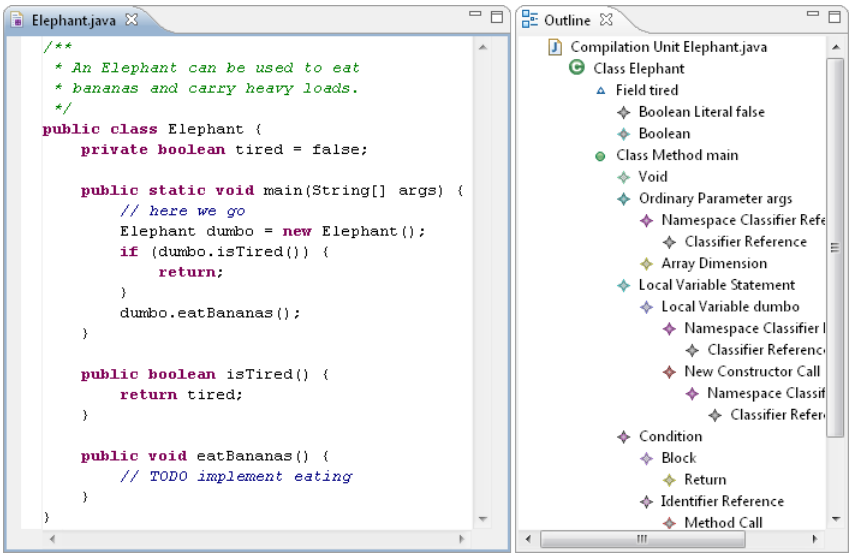


Abbildung 53: Java-Quellcode und JaMoPP-Model
Quelle: http://www.emftext.org/index.php/EMFText_Concrete_Syntax_Zoo_Java_5

Die Abbildung 53 zeigt den Quellcode einer Java-Klasse und das entsprechende JaMoPP-Java-Modell, welches den Quellcode bis zur Statement-Ebene abbildet. Der Fokus liegt vor allem auf Ebene einzelner Dateien und die Unterstützung während der Programmierung.

Das Eclipse-Projekt MoDisco[23] unterstützt die Migration von Legacy-Systemen durch die Modellierung dieser Systeme in EMF. Aus diesen Modellen sollen Analysen zur Unterstützung der Migration und neuer Quellcode erzeugt werden. Bestehender Quellcode und Modelle werden durch Discoverer-Komponenten geparst. Im Rahmen des Projektes wurde auch eine solche Komponente für Java-Quellcode geschaffen. Im Gegensatz zu JaMoPP wird vor dem Hintergrund einer Systemmigration vor allem die Gesamtsoftware betrachtet. Das Java-Metamodell umfasst die Syntax von Java 3 und beinhaltet 126 Klassen.

#### 7.1.2.2 Transformation

Der Java-Quellcode des Produktmodells ist über ein Software-Repository verfügbar und kann ohne weitere Vorbereitung durch die Transformationen der beiden Werkzeuge JaMoPP und MoDisco in EMF-Modelle umgewandelt werden. Hierbei unterscheidet sich das Vorgehen der beiden Werkzeuge geringfügig. JaMoPP ist auf eine zeitnahe Darstellung von Java-Modellen auf Klassenebene als EMF-Modell optimiert und wird entsprechend als Parser in das Eclipse-Framework eingebettet. Der Parser wird bei jeder Änderung einer Java-Datei automatisch aufgerufen und eine entsprechende Abbildung in Form einer XMI-Datei erzeugt. Um diese Dateien zu einem Produktmodell zu vereinigen wurde ein Assistent implementiert, der diese Aufgabe übernimmt. Die Auswahl

[23] http://www.eclipse.org/MoDisco/

eines Verzeichnisses mit XMI-Dateien in einem JaMoPP-Projekt ist in Abbildung 54 dargestellt.

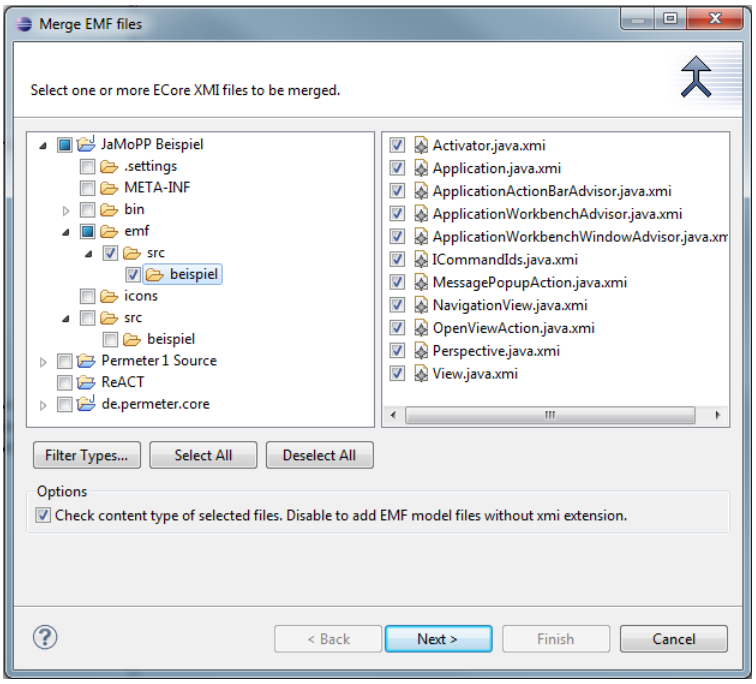


Abbildung 54: Assistent zum Vereinigen von XMI-Dateien

Die Software MoDisco ist vor dem Hintergrund der Migrationsunterstützung hingegen zum einmaligen Einsatz konzipiert. Sie wandelt entsprechend durch einen Assistenten ein gesamtes Java-Projekt in Eclipse in eine einzelne große XMI-Datei um.

Beide Transformationen führen insgesamt zu ähnlichen Produktmodellen in Form einer XMI-Datei. Sie sind außerdem in ihrem Detailgrad konfigurierbar und können entsprechend große detaillierte Modelle mit zum Beispiel mehr als 10 Megabyte XMI-Code erzeugen. Diese Implementierung der Transformation zeigt, dass der Ansatz im Punkt der Modellierung und Transformation ausreichend skalierbar ist. Außerdem ist der Aufwand für die Erzeugung der XMI-Modelle überschaubar und der Vorgang kann automatisiert werden. Für die weitere Validierung wurde das Produktmodell des MoDisco-Parsers verwendet.

#### 7.1.2.3 Definition und Gewichtung der Artefakte

Mit Hilfe der in Abschnitt 6.3 dargestellten Assistenten wurde die Aufbereitung des Produktmodells für die Regression durchgeführt. Als Artefaktklasse wurde die EMF-Klasse `CompilationUnit` definiert, welche im Produktmodell eine Java-Quellcodedatei repräsentiert. Die EMF-Klassen für Java-Klassen und Java-Interfaces wurden nicht als Artefakte definiert, da innere Klassen sowie anonyme Klassen nicht als gut abgrenzbare Komponenten geeignet sind.

Die Gewichtung der Artefakte wurde anhand der Anzahl an Versionen der Quellcodedateien im verwendeten Versionsverwaltungssystem Subversion vorgenommen. Die Eingabe erfolgt über den in Abschnitt 6.3.3 vorgestellten Gewichtungseditor. Die Auswahl dieser Kennzahl als Referenzwert erfolgte unter der Annahme, dass neue Versionen im Mittel einem ähnlichen Umfang an Aufwand entsprechen. Da hierbei jedoch größere Ausreißer nicht auszuschließen sind, wurde für jedes Artefakt eine manuelle Überprüfung der Versionsgeschichte durchgeführt. Waren die Änderungen in ein oder mehreren Versionen minimal, so wurde der Gewichtungswert entsprechend korrigiert. Außerdem wurden einige aus anderen Projekten kopierte Java-Dateien ausgeschlossen.

Als Einheit der Gewichtung wurde die Einheit „commits“ definiert in Anlehnung an den Vorgang der Versionserzeugung. Das gewichtete Produktmodell umfasste 3350 Knoten, wovon es sich bei 44 um Artefakte handelte. Durch den hohen Detailgrad des Produktmodells hatte die resultierende Gewichtungsdatei eine Größe von ungefähr 38 Megabyte.

#### 7.1.2.4 Regression

Der Assistent zur Erstellung eines Komplexitätsmaßes (siehe Abschnitt 6.4.1) wurde im Folgenden benutzt um die Regression durchzuführen. Unter Auswahl aller verfügbaren Attributmuster wurden 321 Attribute erzeugt. Die Generierung der Attribute hat ungefähr 10 Minuten auf einem normal ausgestatteten PC in Anspruch genommen. Die Messung der Attributwerte am Referenzproduktmodell dauerte ungefähr 2 zusätzliche Minuten. Anschließend erfolgte die Auswahl der Attribute nach dem in Abschnitt 5.6.1 dargestellten Vorgehen, welches in zu vernachlässigender Zeitdauer durchgeführt wurde. Die Tabelle 20 zeigt auf, wie viele Attribute auf welcher Stufe vorhanden waren. Nach Durchführung der Regression wurde das Komplexitätsmaß mitsamt den ausgewählten 6 Regressoren als Datei gespeichert.

Tabelle 20: Auswahl Attribute im Validierungsszenario Permeter

| **Selektionsstufe** | **Siehe Abschnitt** | **Attributbewertung** | **Verbleibend** |
|---|---|---|---|
| Vorhandene Attribute | 5.6.1.4 | Erzeugung Attributmuster | 321 Attribute |
| Sinnvolle Attribute | 5.6.1.5 | Vollständigkeit, Streuung | 51 Attribute |
| Signifikante Attribute | 5.6.1.6 | CFS-Attributfilter | 13 Attribute |
| Erklärende Attribute | 5.6.1.7 | Akaike-Kriterium | 6 Attribute |

### 7.1.3 Bewertung des Komplexitätsmaßes

Das entstandene Komplexitätsmaß wird in diesem Abschnitt unter drei Gesichtspunkten bewertet. Zunächst soll geprüft werden, wie konsistent das Modell ausgehend von dem

der Kalibrierung zugrundeliegenden Produktmodell ist. Es wird also untersucht, wie gut sich das Modell an die Ausgangsdaten anlegt. In einem zweiten Abschnitt wird ein aufbauender Vergleich zu zwei Komplexitätsmetriken aus dem Bereich des Software-Engineering gezogen. Danach wird das erzeugte Komplexitätsmaß auf ein unabhängiges aber vergleichbares Produktmodell angewendet um eine Vergleichsmessung durchzuführen.

#### 7.1.3.1 Konsistenz des Modells

Für die Bewertung des resultierenden Komplexitätsmaßes wurde zunächst das Bestimmtheitsmaß $R^2$ ermittelt, welches bei 0,951 liegt. Dies ist bereits ein Indiz für eine gute Annäherung des Regressionsmodells an das Referenzmodell. Zur genaueren Untersuchung wurden außerdem die Daten des Regressionsmodells exportiert und in Abbildung 55 visualisiert. Auf der X-Achse ist die vorgesagte Komplexität aufgetragen, die sich auf Basis des Komplexitätsmodels ergibt. Die Y-Achse stellt die tatsächlich gemessene Komplexität für die Kalibrierung dar. Die Abstände zu der Ausgleichsgeraden entsprechen den Residuen $e$ des Regressionsmodells (siehe Abschnitt 5.5.4). Gut erkennbar sind drei Artefakte, die eine außerordentlich hohe Komplexität aufweisen. Hierbei handelt es sich um sehr große Klassen, wie zum Beispiel die Implementierung der Modellverwaltung in der Software Permeter. Insgesamt ist eine gute Anpassung des Modells an die Ausgangsdaten festzustellen.

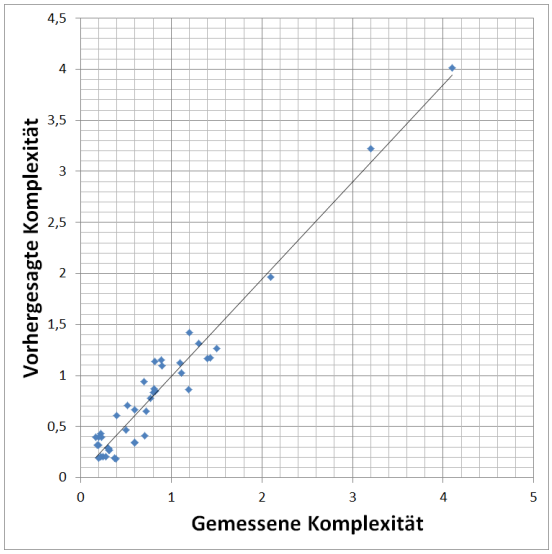


Abbildung 55: Plot des Regressionsmodells im Validierungsszenario Permeter

#### 7.1.3.2 Vergleich zu bestehenden Komplexitäts-Metriken

Insbesondere im Bereich des Software Engineerings existieren bereits zahlreiche Komplexitäts-Metriken. In Abschnitt 2.1 wurden einige dieser Metriken und deren Bezug zum Komplexitätsbegriff vorgestellt. Da in dieser Validierung der Mehrwert des vorge-

stellten Ansatzes untersucht werden soll, wurde ein Vergleich zu populären automatisierbaren Metriken für Softwarekomplexität durchgeführt. Als Metriken wurden Total Lines of Code (inklusive Kommentare) und die McCabe-Komplexität ausgewählt (siehe Abschnitt 2.1.3.3). Für die im Validierungsszenario betrachteten Artefakte wurden deren entsprechende Werte mit Hilfe eines Eclipse Plug-Ins ermittelt und pro Artefakt aggregiert. Die Abbildung 56 stellt den Zusammenhang zwischen den beobachteten Metrikwerten und den Gewichtungswerten für die Kalibrierung des Komplexitätsmaßes dar. Es kann ebenfalls ein linearer Zusammenhang festgestellt werden, auch wenn dieser bei weitem nicht so stark ausgeprägt ist wie beim kalibrierten Komplexitätsmaß des vorgestellten Ansatzes.

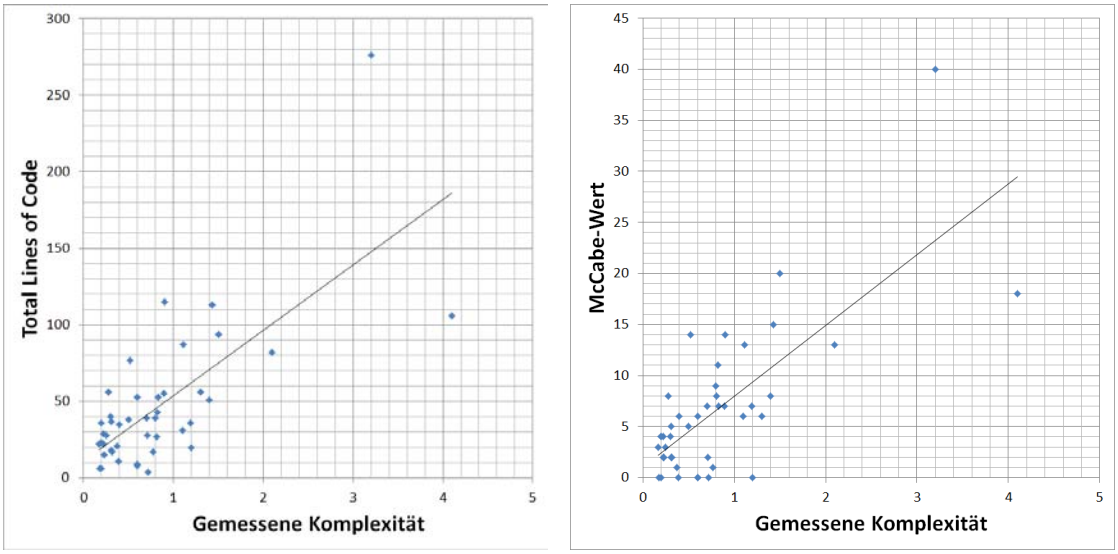


Abbildung 56: Plot der Software-Metriken Lines of Code und McCabe-Komplexität

Für die beiden Zahlenreihen wurden außerdem das Bestimmtheitsmaß über den Pearsonschen Korrelationskoeffizienten berechnet und in Tabelle 21 dem Komplexitätsmaß gegenüber gestellt.

Tabelle 21: Vergleich Bestimmtheitsmaß

| | **Kalibriertes Komplexitätsmaß** | **Mc-Cabe-Komplexität** | **Lines of Code mit Kommentare** |
|---|---|---|---|
| **Bestimmtheitsmaß** | 0,95194824 | 0,5449333 | 0,51713007 |

Das wesentlich bessere Abschneiden des kalibrierten Komplexitätsmaßes ist dadurch zu erklären, dass durch die Regression das Bestimmtheitsmaß bezüglich der Kalibrierungsdaten optimiert wird. Dieser Vergleich kann somit nicht derart interpretiert werden, dass das erzeugte Komplexitätsmaß generell besser geeignet ist; es ist lediglich

bezüglich des vorgegebenen Komplexitätsverständnisses im Sinne von erzeugten Versionen optimiert. Durch die Kalibrierung wurden automatisch diejenigen Faktoren zur Messung selektiert, die den größten Zusammenhang zu dieser individuellen Komplexität besitzen. Dies entspricht der im Anforderungskatalog (Abschnitt 2.2.3) formulierten Anpassbarkeit des Komplexitätsmaßes auf eine bestimmte Umgebung (Anforderung PA2) und eine Formulierung der Komplexität als eine Aufwandskennzahl (Anforderung KA1). Durch die automatische Generierung und Auswahl der Attribute und die Bildung des Regressionsmodells wurde das Komplexitätsmaß bezüglich dieser Zielrichtung optimiert.

#### 7.1.3.3 Vergleichsmessung

Ursprung des Referenzmodells ist das Basis-Plug-In der Software Permeter. Die Verwendung einer anderen Komponente dieser Software als Vergleichsmodell ist naheliegend, da somit beide Produktmodelle miteinander vergleichbar sind. Es wurde hierfür ein Plug-In von Permeter zur Darstellung der semantischen Modelle ausgewählt. Unter Anwendung des in Abschnitt 5.7 beschriebenen Vorgehens zur Durchführung der Messung wurde die Vergleichskomponente durch das Komplexitätsmaß auf Ebene der Einzelartefakte quantifiziert. Es wurde entsprechend dieselbe Transformation für die Umwandlung der nativen Daten verwendet um das graphbasierte Produktmodell zu erhalten. Für die Durchführung der Bewertung wurde zusätzlich wieder eine Gewichtung der Artefakte auf Basis der Versionsgeschichte in der Quellcodeverwaltung durchgeführt. Die durch das Komplexitätsmaß gemessenen Werte wurden dann mit den tatsächlichen Werten verglichen. Die Abbildung 57 zeigt die Ergebnisse dieser Untersuchung: Ein Zusammenhang zwischen gemessener und vorhergesagter Komplexität ist eindeutig zu erkennen. Allerdings liegen Punkte entfernt von der Ausgleichsgeraden. Dies bedeutet, dass eine einzelne Artefaktmessung einen hohen Fehler besitzen kann. Durch die aggregierten Messwerte der Artefakte des Produktmodells können diese aber relativiert werden.

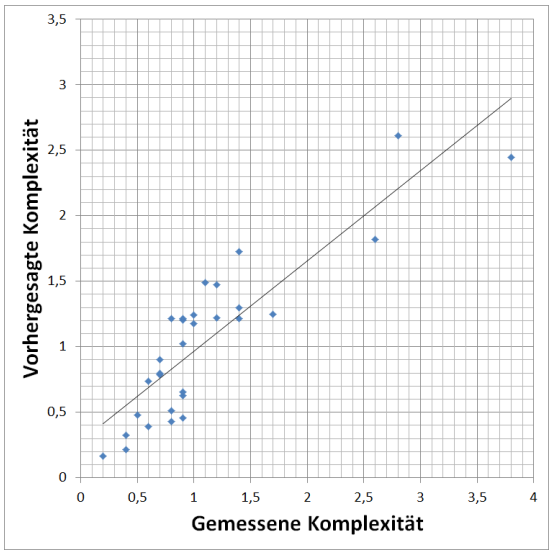


Abbildung 57: Plot der Vergleichsmessung Permeter

Statistisch weißt das Modell ein Bestimmtheitsmaß von $R^2 = 0{,}76$ auf. Die zuvor genannten Abweichungen der Einzelmessungen sind in dieser Kennzahl deutlich eingegangen. Aus Sicht der Produktivitätswertung ist jedoch der Vergleich der summierten Messungen interessanter. Hier wird bei einem vorgesagten Wert von 31,08 im Verhältnis zum gemessenen Wert 33,2 ein Fehlerwert von etwa 6,8% erzielt. Dies ist angesichts der Unterschiede zwischen den beiden Plug-Ins ein relativ guter Wert.

## 7.2 Fallbeispiel: ReACT

Im ersten Fallbeispiel Permeter wurde der Fokus auf die generelle Machbarkeit und Skalierbarkeit des Ansatzes gelegt. Aus diesem Grund wurde ein homogenes Produktmodell in Form eines Java-Quellcodes gewählt. In diesem Fallbeispiel wird der Schwerpunkt auf die Anwendbarkeit des Ansatzes auf ein heterogenes Produktmodell gelegt, welches nicht nur Elemente enthält, die in Form von Quellcode vorliegen. Die Herausforderung dieses Fallbeispiels liegt in der Modellierung des Produktmodells, welches sich über mehrere Domänen erstreckt. Bei der betrachteten Produktentwicklung handelt es sich um ein komplexes Produkt mit Hardware- und Softwarekomponenten. Insbesondere spielen die Abhängigkeiten zwischen den verschiedenen Sichten auf das Produkt eine wichtige Rolle. Es handelt sich somit um eine sinnvolle Ergänzung zum ersten Fallbeispiel und adressiert jene Anforderungen, die noch nicht zufriedenstellend geprüft werden konnten.

### 7.2.1 Hintergrund

Die Projektgruppe ReACT (Remote Android Controlled Transporter) war ein einjähriges studentisches Forschungsprojekt von Master- und Diplomstudierenden am Depart-

ment für Informatik in Kooperation mit dem OFFIS-Institut. Die Durchführung der Projektgruppe fand vor dem wissenschaftlichen Hintergrund der Projekte SaLsA[24] und Cognilog[25] statt, die sich mit autonomen Systemen in der Logistik beschäftigen. Ziel des Projektes war die prototypische Entwicklung eines Fahrzeuges, das autonom navigiert. Zur Überwachung und Missionskontrolle soll dieses Fahrzeug über ein mobiles Gerät in Form eines Handys mit Android-Betriebssystem steuerbar sein. Die Abbildung 58 zeigt die letzte Version dieses Fahrzeuges.

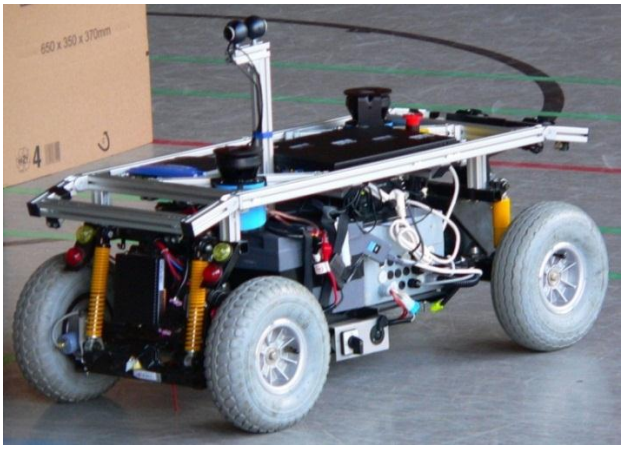

Abbildung 58: Der entwickelte Prototyp eines autonomen Fahrzeuges
Quelle: Abschlussbericht der Projektgruppe ReACT (ReACT-Gruppe 2011)

Die Implementierung dieses Fallbeispiels durch den in dieser Arbeit vorgestellten Ansatz beruht auf dem Abschlussbericht (ReACT-Gruppe 2011), den internen Projektdokumenten sowie Interviews mit den beteiligten Entwicklern. Das Produktmodell ergibt sich entsprechend aus einer Vielzahl von beschreibenden Texten, Abbildungen, Tabellen, Quellcode und Hardwarespezifikationen.

### 7.2.2 Anwendung des Vorgehensmodells

Die Validierung orientiert sich an dem in Kapitel 5 vorgestellten Vorgehensmodell für die Erstellung eines Komplexitätsmaßes. Der Abschnitt zur Modellierung stellt in diesem Fallbeispiel die wichtigsten Ergebnisse der Untersuchung dar.

#### 7.2.2.1 Modellierung

Da sehr viele nicht formell spezifizierte Informationen des Produktmodells modelliert werden müssen, ist im Gegensatz zu dem ersten Fallbeispiel kein Metamodell vorhanden, das bereits alle Informationen abbilden kann. Zwar sind Metamodelle von im Projekt verwendeten Sprachen wie C++ und Java verfügbar, aber insbesondere die interes-

[24] Webseite des Salsa-Projektes: http://www.salsa-autonomik.de/
[25] Webseite des Cognilog-Projektes: http://www.cognilog.uni-hannover.de/

santen Relationen zwischen Hard- und Softwarekomponenten können nicht abgebildet werden. Vielmehr müssen die verschiedenen Sichten der Produktentwicklung berücksichtigt werden.

- Anforderungen: Zur Strukturierung der komplexen Aufgabe werden die Anforderungen formalisiert und in Unteranforderungen aufgegliedert. Außerdem wird beschrieben, wie diese Anforderungen miteinander in Abhängigkeit zueinander stehen. In Kapitel 4 wurden insbesondere matrizenbasierte Ansätze wie House-of-Quality in diesem Zusammenhang erwähnt. In ReACT werden insgesamt fünf Hauptanforderungen behandelt, die anhand eines UML-Diagramms geordnet und in Tabellen detailliert werden. Ein Ausschnitt der Anforderungen für die Navigation mit ihren Abhängigkeiten zum Android-Client ist in Abbildung 59 dargestellt.

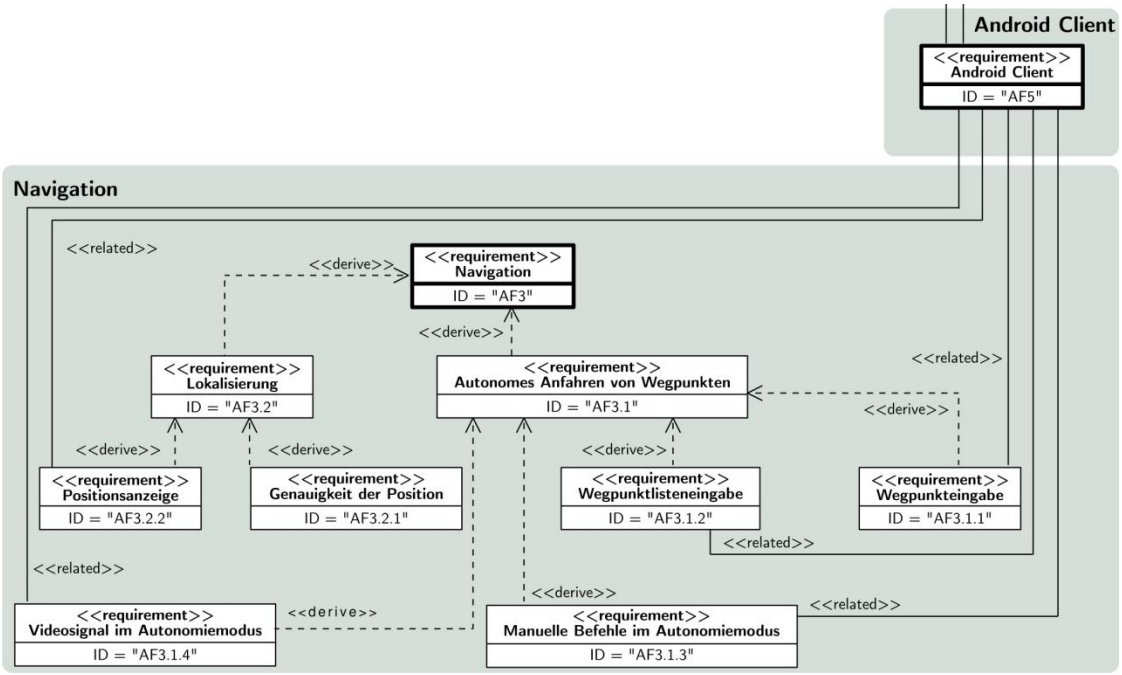


Abbildung 59: Struktur der Anforderungen der Navigation
Quelle: Abschlussbericht der Projektgruppe ReACT (ReACT-Gruppe 2011)

- Hardware-Aufbau: Das autonome Fahrzeug besteht aus einem Basisfahrzeug, dass mit Aktuatoren, Sensoren und Steuerungsgeräten ausgestattet ist. Der Aufbau dieser Hardware-Komponenten muss in dem Produktmodell beschrieben werden.
- Kommunikation: Die Komponenten kommunizieren über verschiedene Protokolle miteinander. Diese Kommunikationsarchitektur muss als Ergänzung zur Hardware-Beschreibung im Produktmodell dargestellt werden.

- Player-Architektur: Das Robotik-Framework Player/Stage (Collett, MacDonald, und Gerkey 2005) wird in ReACT als Basis für die autonome Steuerung verwendet. Dieses Framework unterstützt eine Modularisierung der Aktoren, Sensoren und Logik. Die Beziehungen zwischen diesen Modulen müssen im Produktmodell abgebildet werden. Die Abbildung 60 zeigt beispielhaft einen Ausschnitt der sogenannten Player-Architektur in ReACT mit den abzubildenden Schnittstellenbeziehungen.

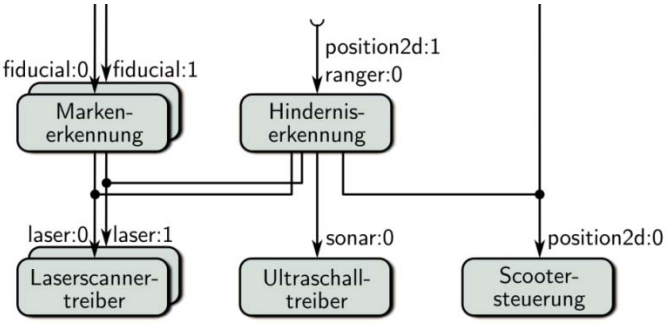


Abbildung 60: Ausschnitt der Player-Architektur
Quelle: Abschlussbericht der Projektgruppe ReACT (ReACT-Gruppe 2011)

- Client-Software: Auf dem Client-Gerät, einem Android-Handy, ist eine Anwendung installiert, die mit dem Fahrzeug kommuniziert und auch Kameradaten anzeigt. Diese Anwendung ist in Java geschrieben und ebenfalls Teil des Produktmodells.

Die dargestellten Sichten zeigen eine Vielzahl an Elementen und Beziehungen im hohen Detailgrad. Für die Modellierung des Produktmodells im Rahmen der Komplexitätsanalyse werden die wichtigsten Elemente und Beziehungen in einem neu erstellten Metamodell abgebildet. In diesem Modell werden sowohl Hardware- als auch Softwarekomponenten erfasst, sowie die Anforderungen die an diese gestellt werden. Zur Vereinfachung werden Kommunikationsbeziehungen, ob über Netzwerkprotokolle oder über Player-Interfaces, durch eine Relation abgebildet. Das mit dem ECore-Editor (siehe Abschnitt 6.2) erstellte Metamodell ist in Abbildung 61 dargestellt. Es ist geeignet die genannten Sichten auf die Produktentwicklung von ReACT in einem Modell zu vereinigen und als Basis für das Produktmodell der Komplexitätsanalyse zu dienen.

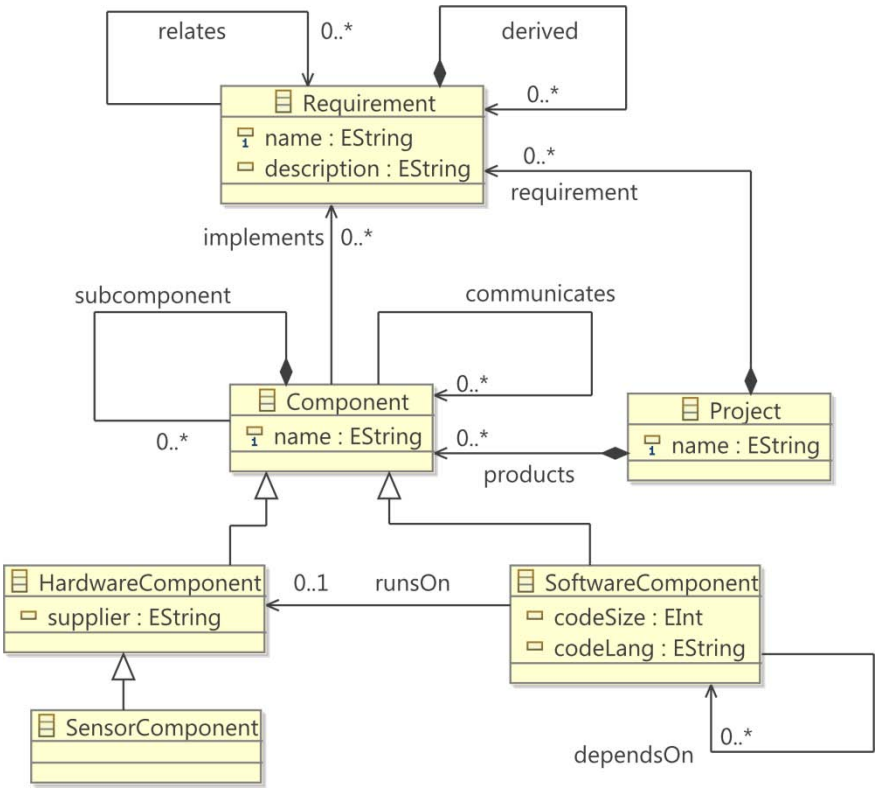


Abbildung 61: Metamodell des Fallbeispiels ReACT

Nach der Erstellung des Metamodells wird das Produktmodell auf Basis des Abschlussberichts und der Projektdateien erstellt. Eine automatische Transformation ist aufgrund der teilweise informellen Beschreibung der Komponenten nicht möglich. Allerdings können auch textuell beschriebene Informationen in das Produktmodell einfließen. Insgesamt zeigt sich bei der Modellierung der Vorteil des hier vorgestellten Ansatzes, dass kein festes Metamodell vorausgesetzt wird. Die Modellierung kann an die jeweiligen Informationsquellen angepasst werden und ist mit Hilfe der EMF-Werkzeuge bequem durchführbar.

#### 7.2.2.2 Definition der Gewichtung der Artefakte

Da die Komponenten des Produktmodells unterschiedlicher Art sind und in einigen Fällen von Drittherstellern bezogen, ist die Definition der Artefakte nicht so eindeutig wie in dem vorherigen Fallbeispiel. Die Entwickler wurden daher in Interviews gebeten Komponenten zu identifizieren, die mit einem Aufwand verbunden waren. Obwohl einige Komponenten mit unterschiedlichen Bezeichnungen benannt wurden, sind die als Artefakte zu betrachtenden Produktkomponenten eindeutig identifizierbar. Als Artefaktklasse wird die allgemeine Oberklasse `Component` gewählt. Die Instanzen dieser Klasse, die keine Artefakte sind, werden im Gewichtungseditor mit einer Komplexität von 0 markiert.

Für die Gewichtung der Artefakte wurden die Entwickler später darum gebeten, in einer Tabelle mit den identifizierten Artefakten zwei Werte einzutragen. Der erste Wert war der geschätzte geleistete Aufwand in Stunden insgesamt für eine Komponente. Der zweite Wert war eine Bewertung der Vertrautheit mit der jeweiligen Komponente auf einer Skala von 0 („Keine Ahnung, was das ist oder tut.“) bis 10 („Ich kenne die Komponente in- und auswendig!“). Durch diese Kennzahlen kann eine Bewertung der Komponenten auf Basis der gewichteten Aussagen durch folgende Formel ermittelt werden.

$$\text{Aggregierter Aufwandswert} = \frac{\sum_{\text{Schätzungen}} \text{Schätzwert} * \text{Vertrauen}}{\sum_{\text{Schätzungen}} \text{Vertrauen}}$$

Es sind fünf ausgefüllte Schätzbogen in die Gewichtung eingegangen. Als Einheit für die Schätzung wurden Arbeitsstunden verwendet. Die Artefakte und der aggregierte Aufwandswert sind in Tabelle 22 dargestellt. Zu beachten ist, dass die Schätzungen sich jeweils nur auf die erfassten Komponenten beziehen. Nicht erfasst wurden Aufwände bei der Entwicklung des ReACT-Prototyps durch Projekttreffen, kleinere Teile am Fahrzeug und andere Tätigkeiten. Der tatsächliche Gesamtaufwand betrug also wesentlich mehr als die summierten 782 Arbeitsstunden. Wenn der Gesamtaufwand des Projektes zum Beispiel durch eine Projektkostenrechnung verfügbar wäre, könnte dieser Aufwand über die berechneten Anteile zugeordnet werden.

Tabelle 22: Gewichtungen auf Basis der Entwicklerbefragung

| **Komponente** | **Systembereich** | **Aufwand** | **Anteil** |
|---|---|---|---|
| Hinderniserkennung | Player | 98,636 | 0,126 |
| Navigation | Player | 96,666 | 0,124 |
| Treiber Ultraschall | Player | 53,750 | 0,069 |
| Treiber Laserscanner | Player | 35,454 | 0,045 |
| Scooter-Steuerung | Player | 94,545 | 0,121 |
| MainApp | Client | 119,473 | 0,153 |
| MethodPool | Client | 10,000 | 0,013 |
| PollingThread | Client | 15,882 | 0,020 |
| ShowMap | Client | 34,722 | 0,044 |
| ValueBean | Client | 5,000 | 0,006 |
| Odometrie | Player | 72,500 | 0,093 |
| Triangulation | Player | 78,333 | 0,100 |
| ClientWrapper | ClientWrapper | 67,307 | 0,086 |
| **Ergebnis** | | **782,2723** | **1,000** |

Die absoluten Werte der aggregierten Aufwandsschätzung wurden mit Hilfe des Gewichtungseditors in das Referenzproduktmodell eingetragen. Die Einheit wurde als „Stunden (h)“ angegeben. Die Abbildung 47 im Abschnitt 6.3.2 zeigt die Eingabe im Gewichtungseditor. Bei den grau hinterlegten Zeilen handelt es sich um Zukaufteile und andere Komponenten, die nicht in die Bewertung eingegangen sind.

#### 7.2.2.3 Regression

Wie im vorherigen Beispiel wird der Assistent zur Erstellung eines Komplexitätsmaßes auf die erzeugte `.weighting`-Datei angewendet. Im Gegensatz zum Permeter-Fallbeispiel ist diese Datei aufgrund der manuellen Erstellung des Produktmodells wesentlicher kleiner. Die Erzeugung und die Anwendung der `Regressor`en hat daher keine nennenswerte Berechnungszeit benötigt.

Es wurden auf Basis des Produktmodells 152 Attribute erzeugt. Die Auswahl der Attribute nach dem im Abschnitt 5.6.1 dargestellten Vorgehen wird in Tabelle 23 dargestellt. Als Ergebnis der Analyse wurde ein Komplexitätsmodell mit 3 Attributen erzeugt und als Komplexitätsmaß abgespeichert.

Tabelle 23: Auswahl Attribute im Validierungsszenario ReACT

| Selektionsstufe | Siehe Abschnitt | Attributbewertung | Verbleibend |
|---|---|---|---|
| Vorhandene Attribute | 5.6.1.4 | Erzeugung Attributmuster | 152 Attribute |
| Sinnvolle Attribute | 5.6.1.5 | Vollständigkeit, Streuung | 12 Attribute |
| Signifikante Attribute | 5.6.1.6 | CFS-Attributfilter | 3 Attribute |
| Erklärende Attribute | 5.6.1.7 | Akaike-Kriterium | 3 Attribute |

### 7.2.3 Bewertung des Komplexitätsmaßes

Der Wert des Bestimmtheitsmaßes $R^2$ für das dem Komplexitätsmaß zugrundeliegende Regressionsmodell beträgt 0,802. Dieser Wert ist wesentlich geringer als im Fallbeispiel Permeter. Dies ist vor allem darauf zurückzuführen, dass weniger Daten zur Verfügung standen und ein Produktmodell betrachtet wurde, dass aus verschiedenen Partialmodellen zusammengesetzt wurde. Der Konvergenzwert (siehe Abschnitt 5.6.1.8) beträgt 18,87. Ein vergleichbares Produkt sollte also mindestens 19 Komponenten besitzen, damit der Messwert nicht zu stark abweicht. Somit lässt sich auf Basis dieses Modell bei der Anwendung auf eine größere Menge Artefakte ein Vergleich mit der Entwicklung des ReACT-Prototypen ziehen.

Interessant ist außerdem ein Blick auf die Attribute des Komplexitätsmaßes, welches durch die folgende Formel dargestellt ist. Während das Attribut `NumericAttribute(codeSize)` die Quellcodegröße (gemessen in Zeichen) beschreibt, werden durch weitere Attribute Abhängigkeiten zu anderen Komponenten abgebildet. `Usage` ist die Anzahl beliebiger eingehender Relationen eines Artefaktes und `TreeCrossRef(communicates)` die Anzahl ausgehender Relationen des Typs `communicates` zu anderen Komponenten.

$$\begin{aligned}\text{Complexity} = \; & 0.0008 * \text{NumericAttribute(codeSize)} + \\ & 9.5125 * \text{Usage} + \\ & 8.8303 * \text{TreeCrossRef(communicates)}\end{aligned}$$

Für ein genaueres Maß wäre allerdings die Untersuchung von mehreren konsekutiven Entwicklungen interessant. Es ist möglich, dass bei ReACT externe Faktoren eine wesentliche Rolle gespielt haben, die bei alleiniger Betrachtung der ReACT-Daten nicht zu erkennen sind. Die Abbildung 62 stellt die in der Entwicklerbefragung ermittelten Komplexitätswerte den vorhergesagten Komplexitätswerten gegenüber. Im Vergleich zu den Ergebnissen aus dem Permeter-Fallbeispiel ist zu erkennen, dass die Messung eines einzelnen Artefakts zu ungenau ist.

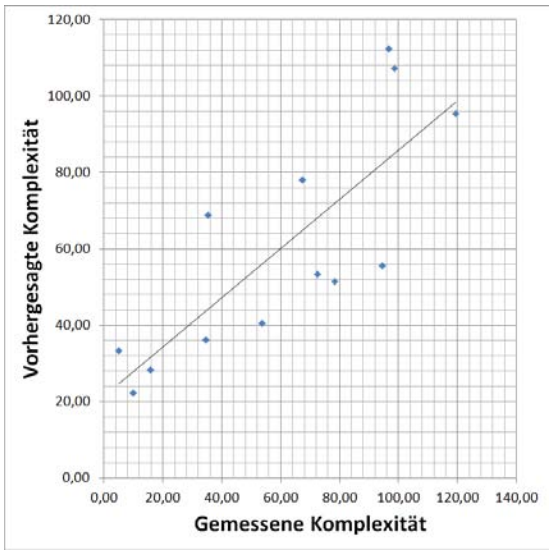


Abbildung 62: Plot des Regressionsmodells im Validierungsszenario ReACT

## 7.3 Fallbeispiel CATIA

Die anderen Beispiele beziehen sich auf Disziplinen, die insgesamt oder wesentlich Software einsetzen. Um die allgemeine Anwendbarkeit des Ansatzes zu validieren, wird in diesem Fallbeispiel ein Produktmodell untersucht, das keinen Quellcode im Sinne eines Softwareprogramms beinhaltet. Als wesentlicher Aspekt der Produktentwicklung soll die geometrische Konstruktion adressiert werden. Die Herausforderung in diesem Fallbeispiel sind die starken strukturellen Unterschiede zu den Softwarepartialmodellen: Es liegen keine Module oder Klassen vor; die Modelle bestehen aus einer Menge von geometrischen Definitionen.

### 7.3.1 Hintergrund

In diesem Fallbeispiel wird untersucht, welche Eigenschaften die Komplexität von geometrischen Modellierungen beeinflussen und wie ein Komplexitätsmaß definiert werden kann, um Modelle miteinander vergleichen zu können. Da es alternative Ansätze zur Darstellung geometrischer Produktmodelle gibt, werden zwei unterschiedliche Darstellungsformen für die Analyse parallel berücksichtigt. Diese Repräsentationen werden im Folgenden beschrieben:

- Definition eines Körpers durch ein parametrisches CAD-Modell in Form einer CATIA-Datei
- Beschreibung der Hüllgeometrie als Boundary Representation (B-Rep) in Form einer STEP-Datei

Die Konstruktion parametrischer CAD-Modelle erfolgt durch die Erstellung von prozeduralen Elementen, die als Features bezeichnet werden. Jedes Feature stellt eine ergänzende Information des geometrischen Modells dar und wird durch seine Parameter wie

Referenzen zu anderen Elementen und zusätzliche Werte definiert. Zum Beispiel kann ein einfacher rechteckiger Kasten durch das Feature `Block` erstellt werden, dem ein zweidimensionales Profil eines Rechtecks (Skizze) und eine Höhe als Parameter übergeben wird. Durch die schrittweise Anwendung von Features werden schließlich auch detaillierte Modelle erstellt. Die Abarbeitung der Features wird im sogenannten Feature-Baum definiert. Die Abbildung 63 zeigt ein einfaches Teil, welches durch drei Features definiert wurde. Zunächst wurde ein einfacher Block definiert und aus diesem durch das Feature `Tasche` eine Runde Form abgezogen. Das Feature `Kantenverrundung` beschreibt die Abrundung der Kanten auf einer Fläche des Blocks. Auf der linken Seite der Abbildung ist der formale Feature-Baum zu sehen.

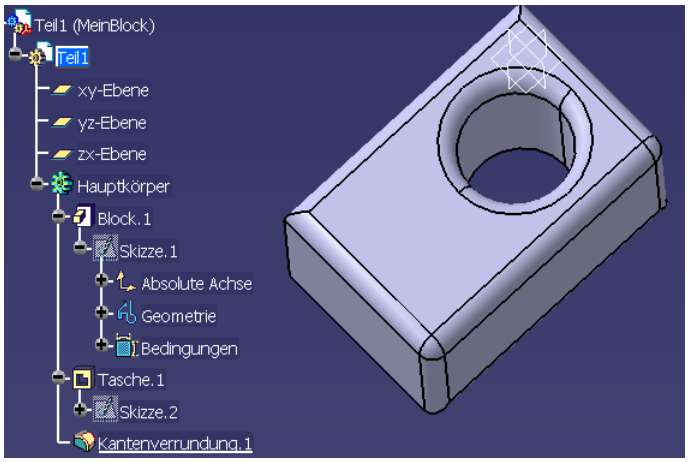


Abbildung 63: Feature-Baum in CATIA mit drei Features

Die parametrischen CAD-Modelle wurden mit Hilfe der CAD-Anwendung CATIA[26] erstellt, welche insbesondere im Automobilbereich allgegenwärtig ist. Diese Software umfasst eine Reihe miteinander integrierter Werkzeuge zur Produktentwicklung, die als Workbenches bezeichnet werden. Die untersuchten Modelle wurden durch die Workbench Part Design erstellt, die der geometrischen Definition von Bauteilen dient. Die Modelle liegen als einzelne Dateien im proprietären Format CATPart vor. Im Folgenden wird vereinfacht von CATIA-Modellen gesprochen.

Die zweite betrachtete Darstellung geometrischer Produktmodelle ist die Boundary Representation (B-Rep), die Objekte über angrenzende Oberflächen beschreibt. Das im Automobilbereich eingesetzte Austauschformat STEP AP 214 implementiert diese Darstellungsform und kann durch CATIA Part Design exportiert werden. Jedes Modell in diesem Beispiel liegt als CATIA-Modell sowie als STEP-Modell vor.

Im Gegensatz zu den anderen Fallbeispielen werden zur Kalibrierung 27 getrennte Modelle verwendet, die jeweils einen einzelnen Komplexitätswert erhalten. Die Modelle

[26] Produktseite des Herstellers Dassault Systèmes zu CATIA: http://www.3ds.com/products/catia

wurden von Maschinenbau-Studierenden im Rahmen einer Übung zur parametrischen Konstruktion in CATIA erzeugt und sind in ihrem Umfang leicht unterschiedlich. Dies ist auf die jeweiligen Vorkenntnisse der Studierenden zurückzuführen. Die Abbildung 64 zeigt sechs Beispiele, die unter anderem einen Klebebandroller, eine Zentrierhalterung und einen Tankstutzen beinhalten.

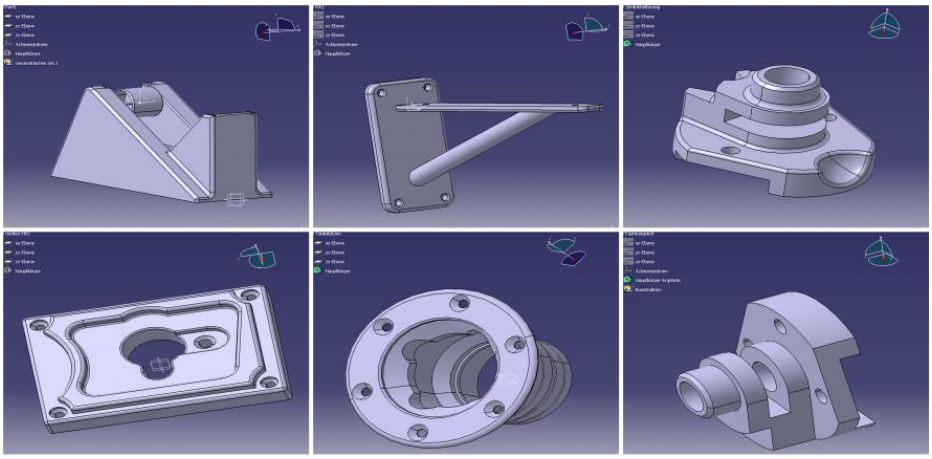

Abbildung 64: Beispiele für CATIA-Modelle

### 7.3.2 Anwendung des Vorgehensmodells

Die Validierung orientiert sich an dem in Kapitel 5 vorgestellten Vorgehensmodell für die Erstellung eines Komplexitätsmaßes. Der Abschnitt zur Modellierung stellt in diesem Fallbeispiel den wichtigsten Aspekt der Untersuchung dar.

#### 7.3.2.1 Modellierung

Um beide Darstellungsformen zu berücksichtigen werden zwei Metamodelle benötigt. Für das STEP-AP214-Modell kann auf die Definition der Klassen von AP 214 im STEP-Metamodellformat EXPRESS zurückgegriffen werden. Für die Untersuchung sind insbesondere die Flächenelemente relevant. Details wie einzelne Punkte oder Linien, die im B-Rep-Modell zur Definition der Flächen verwendet werden, werden in der Analyse nicht berücksichtigt. Die Abbildung 65 zeigt ein Klassendiagramm der wichtigsten geometrischen Klassen des STEP-Metamodells, die verschiedene Flächentypen darstellen. Die Klasse `ShapeRepresentation` steht für das Volumenmodell, das durch eine Menge von `RepresentationItems` definiert wird. In der Regel handelt es sich hierbei um eine Instanz der Klasse `ClosedShell`, die einen geschlossen Körper repräsentiert.

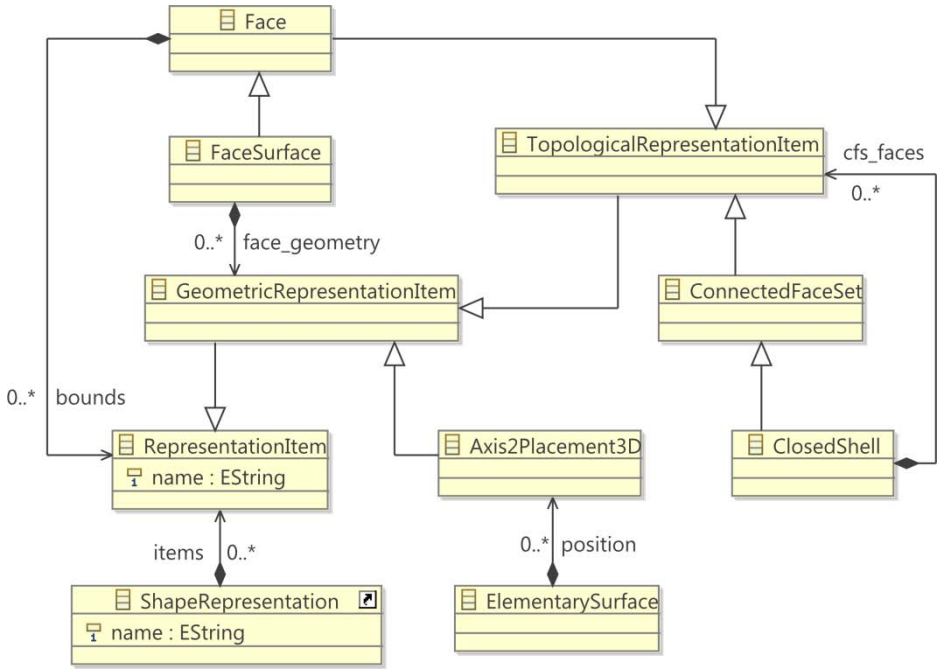


Abbildung 65: Ausschnitt aus dem STEP-AP214-Metamodell

Neben den dargestellten Klassen werden einige Klassen erfasst, die Metadaten wie die Verwendung in einer Stückliste beschreiben. Da die CATIA-Modelle jedoch nicht aus dem Kontext eines Produktdatenmanagementsystems exportiert wurden, sind diese Daten in der Regel nicht definiert.

Für das zweite Metamodell kann kein bestehendes Metamodell als Vorlage genutzt werden. Das Modell ist entsprechend direkt aus dem Programm und der Dokumentation von CATIA abgeleitet. Die zentrale Klasse des Modells ist das `Feature`. Von dieser abstrakten Klasse werden verschiedene Arten von Features abgeleitet. Die Abbildung 66 gibt einen Überblick über die modellierten Feature-Typen. Die Abbildung zeigt außerdem die Anbindung des STEP-Modells durch die Definition des Features `Körper` als Unterklasse von `RepresentationItem`.

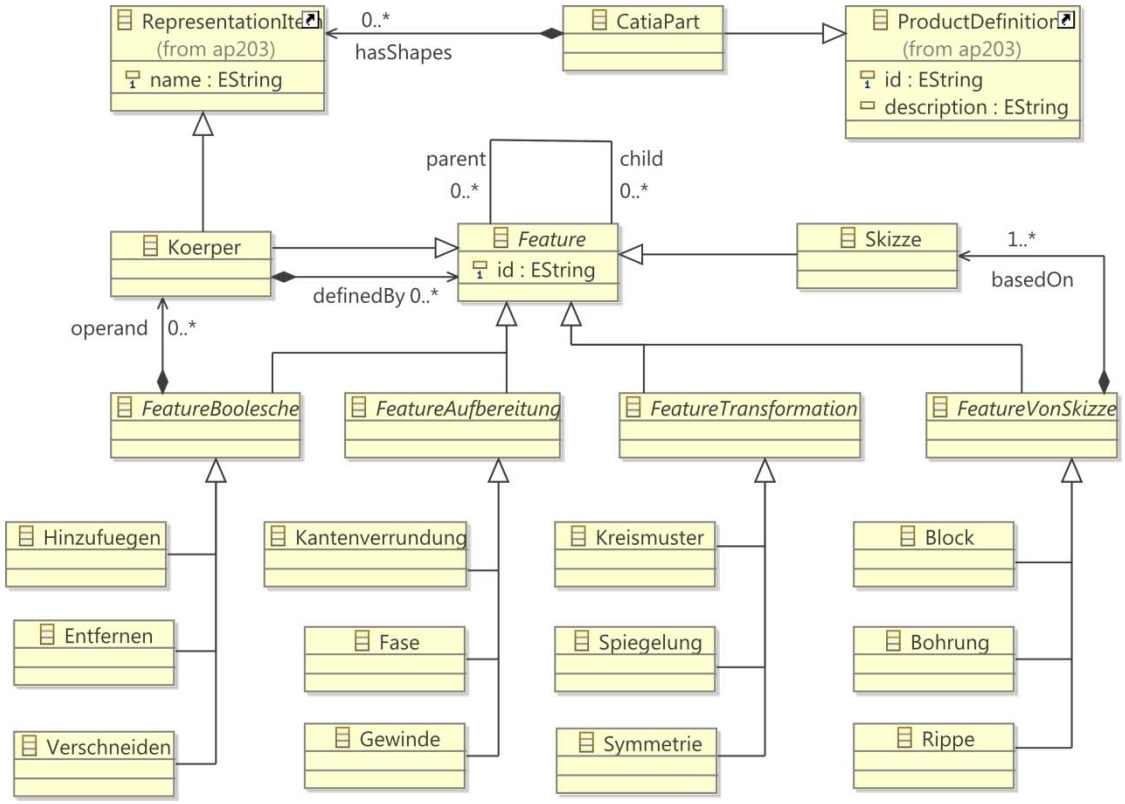


Abbildung 66: Ausschnitt aus dem CATIA-Metamodell

#### 7.3.2.2 Transformation

Für die Transformation der STEP-Daten ist eine automatische Erfassung notwendig, da bereits bei relativ einfachen Teilen die hohe Anzahl der Oberflächen einen sehr hohen manuellen Aufwand bedeuten würde. Aus diesem Grund ist für den Prototyp ein anpassbarer STEP-File-Parser geschrieben worden. In einem ersten Parser-Durchlauf werden Einträge in den nativen STEP-Files auf Klassen in einem gegebenen Metamodell abgebildet und entsprechende Instanzen erzeugt. In einem zweiten Durchlauf werden schließlich die entsprechenden Attribute und Verknüpfungen ergänzt.

Die Erfassung der Feature-Bäume wird durch einen Editor vorgenommen. Wie im Fallbeispiel ReACT kann dieser aus dem Metamodell durch das EMF-Framework automatisch erzeugt werden. Es wurden außerdem einige Symbole aus der Part Design Workbench übernommen. Da die zu untersuchenden Teile jeweils ungefähr zwischen 10 und 30 Features beinhalten, ist dieser Aufwand noch vertretbar. Für eine automatisierte Lösung über den Prototyp hinaus wäre eine Abfrage in der CATIA-eigenen Skriptsprache CATSkript denkbar. Die Abbildung 67 zeigt, dass die Erfassung des Feature-Baumes zu der Darstellung in CATIA sehr ähnlich ist.

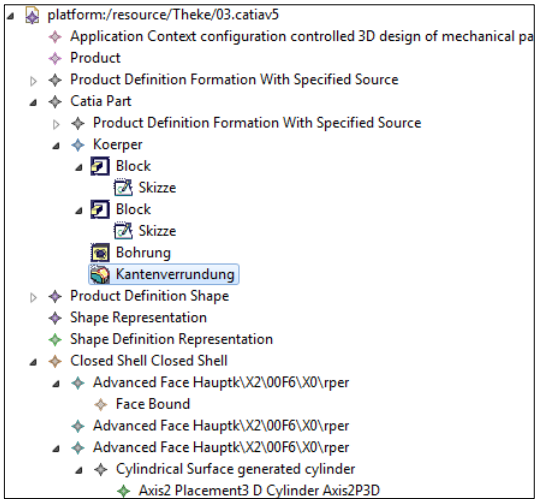


Abbildung 67: Erfassung des Feature-Baums durch den Editor

Nach der Erfassung der beiden Partialmodelle müssen diese miteinander integriert werden. Wie im vorherigen Abschnitt beschrieben, werden dazu in CATIA definierte Körper als entsprechende Elemente eines STEP-Modells definiert. In der obigen Abbildung ist diese Integration durchgeführt worden; Das STEP `Product` wird sowohl durch die B-Rep-Elemente als auch durch den Feature-Baum beschrieben. Der gesamte Prozess der Erfassung und Zusammenführung der beiden Partialmodelle wird durch die Abbildung 68 zusammengefasst.

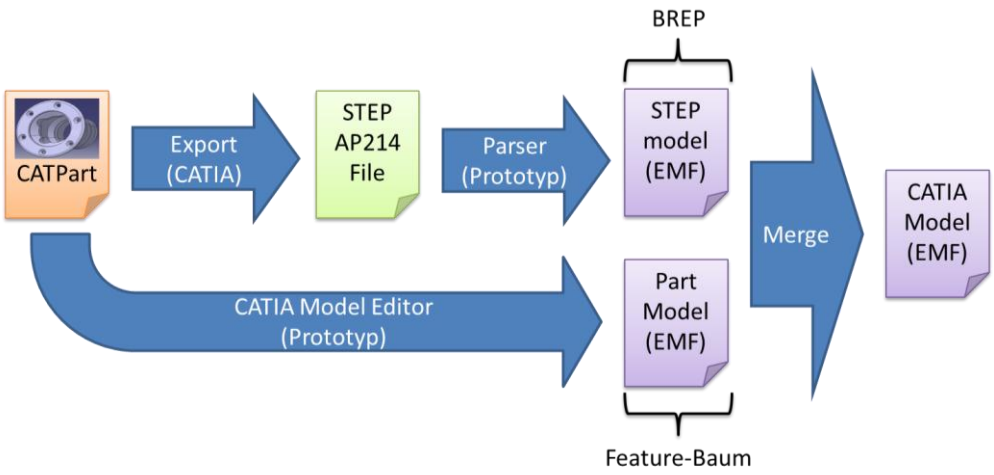


Abbildung 68: Erfassung und Transformation der CATIA-Modelle

#### 7.3.2.3 Definition und Gewichtung der Artefakte

Jedes auf Basis einer CATPart-Datei erzeugte CATIA-Modell stellt in diesem Fallbeispiel ein Artefakt dar. Die Definition dieser Modelle als Artefakte erfolgt über das Wurzelelement des Modells in Form einer Instanz der Klasse `CatiaPart`. Die Modelle

mit jeweils einem Artefakt werden zu Beginn der Komplexitätsanalyse zu einem Referenzproduktmodell vereinigt.

Die Bedienung von CATIA erfordert eine lange Einarbeitungszeit und die Bearbeitungszeit ist hiervon stark abhängig. Durch die unterschiedlichen Vorkenntnisse der Studierenden sind daher die Aufwände innerhalb der Übung nicht sinnvoll miteinander vergleichbar. Da für die Kalibrierung eines Komplexitätsmaßes von dem gleichen oder vergleichbaren Entwicklungssystem ausgegangen wird (siehe hierzu Abschnitt 5.2), war eine externe Bewertung notwendig. Diese wurde durch eine Probandengruppe von fünf Mitarbeitern eines Automobilherstellers durchgeführt, die mehrjährige Erfahrung in der Arbeit mit CATIA besitzen. Die Probanden in der Untersuchung schätzten ab, wie lange sie für die Erstellung der CATPart-Dateien in Minuten benötigen würden. Da die Schätzung eines Probanden eine geringe Korrelation zu den Schätzungen der anderen aufwies, wurden dieser in der Auswertung nicht berücksichtigt.

Für jede berücksichtigte Schätzreihe wurden die relativen Schätzwerte berechnet, indem die absoluten Schätzwerte durch die Summe der Schätzreihe geteilt wurden. Für die relativen Werte wurde dann der Durchschnitt über alle Probanden gebildet. Sei $c_{i,j}$ die Schätzung des Probanden $i$ für das Artefakt $j$ mit $n$ Probanden und $m$ Artefakten, so ergibt sich der aggregierte relative Wert für das Artefakt $a$ also wie folgt:

$$\text{Aggregierter relativer Wert } crel_a = \frac{\sum_{i=0}^{n}\left(\frac{c_{i,a}}{\sum_{j=0}^{m} c_{i,j}}\right)}{n}$$

Hierdurch werden alle Probanden trotz unterschiedlicher durchschnittlicher Wertung gleich bewertet. Dieser relative Wert wird wieder auf einen absoluten Wert abgebildet, indem er mit der Durchschnitt der Summe aller Schätzreihen multipliziert wird.

$$\text{Aggregierter absoluter Wert } cabs_a = \frac{crel_a * \sum_{i=0}^{n}\left(\sum_{j=0}^{m} c_{i,j}\right)}{n}$$

Das Komplexitätsmaß kann somit für CATPart-Modelle eine absolute Schätzung vornehmen. In der Tabelle 24 sind die aggregierten Werte für alle Artefakte aufgeführt. Insbesondere sind die Unterschiede in der Komplexität von ungefähr 13 bis 71 Minuten zu erkennen. Der Detailgrad der Modelle hing also wesentlich von zu anfangs erwähnten unterschiedlichen Vorkenntnissen der Ersteller ab.

Tabelle 24: Aggregierte absolute Schätzwerte im Fallbeispiel CATIA

| Nr | CATPart | Minuten | Nr | CATPart | Minuten |
|---|---|---|---|---|---|
| 1 | Rohr mit Flansch | 19,545 | 15 | Führungsteil | 20,979 |
| 2 | Globushalter | 17,715 | 16 | Kreisplatte | 25,777 |
| 3 | USB-Stick | 37,731 | 17 | Rohr mit Kranz | 27,982 |
| 4 | Frästeil | 16,378 | 18 | Platte mit Halterungen | 41,637 |
| 5 | Rohrhalter | 37,332 | 19 | Doppelscheibe | 26,349 |
| 6 | Arbeitstisch | 34,268 | 20 | Bierkasten | 35,944 |
| 7 | Kolbenkopf | 21,600 | 21 | Innensechskantschlüssel | 14,645 |
| 8 | Drehkolbenwelle | 21,225 | 22 | Taschenrechner | 32,708 |
| 9 | Klebefilmroller | 24,989 | 23 | Armbanduhr | 48,393 |
| 10 | Winkelstück | 13,631 | 24 | Lampenschirm | 32,683 |
| 11 | Zentrierhalterung | 42,304 | 25 | Halterung | 71,700 |
| 12 | Spielwürfel | 35,157 | 26 | Pleuel | 70,214 |
| 13 | Halterungsplatte | 18,726 | 27 | Lautsprecher | 58,510 |
| 14 | Tankstutzen | 21,129 | | | |

Diese Schätzwerte wurden durch den Gewichtungseditor in das Referenzproduktmodell eingetragen (vgl. Abbildung 47). Es wurde als Bezeichnung für die Komplexitätseinheit „Arbeitsminuten“ gewählt.

#### 7.3.2.4 Regression

Die erzeugte Gewichtungsdatei wurde ebenfalls mit Hilfe des Assistenten zur Erzeugung des Komplexitätsmaßes ausgewertet. Die Menge der Daten war wesentlich geringer als im Fallbeispiel Permeter aber höher als im Fallbeispiel ReACT. Die Erstellung der Regressoren und die Bewertung des integrierten Modells durch die Regressoren wurden im Assistenten entsprechend in wenigen Sekunden durchgeführt.

Da es sich bei den Artefakten in diesem Szenario um voneinander unabhängige Modelle mit jeweils einem Artefakt handelte, konnten diese für die Validierung des Komplexitätsmaßes in eine Kalibrierungsmenge von 18 Artefakten und eine Testmenge von 9 Artefakten aufgeteilt werden. Die Aufteilung zwischen diesen beiden Gruppen erfolgte hierbei nach Zufall.

Es wurden auf Basis des Produktmodells 405 Attribute erzeugt. Die Auswahl der Attribute nach dem im Abschnitt 5.6.1 dargestellten Vorgehen wird in Tabelle 23 dargestellt. Als Ergebnis der Analyse wurde ein Komplexitätsmaß mit 3 Attributen erzeugt und als Komplexitätsmaß abgespeichert.

Tabelle 25: Auswahl Attribute im Validierungsszenario CATIA

| **Selektionsstufe** | **Siehe Abschnitt** | **Attributbewertung** | **Verbleibend** |
|---|---|---|---|
| Vorhandene Attribute | 5.6.1.4 | Erzeugung Attributmuster | 405 Attribute |
| Sinnvolle Attribute | 5.6.1.5 | Vollständigkeit, Streuung | 70 Attribute |
| Signifikante Attribute | 5.6.1.6 | CFS-Attributfilter | 7 Attribute |
| Erklärende Attribute | 5.6.1.7 | Akaike-Kriterium | 3 Attribute |

### 7.3.3 Bewertung des Komplexitätsmaßes

Die Regression auf der Kalibrierungsmenge erzeugte ein Komplexitätsmodell mit 3 Attributen. Durch die Anwendung dieses Modells auf die Testmenge und der Vergleich mit den tatsächlichen Werten wurde ein Bestimmtheitsmaß von $R^2 = 0{,}91$ erzielt. Der Konvergenzwert bei einem akzeptablen relativen Fehler von $x_{rel} = 0{,}1$ des Komplexitätsmaßes beträgt 13,40. Ein CATIA-Produkt sollte also durch mindestens 14 CATPart-Modelle definiert werden, damit eine Genauigkeit von 10% erzielt werden kann. Da diese Anzahl für solche Produkte realistisch ist, ist die Eignung des Komplexitätsmaßes zur Vermessung von CATIA-Produkten gegeben. Durch eine Einbindung in Produktdatenmanagementsysteme wäre eine wesentliche Unterstützung für die Bewertung von Produktentwicklungsprozessen realisierbar. Insbesondere kann eine auf die Komplexitätsmessung basierende Earned-Complexity-Analyse (siehe Abschnitt 5.8.3) bessere Einblicke in die Dynamik des Entwicklungsprozesses geben.

Eine interessante Erkenntnis aus der Analyse des Komplexitätsmodells ist, dass Attribute, die sich auf die BREP-Darstellung beziehen, nicht in das Komplexitätsmaß eingegangen sind. Es kann also davon ausgegangen werden, dass die Eigenschaften dieses Partialmodells keinen wesentlichen Einfluss auf die Komplexität der Produktentwicklung besitzen. Als wesentliche Einflussfaktoren werden in der Analyse die Anzahl bestimmter Feature-Typen identifiziert. Hierbei besitzen auf Skizzen basierende Features eine höhere Auswirkung auf die Komplexität als auf Transformationen basierende Features (vgl. Abbildung 66).

Auch wenn die Eignung über eine realistische Anzahl von Artefakten gegeben ist, zeigt jedoch insbesondere das Bestimmtheitsmaß, dass einige Aspekte der Komplexität nicht erfasst wurden. Eine naheliegende Erklärung ist, dass einige komplexitätsrelevante De-

tails bei der manuellen Erfassung des Feature-Baumes nicht berücksichtigt werden konnten. Die Implementierung einer automatischen Transformation, die in der CATIA Part Workbench integriert ist, ist daher für eine Anwendung in der Praxis zu empfehlen. Insbesondere könnten so die Details der verwendeten Skizzen sowie weitere Parameter der Features berücksichtigt werden. Die Abbildung 69 stellt die auf Basis des Komplexitätsmaßes hervorgesagten Komplexitätswerte der Testmenge und die tatsächlichen Werte gegenüber.

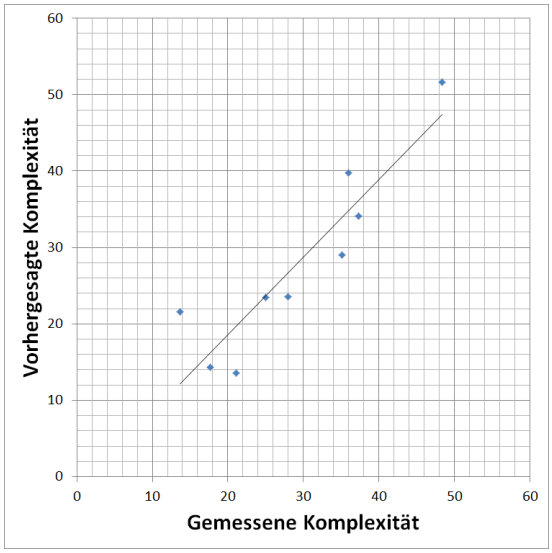


Abbildung 69: Plot der Werte der Testmenge im Validierungsszenario CATIA

## 7.4 Überprüfung der Anforderungen

In diesem Abschnitt werden die Ergebnisse der Validierung zusammenfassend diskutiert. Welche Vorteile und Herausforderungen besaß der im Kapitel 6 konzeptionell herausgearbeitete Ansatz in der praktischen Anwendung? Anhand dieser Ergebnisse sollen die Anforderungen aus Abschnitt 2.2.1.2 geprüft werden, die an dem in dieser Arbeit vorgestellten Ansatz gestellt wurden.

Die Fallbeispiele behandeln den Ansatz mit sehr unterschiedlichem Schwerpunkt. Insbesondere konzentrieren sich das erste und dritte Fallbeispiel auf eine bestimmte Domäne, während das zweite Fallbeispiel einen domänenübergreifenden Charakter besitzt. Die besten Ergebnisse wurden in dem Permeter-Fallbeispiel erzielt. Die liegt vor allem daran, dass die Vorrausetzungen für eine Komplexitätsmessung bei Quellcode-Modellen am ehesten gegeben sind: Die Beziehungen zwischen den Elementen sind formell spezifiziert und die einzelnen Artefakte in Form von Quellcodeeinheiten gut abgrenzbar. Viele der in der Literatur existierenden Ansätze stammen entsprechend aus diesem Gebiet. In den beiden anderen Fallbeispielen erfolgt eine stärkere Abgrenzung vom Stand der Technik durch die Anwendung auf andere Domänen. Insgesamt kann ein

sehr gutes Bild von den Eigenschaften des Ansatzes gewonnen werden, so dass eine Überprüfung der geforderten Anforderungen möglich ist.

### 7.4.1 Einzelbewertung

Im Folgenden wird die Einzelüberprüfung der Anforderungen vorgenommen. Die detaillierte Beschreibung der Anforderungen ist der Tabelle 2 auf Seite 41 zu entnehmen.

#### 7.4.1.1 Quantifizierung als Schwierigkeit der Erzeugung

In den Fallbeispielen werden die Komplexitätsmaße anhand von Kennzahlen des Aufwandes in der Referenzproduktentwicklung kalibriert. Bei Permeter sind dies Commits im Versionssystem, bei ReACT die gewichteten Schätzungen der Entwickler in Arbeitsstunden und bei CATIA die nicht gewichteten Schätzungen anderer Entwickler in Arbeitsminuten. Die resultierenden Komplexitätsmaße sind somit Quantifizierungen auf Basis des Verständnisses von Komplexität als Schwierigkeit der Erzeugung.

#### 7.4.1.2 Bewertung der Semantik von Relationen

Die Semantik der Relationen der Artefakte wird durch entsprechende Attribute abgebildet. Im Rahmen der Regression wird eine Bewertung dieser Attribute im Sinne des Beitrages zur Komplexität durchgeführt. Dies ist für den Fall der domänenübergreifenden Relationen insbesondere im Fallbeispiel ReACT zu beobachten. Über diese Anforderung hinaus werden durch weitere Attributmuster auch strukturelle Eigenschaften wie Baumtiefe bewertet.

#### 7.4.1.3 Minimierung des Messaufwandes

Die erstellten Komplexitätsmaße können automatisch auf den in EMF-Modellen beschriebenen Produktmodellen ausgeführt werden. Die Einbindung der Erstellung und Ausführung eines Komplexitätsmaßes als Eclipse-Plug-In im Prototyp zeigt die gute Integrationsfähigkeit der Messung in die tägliche Entwicklungsarbeit. Allerdings erfordert die Erstellung des Komplexitätsmaßes mit Definition und Gewichtung der Artefakte einen nicht unwesentlichen einmaligen Aufwand. Dennoch kann gegenüber bestehenden manuellen Methoden eine wesentliche Reduktion des Messaufwandes erzielt werden.

#### 7.4.1.4 Domänenunabhängigkeit

Die Fallstudien beziehen sich auf verschiedene Domänen. Im Rahmen der Fallstudie ReACT werden außerdem die verschiedenen Sichten auf das Produkt in einheitlicher Form in die Komplexitätsanalyse eingebunden. Durch die Unterstützung des EMF-Frameworks ist die Anpassung an die jeweilige Domäne komfortabel möglich.

#### 7.4.1.5 Individuelles Komplexitätsmaß

In keinem der Fallbeispiele wird ein bestehendes Komplexitätsmaß verwendet. Lediglich bei Permeter werden bekannte Komplexitätsmetriken zum Leistungsvergleich herangezogenen. In jedem Fallbeispiel führte die Regression zu einem auf das jeweilige Produktmodell zugeschnittenen Komplexitätsmaß.

#### 7.4.1.6 Partialmodellübergreifende Beschreibung

Diese Anforderung wurde eingehend im ReACT-Fallbeispiel geprüft. Durch die flexible Erstellung eines Metamodells konnten die in Abschnitt 7.2.2.1 genannten Partialmodelle in einem gemeinsamen Modell integriert werden. Allerdings ist die Erstellung dieses integrierten Produktmodells auf Basis formeller und informeller Teilmodelle relativ aufwendig, da ein manuelles Durcharbeiten der Projektdokumente nicht zu vermeiden ist.

#### 7.4.1.7 Greifbare Komplexitätseinheit

Die Einheiten der Komplexitätsmaße der Fallbeispiele beziehen sich auf die jeweiligen zugrundeliegenden Referenzproduktmodelle. Die Maße sind relativ zu den Gesamtaufwänden zu lesen, die mit diesen Produktentwicklungen verbunden waren. Die Interpretation der Komplexitätsmaße erfordert somit eine Kenntnis dieser Aufwände, was jedoch in den Fallbeispielen nicht immer gegeben ist. So ist bei ReACT und Permeter ein genauer Gesamtaufwand nicht zu benennen. Entsprechend erfordert die Interpretation der Komplexitätsmaße ein grobes Verständnis des zugrundeliegenden Ansatzes zur Erstellung eines Komplexitätsmaßes. Die Anforderung ist somit als teilweise erfüllt zu betrachten.

### 7.4.2 Gesamtbewertung

Die folgende Tabelle fasst die Ergebnisse der Überprüfung der Anforderungen zusammen. Bis auf die Anforderung einer greifbaren Komplexitätseinheit können alle Anforderungen als sehr gut bis gut erfüllt betrachtet werden. Die Fallbeispiele zeigen, dass die prototypische Implementierung des in dieser Arbeit vorgestellten Ansatzes die Anforderungen an ein Komplexitätsmaß für die Produktivitätsbewertung erfüllt. Der Ansatz hat sich als vorteilhaft gegenüber bestehenden Ansätzen zur Leistungsmessung erwiesen.

Tabelle 26: Ergebnisse der Einzelprüfung der Anforderungen

| ID | Beschreibung | Adressierung | Erfüllung |
|---|---|---|---|
| KA1 | Quantifizierung als Schwierigkeit der Erzeugung | Kalibrierung des Maßes über Aufwandskennzahlen | Sehr gut |
| KA2 | Bewertung der Semantik von Relationen | Abbildung von Relationen durch entsprechende Attribute | Sehr gut |
| KA3 | Domänenunabhängigkeit | Freie Gestaltung des Metamodells | Sehr gut |
| PA1 | Minimierung des Messaufwandes | Automatisierbarkeit der Messdurchführung | Gut |
| PA2 | Individuelles Komplexitätsmaß | Freie Gestaltung des Metamodells | Sehr gut |
| PA3 | Partialmodellübergreifende Beschreibung | Einbindung der Partialmodelle in ein gemeinsames Metamodell | Gut |
| PA4 | Greifbare Komplexitätseinheit | Relativer Bezug zu den Referenzproduktentwicklungen | Mittel |

Für weitere Implementierungen auf Basis anderer Technologien wie Semantic-Web-Technologien sind aufgrund der gemachten Erfahrungen ähnliche Ergebnisse zu erwarten. Die Validierung zeigt, dass auf Basis einer flexiblen Modellierungssprache und einer Unterstützung durch Werkzeuge der Ansatz sinnvoll implementiert werden kann. Die Auswahl einer entsprechenden Basistechnologie sollte vor diesem Hintergrund sorgfältig durchgeführt werden.

Weitere Vorteile sind bezüglich des Aufwandes bei der Erstellung des Komplexitätsmaßes zu erwarten, wenn die Implementierung des Ansatzes in den nativen Entwicklungsumgebungen vorgenommen wird. Durch eine Integration in diese Umgebungen ist das Ziel einer Produktivitätsmessung ohne Beeinträchtigung der Produktentwickler erreichbar.

# 8 Abschluss und Diskussion der Ergebnisse

Die hier vorgestellte Arbeit hat in den zurückliegenden Kapiteln einen großen Umfang an Themen vom Komplexitätsbegriff, über vorhandene Ansätze und der Entwicklung eines Ansatzes zur Komplexitätsmessung behandelt. In diesem Kapitel sollen die einzelnen Fäden wieder aufgegriffen werden um eine Zusammenfassung und eine Schlussbewertung der vorliegenden Arbeit durchzuführen.

## 8.1 Problemstellungen der Arbeit

Aufbauend auf der Motivation einer Leistungsmessung in der Produktentwicklung wurden in Abschnitt 1.2 drei wissenschaftliche Fragestellungen dieser Arbeit entwickelt. Im Rückblick auf diese Arbeit wird nun dargestellt, wie diese Fragestellungen durch die erzielten Ergebnisse beantwortet werden.

### 8.1.1 Theoretische Fragestellung

*Wie können Verständnisse von Komplexität in der Produktentwicklung in der Literatur geordnet werden und wie kann eine Definition gestaltet werden, die sich an den Zielen der Produktivitätsmessung in der Produktentwicklung orientiert? (Siehe Seite 13)*

Da der Komplexitätsbegriff schwer zu fassen ist und es sehr unterschiedliche Verständnisse von Komplexität in den Disziplinen der Produktentwicklung gibt, wurde der Komplexitätsbegriff ergiebig in Abschnitt 2.1 diskutiert. Zunächst lässt sich feststellen, dass sich der Komplexitätsbegriff bestehender Ansätze in der Literatur an der jeweiligen Aufgabenstellung orientiert. Trotz der vielen Unterschiede lassen sich für die Quantifizierung der Komplexität drei Prinzipien unterscheiden: Schwierigkeit der Beschreibung, Aufwand der Erzeugung und Grad der Organisation. Eine Einordnung der Literatur in Form des Abschnitts 2.1.3.3 rundet die Beantwortet des ersten Teils dieser Fragestellung ab.

Der zweite Teil der Fragestellung wird durch den Abschnitt 2.2 adressiert. Der wichtigste Aspekt der Definition ist das Verständnis von Komplexität als Schwierigkeit der Erzeugung. Aufbauend auf dieser Grundüberlegung und der beobachteten Bedeutung von Relationen in der Literatur wurde eine Definition der Komplexität von Produktmodellen entwickelt. Ergänzende hierzu wurde ein Anforderungskatalog für einen Ansatz zur Messung dieser Komplexität erstellt.

### 8.1.2 Konzeptionelle Fragestellung

*Wie kann ein Metamodell von Produktmodellen zur Komplexitätsbewertung definiert und die quantifizierende Analyse dieser Modelle gestaltet werden? (Siehe Seite 13)*

Die Konzeption des Ansatzes zur Komplexitätsmessung von Produktmodellen in Kapitel 5 behandelt diese Fragestellung auf Basis der Beantwortung der theoretischen Fragestellung in Kapitel 2 und der Untersuchung der verwandten Ansätze in der Literatur. Kapitel 3 betrachtet hierbei die Zielsetzung, während Kapitel 4 methodisch relevante Ansätze diskutiert. Es werden jeweils Abgrenzung und gegebenenfalls Bezüge zum Stand der Technik in Abschnitt 5.9 hergestellt.

Die Formulierung des Metamodells erfolgt in Abschnitt 5.3. Es setzt bewusst lediglich eine Beschreibung des Produktmodells als Graph voraus. Auf Basis der Betrachtung eines Produktmodells als Graphen wird die quantifizierende Analyse durch die Kombination von Knotenmaßen als Attribute und einer Regressionsanalyse gestaltet. Ein besonderes Augenmerk wird auf die Selektion der Attribute gelegt.

### 8.1.3 Praktische Fragestellung

*Wie muss ein Werkzeug zur Komplexitätsanalyse gestaltet sein, um eine Produktivitätsbewertung anhand von Produktmodellen durchzuführen? (Siehe Seite 14)*

Teil dieser Arbeit ist die Entwicklung einer prototypischen Implementierung des Ansatzes. Die technologischen Entscheidungen und die Gestaltung der Implementierung werden in Kapitel 6 beschrieben. Grundlage eines Werkzeuges zur Komplexitätsanalyse ist ein flexibles Modellierungsframework und die Unterstützung bei der Erstellung und Ausführung von Komplexitätsmaßen durch Assistenten. Die Erfahrungen aus der Implementierung auf Basis des Eclipse Modeling Frameworks können auf andere Technologien übertragen werden.

Die Erkenntnisse aus der Implementierung wurden durch die Erfahrungen aus der Validierung in Kapitel 7 bestätigt und ergänzt. Für die Durchführung der Komplexitätsanalyse ist die Integration in bestehende Entwicklungsumgebungen zu empfehlen. Hierdurch kann insbesondere die aufwändige Erfassung der Produktmodelle im Vergleich zu den Fallstudien vereinfacht werden.

## 8.2 Wissenschaftlicher Beitrag

Die vorliegende Arbeit versteht sich als einen Beitrag zur wissenschaftlichen Untersuchung von Produktentwicklungen. In diesem Abschnitt soll der wesentliche wissenschaftliche Beitrag zusammengefasst werden. Die Ergebnisse dieser Arbeit stellen einen signifikanten Mehrwert gegenüber dem Stand der Wissenschaft in folgenden Punkten dar.

- Diese Arbeit liefert eine Strukturierung der Komplexitätsbegriffe in der Produktentwicklung, die in dieser Breite bisher nicht erfolgt ist. Es werden bestehende Ansätze strukturiert und voneinander abgegrenzt. Somit wird ein wertvoller Beitrag zur wissenschaftlichen Diskussion der Komplexität gemacht, die leider bisher oft von gegenseitigen Missverständnissen geprägt ist.
- Bisherige Ansätze von Komplexitätsmaßen basieren in der Regel auf der einmaligen Analyse einer Menge von Referenzprojekten. Die gegebenenfalls ebenfalls mit statistischen Mitteln erlangten Komplexitätsmaße können jedoch nicht als allgemeingültig anerkannt werden und besitzen nur eine begrenzte „Haltbarkeit“. Durch die immer schnelleren Änderungen im Bereich der Softwareentwicklung erscheint dieses wissenschaftliche Vorgehen nicht mehr nachhaltig. Dem gegenüber bietet diese Arbeit einen Ansatz zur individuellen Erzeugung von Komplexitätsmaßen, die an neue Gegebenheiten schnell angepasst werden können.
- Bestehende Arbeiten konzentrieren sich auf bestimmte Partialmodelle. Der hier vorgestellte Ansatz ist partialmodellübergreifend. Die Validierung zeigt, dass er in verschiedenen Domänen eingesetzt werden kann. Durch die Beschränkung auf die Bedingung einer graphenbasierten Darstellung ist gegenüber bestehenden Ansätzen die Flexibilität hervorzuheben.
- Nach Kenntnis des Autors ist die Beschreibung von Produktmodellen mit Hilfe automatisch generierter und bewerteter Attribute in den Ingenieurwissenschaften bisher nicht vorgenommen worden. Diese computergestützte und rechenintensive Form der Untersuchung von Produktmodellen stellt einen wesentlichen Beitrag zum Feld der Strukturanalyse dar.

Der entwickelte Ansatz kann als Grundlage und Anregung für weitere Arbeiten gesehen werden. Insbesondere die Kombination mit den klassischen matrizenbasierten Ansätzen aus Abschnitt 4.1 wie der Design Structure Matrix stellt einen interessanten wissenschaftlichen Ausblick dar. Die vorgestellte Arbeit ist insgesamt als wertvoller Beitrag zum Stand der Wissenschaft zu sehen.

## 8.3 Fazit

Komplexität ist und bleibt ein vielschichtiger und schwieriger Aspekt der Produktentwicklung. An dieser Tatsache ändert diese Arbeit nichts, aber sie eröffnet einen Weg zu einem besseren Verständnis und einem besseren Umgang mit Komplexität. Vor dem Hintergrund der Produktivitätsmessung wurde ein flexibler und wissenschaftlich fundierter Ansatz zur Quantifizierung von Produktmodellen entwickelt. Die Beantwortung der Problemstellungen und der wissenschaftliche Beitrag zeigen, dass diese Arbeit das gesteckte Ziel erreicht hat: Die Komplexitätsmessung von Produktmodellen.

# Anhang

## Glossar

**Attribut**

Eine nominal oder metrisch messbare Eigenschaft eines Elements in einem Produktmodell. Beispiel: Anzahl ausgehender Relationen eines Typs. Siehe Abschnitt 5.5.

**Attributmuster**

Ein abstraktes Attribut, das durch ein oder mehrere Parameter als Attribut definiert werden kann. Beispiel: Anzahl ausgehender Relationen mit dem Typ der Relation als Parameter. Siehe Abschnitt 5.5.1 zur Erzeugung von Attributen aus Attributmustern.

**Artefakt**

Ein Element eines Produktmodells, dem eine Komplexität zugeordnet werden kann. Zum Verhältnis Artefakt und Komplexität siehe Abschnitt 2.2.2.

**CAD (Computer Aided Design)**

Der computergestützte Entwurf, insbesondere für die geometrische Modellierung.

**Disziplin**

Eine durch Aufgabenbereich, Sprache, Methodik und Kultur abgrenzbare Einzelwissenschaft der Produktentwicklung. Beispiele: Maschinenbau, Elektrotechnik.

**Domäne**

Der Anwendungsbereich einer Produktentwicklung oder die Disziplin. Meistens bestimmte Industriezweige. Beispiele: Luft- und Raumfahrtindustrie, Softwareentwicklung, Anwaltskanzleien, Konsumentenelektronik.

**Effizienz**

Das Verhältnis von Ertrag zu einem eventuell nur theoretisch möglichen, maximalen Ertrag. Zur Effizienztheorie siehe Abschnitt 3.1.1.

**EMF (Eclipse Modeling Framework)**

Java-basiertes Modellierungs-Framework, das als Basistechnologie des Prototyps fungiert. Zu Eigenschaften und Vergleich siehe Abschnitt 6.1.

**Entwicklungssystem**

Die Gesamtheit aller Elemente, inklusive deren Beziehungen untereinander, die ein bestimmtes Produkt erzeugen.

**Entwicklungsprozess**

Ein aus Einzelschritten definierter Prozess für die Entwicklung ähnlicher Produkte.

**Feature**

Ein prozedurales Element eines CATIA-Modells zur Definition eines 3D-Objekts.

**Komplexität**

Im Rahmen dieser Arbeit die durch den Aufwand gewichtete Semantik einer Beziehung. Zur Literatur der Komplexität siehe 2.1. Zur Definition für diese Arbeit siehe Abschnitt 2.2.2.

**Konvergenzwert**

Eine Kennzahl eines Komplexitätsmaßes. Bezeichnet die in der Regression statistisch ermittelte benötigte Anzahl an Artefakten in einem Produktmodell um mit einer Wahrscheinlichkeit von mindestens 95,45% einen Messfehler von höchstens 10% zu erhalten. Zur Erläuterung und Herleitung siehe 5.6.1.8.

**Partialmodell**

Funktional, methodisch oder durch Werkzeugunterstützung abgrenzbare Teile des Produktmodells. Beispiele: Anforderungsmodell, CAD-Modell, Funktionsmodell.

**Produkt**

Im Sinne der Produktentwicklung eine abgrenzbare Menge von Informationen, um bestimmte Waren zu erzeugen. Hier zu unterscheiden von der physischen Produktinstanz.

**Produktentwicklung**

Der Vorgang der Erzeugung von Produktinformationen.

**Produktmodell**

Die Gesamtheit aller Informationen über ein Produkt. Siehe Abschnitt 1.1.

**Produktivität**

Das Verhältnis von Aufwand zum Ertrag. Siehe Abschnitt 3.1.1.

**Produktstruktur**

Die Elemente eines Produktes und deren Relationen, die den generellen Aufbau bestimmen. Bei einem physischen Bauteil sind dies zum Beispiel die Unterteile und deren Zusammenbau. Bei einer Software sind dies in der Regel die Klassen und deren Zuordnung zu Paketen.

**Prototyp**

Im Rahmen dieser Arbeit die speziell erstellte Software, die zur Analyse der Validierungsfallbeispiele eingesetzt wurde. Siehe Kapitel 6.

**Referenzproduktmodell**

Ein Produktmodell, das zur Erstellung eines Komplexitätsmaßes eingesetzt wird. Zur relativen Komplexitätsmessung siehe Abschnitt 5.2.

**Regressor**

Innerhalb des Prototyps ein Java-Interface, das ein Attribut implementiert.

**RegressorFactory**

Innerhalb des Prototyps ein Java-Interface, das Attribute auf Basis eines Produktmodells erzeugt und somit einem Attributmustern im Wesentlichen entspricht.

## Attributmuster

Attribute bewerten Artefaktknoten in einem Graphen metrisch oder nominal. Sie werden im Rahmen des Ansatzes dieser Arbeit zur Abbildung von Eigenschaften des Produktmodells in das Regressionsmodell verwendet. Im Prototyp der Validierung werden Attribute durch Unterklassen der abstrakten Klasse `Regressor` implementiert.

Einige Regressoren beziehen sich auf EReferences (Relationen in EMF), die als Containment-References definiert sind. Containments entsprechen einer Aggregation in UML und werden genutzt um eine Baumstruktur der Objekte im EMF-Modell zu beschreiben. Die Berücksichtigung dieser Baumstruktur findet bei den `Tree`-Regressoren statt.

In der Tabelle ist der Name der Implementierung ohne die Endung Regressor angegeben (zum Beispiel `ClassRegressor` nur `Class`). In der Spalte Parameter sind die EMF-Objekte mit Variablennamen angegeben, die zur Konfiguration einer Instanz verwendet werden. Das Artefakt wird allgemein nur mit A bezeichnet.

Tabelle 27: Attributmuster im Prototyp

| **Name** | **Parameter** | **Beschreibung auf Graphebene** |
|---|---|---|
| Class | EClass C | Testet ob A direkte oder indirekte Instanz von C ist (Wert 1 oder 0). |
| Numeric-Attribute | EAttribute B | Wert des numerischen Attributs B von A. |
| Reference-Cardinality | EReference D | Anzahl ausgehender Kanten des Typs D |
| Reference-Value | EReference D<br>EObject O | Testet ob A über die Kante D mit Knoten O verbunden ist (Wert 1 oder 0). |
| Usage | EReference D | Anzahl eingehender Kanten des Typs D |
| Path | List<EReference>D*<br>Regressor R | Wendet den Regressor R auf alle Knoten an, die durch einen Kantenzug D* von A aus erreichbar sind. |
| TreeClass | EClass C | Die Anzahl Instanzen von C im Baum unter A. |
| TreeCross-Reference | EReference D | Anzahl ausgehender Relationen der Knoten im Baum unter A, deren Ziel kein Knoten in diesem Baum ist. |
| TreeNode-Depth | | Tiefe des Baumes unter A |
| TreeAverage-Children | | Durchschnittliche Anzahl der Kinderknoten für die Knoten des Baumes unter A. |

# Literatur


Abran, Alain, und Pierre N Robillard. 1994. „Function Points: A Study of Their Measurement Processes and Scale Transformations“. *JOURNAL OF SYSTEMS AND SOFTWARE* 25: 171--184.

Ahn, J. S, S. Ramaswamy, und R. H Crawford. 1996. „A formalism for modeling engineering design processes“. *IEEE Transactions on Systems, Man and Cybernetics (SMC)* (Juli).

Aigner, D., C. A.K Lovell, und P. Schmidt. 1977. „Formulation and estimation of stochastic frontier production function models“. *Journal of econometrics* 6 (1): 21–37.

Akaike, H. 1974. „A new look at the statistical model identification“. *IEEE Transactions on Automatic Control* 19 (6) (Dezember): 716- 723. doi:10.1109/TAC.1974.1100705.

Akao, Yoji. 1994. The Customer Driven Approach to Quality Planning and Deployment. In *Development History of Quality Function Deployment*, 339. Tokyo, Japan: Asian Productivity Organization.

Albrecht, A. J. 1979. Measuring Application Development Productivity. In *Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium*, 83—92. IBM Corporation, Oktober.

Alpaydin, Ethem. 2010. *Introduction to Machine Learning, Second Edition*. 2. Aufl. The MIT Press, Februar 26.

Ameri, Farhad, Joshua Summers, Gregory Mocko, und Matthew Porter. 2008. „Engineering design complexity: an investigation of methods and measures“. *Research in Engineering Design* 19 (2) (November 1): 161-179. doi:10.1007/s00163-008-0053-2.

Anderl, Reiner, und Dietmar Trippener. 2000. *STEP, Standard for the Exchange of Product Model Data*. Teubner Verlag.

Automotive SIG - The SPICE User Group, Hrsg. 2010. *Automotive SPICE - Process Assessment Model*. http://www.automotivespice.com/.

Baccarini, David. 1996. „The concept of project complexity--a review“. *International Journal of Project Management* 14 (4) (August): 201-204. doi:10.1016/0263-7863(95)00093-3.

Backhaus, Klaus, Bernd Erichson, Wulff Plinke, und Rolf Weiber. 2008. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 12. Aufl. Springer, Berlin, September 22.

Bailey, John W., und Victor R. Basili. 1981. A meta-model for software development resource expenditures. In *Proceedings of the 5th international conference on Software engineering*, 107-116. San Diego, California, United States: IEEE Press.

Balazova, Maria. 2004. Methode zur Leistungsbewertung und Leistungssteigerung der Mechatronikentwicklung. Dissertation, Paderborn: Universität Paderborn. http://deposit.ddb.de/cgi-bin/dokserv?idn=974402613&dok_var=d1&dok_ext=pdf&filename=974402613.pdf.

Bashir, Hamdi A, und Vince Thomson. 1999a. „Metrics for design projects: a review“. *Design Studies* 20 (3) (Mai): 263-277. doi:10.1016/S0142-694X(98)00024-6.

Bashir, Hamdi A., und Vince Thomson. 1999b. „Estimating Design Complexity“. *Journal of Engineering Design* 10 (3): 247. doi:10.1080/095448299261317.

Basili, Victor R, Gianluigi Caldiera, H. Dieter Rombach, und John J Marciniak. 1994. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*, 1:528–532. John Wiley & Sons.

Basili, Victor R, Gianluigi Caldiera, und H. Dieter Rombach. 1994. Goal Question Metric Approach. In *Encyclopedia of Software Engineering*, 1:528—532. John Wiley & Sons.

Billson, Cleve. 2002. A History of the London Tube Maps. http://homepage.ntlworld.com/clivebillson/tube/tube.html.

Boehm, B. W. 1981. „Software engineering economics“. *Englewood Cliffs*.

Boehm, B. W, R. Madachy, und B. Steece. 2000. *Software Cost Estimation with Cocomo II with Cdrom*. Prentice Hall PTR Upper Saddle River, NJ, USA.

Browning, T. R. 2001. „Applying the design structure matrix to system decomposition and integration problems: a review and new directions“. *IEEE Transactions on Engineering management* 48 (3): 292–306.

Bürgel, Hans Dietmar. 1989. *Controlling von Forschung und Entwicklung. Erkenntnisse und Erfahrungen aus der Praxis*. 1. Aufl. Vahlen Franz GmbH, November 10.

Burghardt, Manfred. 2006. *Projektmanagement: Leitfaden für die Planung, Überwachung und Steuerung von Entwicklungsprojekten*. 7. Aufl. Publicis Corporate Publishing, August 21.

Campbell, Donald J. 1988. „Task Complexity: A Review and Analysis“. *The Academy of Management Review* 13 (1) (Januar): 40-52.

Carnegie Mellon University - Software Engineering Institute. 2010. *CMMI for Development*. http://www.sei.cmu.edu/cmmi/.

Charnes, A., W. W. Cooper, und E. Rhodes. 1978. „Measuring the efficiency of decision making units“. *European Journal of Operational Research* 2 (6) (November): 429-444. doi:10.1016/0377-2217(78)90138-8.

Chen, P. P. 1976. „The entity-relationship model—toward a unified view of data“. *ACM Transactions on database systems* 1 (1): 9–36.

Coelli, Timothy J., Dodla Sai Prasada Rao, Christopher J. O'Donnell, und George Edward Battese. 2005. *An Introduction to Efficiency and Productivity Analysis*. 2. Aufl. Springer, Juli 22.

Collett, Toby H J, Bruce A MacDonald, und Brian Gerkey. 2005. „Player 2.0: Toward a practical robot programming framework“. *Australasian Conference on Robotics and Automation*.

Danilovic, M., und H. Börjesson. 2001. Managing the multiproject environment. In Cambridge.

Dardenne, Anne, Axel van Lamsweerde, und Stephen Fickas. 1993. „Goal-directed requirements acquisition“. *Science of Computer Programming* 20 (1-2) (April): 3-50. doi:10.1016/0167-6423(93)90021-G.

Davis, J.S., und R.J. LeBlanc. 1988. „A study of the applicability of complexity measures“. *Software Engineering, IEEE Transactions on* 14 (9): 1366-1372. doi:10.1109/32.6179.

DeMarco, Tom. 1983. *Controlling Software Projects: Management, Measurement and Estimation*. Yourdon Press, Januar 1.

Dierneder, Stefan, und Rudolf Scheidl. 2001. Complexity Analysis of Systems from a Functional and Technical Viewpoint. In *Computer Aided Systems Theory — EUROCAST 2001*, hg von. Roberto Moreno-Díaz, Bruno Buchberger, und José Luis Freire, 2178:261-272. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. http://dx.doi.org/10.1007/3-540-45654-6_18.

Eckert, C. M, C. E Earl, und P. J Clarkson. 2005. Predictability of change in engineering: A complexity view. Book Chapter. http://oro.open.ac.uk/7413/.

Edmonds, Bruce. 1999. Syntactic Measures of Complexity. University of Manchester.

Ehrlenspiel, Klaus. 2006. *Integrierte Produktentwicklung. Denkabläufe, Methodeneinsatz, Zusammenarbeit*. Hanser Fachbuchverlag.

Eichinger, Markus, Maik Maurer, Udo Pulm, und Udo Lindemann. 2006. „Extending Design Structure Matrices and Domain Mapping Matrices by Multiple Design Structure Matrices“. *ASME Conference Proceedings* 2006 (42487) (Januar 1): 889-898. doi:10.1115/ESDA2006-95266.

El-Haik, B., und K. Yang. 1999. „The components of complexity in engineering design“. *IIE Transactions* 31 (10): 925–934.

Eversheim, W., T. Breuer, und M. Grawatsch. 2001. Combining the scenario technique with QFD and TRIZ to a product innovation methodology. In .

Fahrmeir, Ludwig, Thomas Kneib, und Stefan Lang. 2008. *Regression. Modelle, Methoden und Anwendungen*. 1. Aufl. Springer, Berlin, November.

Farrell, M. J. 1957. „The Measurement of Productive Efficiency“. *Journal of the Royal Statistical Society. Series A (General)* 120 (3): 253-290.

Feldhusen, J., und Markus Koy. 2002. „Methode zur Produktivitätsmessung für Entwicklung und Konstruktion“. *Konstruktion* (09/2002).

Feldman, D. P, und J. Crutchfield. 1998. „A Survey of “Complexity Measures”“. *Santa Fe Institute, USA* 11.

Fenton, Norman, und Shari Lawrence Pfleeger. 1996. *Software Metrics - A Rigorous and Practical Approach*. 2. Aufl. London: International Thomson Computer Press.

Fleming, Quentin W, und Joel M Koppelman. 2006. *Earned Value Project Management*. 3. Aufl. Project Management Institute.

Gell-Mann, M., und S. Lloyd. 2004. „Effective complexity“. *Nonextensive entropy: interdisciplinary applications*: 387.

Golder, P. N. 2000. „Insights from senior executives about innovation in international markets“.

Gotel, O. C. Z, und C. W Finkelstein. 1994. An analysis of the requirements traceability problem. In *Requirements Engineering, Proceedings of the First International Conference on*, 94—101. April. doi:10.1109/ICRE.1994.292398.

Griffin, Abbie. 1993. „Metrics for measuring product development cycle time“. *Journal of Product Innovation Management* 10 (2) (März): 112-125. doi:10.1016/0737-6782(93)90003-9.

Hall, M. A. 1999. Correlation-based feature selection for machine learning. Citeseer.

Hall, M. A, und L. A Smith. 1997. Feature subset selection: a correlation based filter approach. In *International Conference on Neural Information Processing and Intelligent Information Systems*, 855–858.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, und I.H. Witten. 2009. „The WEKA data mining software: an update“. *ACM SIGKDD Explorations Newsletter* 11 (1): 10–18.

Halstead, Maurice H. 1977. *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc.

Hauser, John R, und Florian Zettelmeyer. 1996. *Metrics to evaluate R,D&E*. Massachusetts Institute of Technology (MIT), Sloan School of Management. Massachusetts Institute of Technology, Oktober. http://hdl.handle.net/1721.1/2637.

Häusler, Stefan, Frank Poppen, Sonja Preis, Kevin Hausmann, Wolfgang Nebel, Axel Hahn, Peter Leppelt, Amir Hassine, und Erich Barke. 2007. Modellierung von Komplexität und Qualität als Faktoren von Produktivität in Design-Flows für integrierte Schaltungen. In *edaWorkshop*.

Hausmann, Kevin. 2008. Permeter – Performanzmessung in der Produktentwicklung auf Basis semantisch integrierter Produktmodelle. Carl von Ossietzky Universität Oldenburg.

Henry, S., und D. Kafura. 1981. „Software Structure Metrics Based on Information Flow“. *IEEE Trans. Softw. Eng.* 7 (5): 510-518.

Hoare, C. A. R. 1962. „Quicksort“. *The Computer Journal* 5 (1) (Januar 1): 10-16. doi:10.1093/comjnl/5.1.10.

Horvath, Peter. 1996. *Controlling*. 6. Aufl. Vahlen.

Hubbert, J. 2003. „Bis an die Grenzen gefordert“. *Automobilindustrie* 48 (11): 28-32.

Humphreys, Gary C. 2002. *Project Management Using Earned Value*. 1. Aufl. Humphreys & Assoc., Januar 1.

Jacome, M.F., und V. Lapinskii. 1997. „NREC: risk assessment and planning of complex designs“. *IEEE Design & Test of Computers* 14 (1) (März): 42-49. doi:10.1109/54.573364.

Johnson, C. W. 2005. „What are emergent properties and how do they affect the engineering of complex systems“. *Reliability Engineering and System Safety* 91: 1475–1481.

Jungkunz, Ralf M. 2005. *PDM-basierte Überwachung komplexer Entwicklungsprojekte*. München: Herbert Utz Verlag.

Kearney, Joseph P., Robert L. Sedlmeyer, William B. Thompson, Michael A. Gray, und Michael A. Adler. 1986. „Software complexity measurement“. *Commun. ACM* 29 (11): 1044-1050. doi:10.1145/7538.7540.

Kerssens-van Drongelen, Inge c., und Jan Bilderbeek. 1999. „R&D performance measurement: more than choosing a set of metrics“. *R and D Management* 29 (1) (Januar): 35-46. doi:10.1111/1467-9310.00115.

Kohn, Wolfgang. 2004. *Statistik*. Springer, September 15.

Kreimeyer, Matthias F. 2010. A Structural Measurement System for Engineering Design Processes. Dissertation, München: Technische Universität München.

Lindemann, Udo, Maik Maurer, und Thomas Braun. 2008. *Structural Complexity Management: An Approach for the Field of Product Design*. illustrated edition. Springer, Berlin, November.

Lloyd, S. 2001. Measures of complexity: a nonexhaustive list. *Control Systems Magazine, IEEE* 21, no. 4: 7-8. doi:10.1109/MCS.2001.939938.

Lloyd, S. 2007. *Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos*. Vintage, März 13.

Manso, Ma Esperanza, Marcela Genero, und Mario Piattini. 2003. No-redundant Metrics for UML Class Diagram Structural Complexity. In *Advanced Information Systems Engineering*, hg von. Johann Eder und Michele Missikoff, 2681:127-142. Berlin, Heidelberg: Springer Berlin Heidelberg. http://www.springerlink.com/content/b02cfuyxr1acf3gj/.

Mario Piattini, Marcela Genero, Geert Poels, und Jim Nelson. 2005. Towards A Framework for Conceptual Modelling Quality. In *Metrics For Software Conceptual Models*. Imperial College Press.

Maurer, S. Maik. 2007. Structural Awareness in Complex Product Design. Technische Universität München.

McAllister, J. W. 2003. „Effective complexity as a measure of information content“. *Philosophy of Science* 70 (2): 302–307.

McCabe, TJ. 1976. „A Complexity Measure“. *Software Engineering, IEEE Transactions on* SE-2 (4): 320, 308.

Meeusen, Wim, und Julien van Den Broeck. 1977. „Efficiency Estimation from Cobb-Douglas Production Functions with Composed Error“. *International Economic Review* 18 (2) (Juni): 435-444.

Mill, John Stuart. 1843. *A system of logic : ratiocinative and inductive ; being a connected view of the principles of evidence and the methods of scientific investigation*.

Moody, D. L. 2005. „Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions“. *Data & Knowledge Engineering* 55 (3): 243–276.

Moody, Daniel L. 1998. Metrics for Evaluating the Quality of Entity Relationship Models. In *Object-Oriented and Entity-Relationship Modelling/International Conference on Conceptual Modeling / the Entity Relationship Approach*, 211–225.

Norden, P.V. 1964. Manpower utilization patterns in research and development projects. Columbia University, New York.

Ojanen, Ville, und Olli Vuola. 2006. „Coping with the multiple dimensions of R&D performance analysis“. *International Journal of Technology Management* 33 (2/3): 279—290. doi:10.1504/IJTM.2006.008315.

Oviedo, E.I. 1980. Control Flow, Data Flow and Program Complexity. In *Proceedings of the 4th IEEE International Computer Software and Applications Conference*, 146-152.

Parr, T. J., und R. W. Quong. 1995. „ANTLR: a predicated-LL(k) parser generator“. *Software - Practice & Experience* 25 (7): 789-810.

Pearson, K. 1901. „On lines and planes of closest fit to systems of points in space“. *Philosophical Magazine* 2 (6): 559-572.

Pimmler, Thomas U., und Steven D. Eppinger. 1994. Integration analysis of product decompositions. In *Design theory and methodology, DTM'94*, 343. Minneapolis, Minnesota.

Project Management Institute. 2008. *A Guide to the Project Management Body of Knowledge:* 4. Aufl. Project Management Institute, Dezember 31.

ReACT-Gruppe. 2011. Abschlussdokument der Projektgruppe „ReACT“an der Universität Oldenburg. März 31. http://www-is.informatik.uni-oldenburg.de/~dibo/pg_fb10/endberichte/2011/ReACT.pdf.

Rosenberg, J. 1997. Some Misconceptions About Lines of Code. In *Software Metrics, IEEE International Symposium on*, 137. Los Alamitos, CA, USA: IEEE Computer Society. doi:http://doi.ieeecomputersociety.org/10.1109/METRIC.1997.637174.

Royce, W. W. 1970. Managing the development of large software systems. In *Proceedings of IEEE Wescon*, 26:9.

Salman, Nael, und Ali H Dogru. 2006. Complexity and Development Effort Prediction Models Using Component Oriented Metrics. In *Proceedings of MENSURA 2006*. November 6.

Schwarz, Gideon. 1978. „Estimating the Dimension of a Model“. *The Annals of Statistics* 6 (2) (März): 461-464. doi:10.1214/aos/1176344136.

Segal, David. 2010. „It's Complicated: Making Sense of Complexity“. *The New York Times*, Mai 1, Abschn. Week in Review. http://www.nytimes.com/2010/05/02/weekinreview/02segal.html.

Shannon, C. E. 1948. „A Mathematical Theory of Communication“. *The Bell System Technical Journal* 27 (3): 379–423.

Solomonoff, R. J. 1964. „A formal theory of inductive inference. Parts I and II“. *Information and control* 7 (2): 224–254.

Stelzer, Dirk, und Winfried Bratfisch. 2007. „Earned-Value-Analyse – Controlling-Instrument für IT-Projekte und IT-Projektportfolios“. *HMD - Praxis der Wirtschaftsinformatik* (254): 61 -70.

Steward, D. 1981. „The design structure matrix: a method for managing the design of complex systems“. *IEEE Transactions on Engineering Management* 28 (3): 71–74.

Strickmann, Jan. 2008. Analysemethoden zur Bewertung vonEntwicklungsprojekten. Universität Oldenburg.

Suh, Nam P. 1990. *The Principles of Design*. Oxford Univ Pr, April 19.

Suh, Nam P. 1999. „A Theory of Complexity, Periodicity and the Design Axioms“. *Research in Engineering Design* 11 (2): 116-132. doi:10.1007/PL00003883.

Suh, Nam Pyo. 2001. *Axiomatic Design: Advances and Applications*. Oxford University Press, USA, Mai 17.

Summers, Joshua D., und Jami J. Shah. 2003. Developing measures of complexity for engineering design. In *ASME DETC-2003*.

Terninko, John, Alla Zusman, und Boris Zlotin. 1998. *Systematic innovation: an introduction to TRIZ ; (theory of inventive problem solving)*. St. Lucie Press.

Turner, J. R., und R. A. Cochrane. 1993. „Goals-and-methods matrix: coping with projects with ill defined goals and/or methods of achieving them“. *International Journal of Project Management* 11 (2) (Mai): 93-102. doi:10.1016/0263-7863(93)90017-H.

Usher, John M., Utpal Roy, und Hamid Parsaei. 1998. *Integrated Product and Process Development: Methods, Tools, and Technologies*. 1. Aufl. Wiley-Interscience, Februar 27.

VDI2221E. 1985. „Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte“. *Düsseldorf: VDI-Verlag*.

Wallace, Linda Genelle. 1999. The development of an instrument to measure software project risk. Georgia State University. http://portal.acm.org/citation.cfm?id=928882.

Wallance, Linda, Mark Keil, und Arun Rai. 2004. „Understanding software project risk: a cluster analysis“. *Information & Management* 42 (1) (Dezember): 115-125. doi:10.1016/j.im.2003.12.007.

Weaver, W. 1948. „Science and complexity“. *American Scientist* 36 (4) (Oktober): 536-544.

Williams, T. M. 1999. „The need for new paradigms for complex projects“. *International Journal of Project Management* 17 (5) (Oktober): 269-273. doi:10.1016/S0263-7863(98)00047-7.

Williams, Terry. 2002. *Modelling Complex Projects*. John Wiley & Sons, August 27.

Yassine, A. 2004. „An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method“. *Italian Management Review* 2004 (9).

Zickert, Frank, und Roman Beck. 2010. „Because Effort Matters!“ *Business & Information Systems Engineering* 2 (3) (April): 165-173. doi:10.1007/s12599-010-0103-y.

Ziegenbein, Klaus. 2004. *Controlling. Kompendium der praktischen Betriebswirtschaft*. 8. Aufl. Kiehl Friedrich Verlag G, August 27.

Zuse, Horst. 1990. *Software Complexity: Measures and Methods*. Walter de Gruyter, Dezember.