Flexible Methodenintegration in anpassbare Vorgehensmodelle

Ulrike Hammerschall

Institut für Informatik
der Technischen Universität München

Flexible Methodenintegration in anpassbare Vorgehensmodelle

Ulrike Hammerschall

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. H. Krcmar
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h.c. M. Broy
2.	Univ.-Prof. B. Brügge, Ph.D.

Die Dissertation wurde am 10.03.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 26.08.2008 angenommen.

Danksagung

An dieser Stelle möchte ich mich herzlich bei Herrn Prof. Broy für die Betreuung und Unterstützung bei der Erstellung dieser Arbeit bedanken. Mit seine gezielten und hilfreichen Anmerkungen und Kommentaren half er immer wieder diese Arbeit in die richtige Richtung zu lenken und weiter voran zu bringen. Gleichzeitig möchte ich Herrn Prof. Brügge für seine Bereitschaft danken, die Zweitgutachtung meiner Arbeit zu übernehmen.

Besonders möchte ich mich auch bei meinen langjährigen Arbeitskollegen und -kolleginnen am Lehrstuhl, insbesondere aus dem Projektumfeld von Weit und V-Bench, für die gute Zusammenarbeit und das angenehme und kollegiale Umfeld bedanken. Erwähnen möchte ich an dieser Stelle vor allem auch meinem langjährigen Zimmerkollegen Gerd Beneken sowie meinen Lehrstuhlkollegen Tilman Seifert. Die vielen Diskussionen zu verschiedenen Themen haben mir immer wieder neue und hilfreiche Denkanstöße gegeben.

Besonders wichtig ist mir an dieser Stelle meinem Mann Thomas danken: für seine Geduld beim Korrekturlesen der verschiedenen Zwischenversionen, Endversionen und 'endgültigen' Endversionen, für seine Bereitschaft, zu allen möglichen (und teilweise auch unmöglichen) Zeiten als geduldiger und aufmerksamer Diskussionspartner zur Verfügung zu stehen und nicht zuletzt für sein Verständnis und seine Unterstützung während der gesamten Zeit meiner Promotion.

Zusammenfassung

Ein Vorgehensmodell definiert Vorgaben zur Planung, Steuerung und Durchführung von Softwareentwicklungsprojekten. Existierende Vorgehensmodellstandards können danach unterschieden werden, ob sie eine in Projekten anzuwendende Methodik starr vorgeben (methodenspezifisch) oder ob sie methodenneutral sind. Organisationen stehen bei der Einführung eines Vorgehensmodellstandards somit vor der Wahl zwischen mangelnder Anpassbarkeit methodenspezifischer oder mangelnder methodischer Unterstützung methodenneutraler Vorgehensmodelle.

Ziel dieser Arbeit ist die Erweiterung von Vorgehens-Metamodellen um ein Metamodell zur flexiblen Integration von Methoden. Dazu wird eine Definition und Klassifikation von Methoden sowie eine formale Beschreibung von Methoden und ihrer Schnittstelle zu Vorgehensmodellen erarbeitet. Diese wird in Form eines Metamodells formalisiert. Mit diesem Ansatz werden die Vorteile methodenneutraler und methodenspezifischer Vorgehensmodelle geeignet kombiniert.

Abstract

A process model defines rules and guidelines for planing, controlling and executing software development projects. Existing process model standards differ in the way that they either provide a fixed methodology for all projects or that they don't offer any methodology whatsoever. Organizations that plan to introduce a new process model have to choose between a process model that lacks adaptability due to predefined methodology or a process model that lacks any methodical support.

Goal of the thesis is the development of a meta model that extends process meta models insofar that they support flexible method integration at project level. For that purpose a definition and classification of methods is given. Furthermore a formal specification of methods and their connection to process models is defined and a meta model for method integration is derived. In this way the approach combines the advantages of method-specific process models and process models without methodology.

Inhaltsverzeichnis

Kapitel 1

Einführung

In den letzten 60 Jahren erlebte unsere Gesellschaft einen rasanten Wandel von der Industriegesellschaft hin zur Informationsgesellschaft. Dies wurde durch die digitale Repräsentation von Informationen möglich, aber vor allem auch durch die Digitalisierung von Prozeduren zur Manipulation der Informationen, kurz durch Software. Mit der Digitalisierung wurden die Grenzen der Mechanik weitgehend überwunden, gleichzeitig mit den neuen Möglichkeiten stieg jedoch die Komplexität der zu erstellenden Softwaresysteme. Gerade durch ihre Flexibilität erlaubt Software die Umsetzung immer komplexerer Abläufe: sei dies im internationalen Geldverkehr, bei der Verwaltung von Versicherungen, bei der Überwachung von Flughäfen oder im medizinischen Bereich, um nur einige Beispiele zu nennen.

Heute lässt sich Software nicht mehr aus unserer Welt weg denken. Sie begleitet uns und beeinflusst unser Leben in vielen Bereichen und Situationen. Solange sie fehlerfrei arbeitet, nehmen wir sie in der Regel nicht mehr wahr. Unsere Abhängigkeit von funktionierender Software im täglichen Leben wird uns erst bewusst, wenn diese, aus welchen Gründen auch immer, einmal nicht wie erwartet funktionieren sollte; eine Tatsache, die immer wieder durch medienwirksame Softwarefehler unterstrichen wird.

Die Erfahrung zeigt, dass unsere Fähigkeit Software zu erstellen sich nicht parallel zu den steigenden fachlichen Anforderungen und der daraus resultierenden steigenden Komplexität der Software entwickelt hat. Bereits in den 60er Jahren wurde das Problem mangelnder Softwarequalität auf Grund wachsender Komplexität der Anforderungen erkannt. Man sprach von der 'Softwarekrise'. Damit war und ist die tägliche Erfahrung gemeint, dass Software in der Regel nicht termingerecht fertig gestellt wird, in der Entwicklung die geschätzten und genehmigten Budgets regelmäßig überschritten werden, die Wartungskosten jede Schätzung übertreffen und dies alles bei eher mangelhafter Gesamtqualität des Softwareproduktes (vgl. [96]). Eine treffende Beschreibung für die Ursachen der Softwarekrise findet sich bei Edsgar Dijkstra in seiner Dankesrede zum Turing Preis, später unter dem Titel 'The Humble Programmer' in den Communications of the ACM veröffentlicht:

> The major cause of the software crisis is, that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem. [37]

Als Antwort auf das Problem der Softwarekrise wurde 1968 auf der NATO Software Engineering Konferenz in Garmisch-Partenkirchen der Begriff des 'Software Engineerings' geprägt. Damit sollte die Notwendigkeit deutlich gemacht werden, Softwareentwicklung als eine Ingenieursdisziplin zu betrachten. Ziel des Software Engineering ist es, mit Hilfe ingenieursmäßiger Verfahren und Techniken die Fähigkeit zur Durchführung von Softwareentwicklungsprojekten und damit die Qualität der resultierenden Software deutlich zu verbessern (vgl. [84]).

Wurde in den ersten Jahren der Informationstechnologie die Softwareentwicklung noch als Kunsthandwerk angesehen (vgl. [46]) und der Entwickler als kreativer Künstler, so entspricht dies heute längst nicht mehr den Erfordernissen der Softwareentwicklung. Komplexe Anwendungen fordern ein ingenieurmäßiges Vorgehen bei der Entwicklung. Dies impliziert insbesondere die Notwendigkeit eines adäquaten Entwicklungsprozesses. Ein geeigneter Entwicklungsprozess unterstützt die Ermittlung komplexer Anforderungen und ihre Dokumentation, die Umsetzung komplexer technischer Lösungen auf Basis der Anforderungen sowie die Verwaltung komplexer Projektabläufe. Dem liegt die Erfahrung zu Grunde, dass durch einen qualitativ hochwertigen Prozess die Qualität des Endprodukts, in diesem Fall der Software, vorhersagbar und planbar wird (vgl. [112]).

Da es sich bei der Informatik um eine relativ junge Wissenschaft handelt, liegt es nahe, hier direkt auf Erfahrungen der Ingenieurswissenschaften selbst zurückzugreifen und diese geeignet zu adaptieren. Entwicklungsprozesse können demnach im weitesten Sinn als Produktionsprozesse für Software betrachtet werden. Ein typischer Produktionsprozess zeichnet sich durch eine Reihe charakteristischer Eigenschaften aus. Nach [46] ist ein Prozess:

- Planbar: Der Prozess kann auf der Basis einzelner Arbeitsschritte vollständig geplant werden.
- Kalkulierbar: Die zur Durchführung der Arbeitsschritte notwendigen Aufwände und Ressourcen können im Vorfeld bestimmt werden.
- Überprüfbar: Zu jedem Zeitpunkt kann der Stand des Projekts hinsichtlich Qualität, Fortschritt und Ergebnis geprüft werden.
- Wiederholbar: Der Prozess kann in ähnlicher Form beliebig oft wiederholt werden.
- Produktneutral: Der Prozess kann für die Produktion unterschiedlicher Produkte verwendet werden.
- Flexibel: Der Prozess kann bei Bedarf flexibel an sich ändernde Anforderungen oder Rahmenbedingungen angepasst werden.

- Personenneutral: Der Prozess ist nicht personengebunden. Der Erfolg hängt nicht vom Wissen einzelner Personen ab.
- Arbeitsteilig: Der Prozess unterstützt explizit die Verteilung von Aufgaben und Verantwortlichkeiten auf unterschiedliche Personen.
- Werkzeugunterstützt: Der Prozess ist geeignet durch eine Werkzeugkette unterstützt.
- Messbar: Der Prozess kann mit Hilfe geeigneter Metriken gemessen werden. Dies ist Voraussetzung für eine kontinuierliche Bewertung und Verbesserung des Prozesses.

Vorgehensmodelle beschreiben Produktionsprozesse für Software. Ein Vorgehensmodell macht Vorgaben zur Planung, Steuerung und Durchführung von Softwareentwicklungsprojekten. Dazu definiert es, welche Rollen welche Tätigkeiten wann in einem Projekt durchzuführen haben um das gewünschte Ergebnis zum vorgegebenen Termin innerhalb des vorgegebenen Budgets und mit geforderter Qualität und Leistung zu erhalten. Ziel ist die Wiederholbarkeit, Überprüfbarkeit, Vergleichbarkeit und Planbarkeit des Softwareentwicklungsprozesses.

Der Einsatz von Vorgehensmodellen zur Verbesserung des Entwicklungsprozesses in Organisationen ist nicht neu. So wurden bereits in den 50er Jahren einfache Phasenmodelle eingesetzt (vgl. [14]), später konnte sich vor allem das Wasserfallmodell in vielen Organisationen etablieren. Die Situation hinsichtlich der Qualität von Softwareentwicklung hat sich seit den 60er Jahren, nicht zuletzt durch den Einsatz von Vorgehensmodellen, durchaus verbessert, kann jedoch immer noch nicht als optimal betrachtet werden.

So schwankt je nach Studie der Anteil erfolgreich durchgeführter IT Projekte zwischen 29 % weltweit (vgl. Chaos Report der Standish Group 2004 [130]), 16 % in Großbritannien für den Zeitraum 2002 / 2003 (vgl. [135]) und 50,7 % in Deutschland (vgl. SUCCESS Studie 2006 [24]). Die Unterschiede der Ergebnisse resultieren zum Großteil aus den unterschiedlichen Bewertungsmethoden und Kritierien. Ein objektives Maß für den Projekterfolg ist nicht immer eindeutig zu finden. Buschermöhle et al. als Autoren der SUCCESS Studie vermuten jedoch auch finanzielle Hintergründe: schlechte Studienergebnisse lassen sich besser verkaufen (vgl. [23]). Es kann daher angenommen werden, dass die Lage insgesamt etwas optimistischer ist, als es einige der Studien scheinen lassen. Dennoch ist deutlich zu erkennen, dass hier noch Verbesserungsbedarf besteht.

1.1 Vorgehensmodelle in Organisationen

Aufgabe eines Vorgehensmodells ist die Unterstützung einer Organisation bei der Kontrolle und Verbesserung ihrer Prozesse. Es leistet so einen nicht unerheblichen Beitrag zum Projekterfolg. Insbesondere große Unternehmen haben die Notwendigkeit unternehmensweiter Prozessvorgaben für die Softwareentwicklung erkannt. Aber auch für kleinere Unternehmen tritt die qualitative Verbesserung ihrer Entwicklungsprozesse mehr und mehr in den Vordergrund. Es wächst die Bereitschaft

zur Vereinheitlichung und Standardisierung der Prozesse, insbesondere auch im Zusammenhang mit Expansionsplänen. Einer Organisation, die zur Verbesserung der internen Prozesse ein Vorgehensmodell einführen möchte, stehen prinzipiell zwei Möglichkeiten zur Verfügung: die Entwicklung eines eigenen, proprietären Vorgehensmodells oder die Anpassung eines Vorgehensmodellstandards auf die eigenen Rahmenbedingungen und Bedürfnisse (der Begriff Standard ist dabei als Oberbegriff für offizielle Standards wie auch Herstellerstandards zu verstehen).

Proprietäre Vorgehensmodelle sind vollständig auf die Bedürfnisse einer spezifischen Organisation angepasst. Sie werden häufig auf Basis einzelner Beispielprojekte entwickelt. Die Projektergebnisse werden aufbereitet und zur Anwendung innerhalb der Organisation verallgemeinert. Als objektives Maß der Prozessqualität werden beispielsweise Reifegradmodelle wie CMMI [111] oder SPICE [67] herangezogen. Auf Grund ihrer Individualität weisen proprietäre Vorgehensmodelle teilweise starke Unterschiede hinsichtlich Darstellungsform, Umfang und Detaillierungsgrad auf. Eine Verallgemeinerung ist weder möglich noch sinnvoll. Proprietäre Vorgehensmodelle werden daher in dieser Arbeit nicht weiter betrachtet. Im Fokus stehen vielmehr frei oder kommerziell verfügbare Vorgehensmodellstandards, im Weiteren kurz als Vorgehensmodelle bezeichnet.

Die Heterogenität der Unternehmen erlaubt häufig nicht den unmittelbaren Einsatz eines Vorgehensmodells. Vielmehr ist vor der Einführung und Anwendung eines Vorgehensmodells eine explizite Anpassung an die Rahmenbedingungen und Ziele der Organisation notwendig. Der Anpassungsprozess erstreckt sich hierbei über mehrere Stufen.

Ausgangspunkt ist ein *generisches Vorgehensmodell*. Ein generisches Vorgehensmodell definiert über Meilensteine, Phasen, Produkte, Aktivitäten und Rollen ein allgemeines Vorgehen zur strukturierten, risikominimierten und qualitätsgesicherten Projektdurchführung. Es ist in seinen Inhalten unabhängig von einem spezifischen Einsatzkontext, beispielsweise einer Organisation oder einem Projekt. So kennt ein generisches Vorgehensmodell keine organisationsspezifischen Produkte oder Rollen und berücksichtigt in seinen Prozessbeschreibungen keine sonstigen organisationsspezifischen Besonderheiten.

Im Rahmen der *organisationsspezifischen Anpassung* wird ein generisches Vorgehensmodell auf die Belange einer konkreten Organisation bzw. Organisationseinheit angepasst. Das Vorgehensmodell wird hierzu bezüglich seiner Produkte, Aktivitäten und Rollen modifiziert, das Ablaufmodell auf der Basis von Meilensteinen und Phasen wird an die jeweiligen organisationsspezifischen Projektabläufe angepasst. Gegebenenfalls wird das Vorgehensmodell um Methoden erweitert. Ergebnis des Anpassungsprozesses ist ein *organisationsspezifisches Vorgehensmodell*. Ein organisationsspezifisches Vorgehensmodell definiert einheitliche Vorgaben und Richtlinien für alle innerhalb der jeweiligen Organisation durchzuführenden Projekte.

Um seine Allgemeingültigkeit und seinen Charakter als Leitfaden und Richtschnur zu erhalten, abstrahiert ein organisationsspezifisches Vorgehensmodell von den individuellen Eigenheiten einzelner Projekte und konzentriert sich auf die Festlegung verpflichtend einzuhaltende Vorgaben und

Richtlinien für bestimmte Projekttypen. Der Abstraktionsgrad eines Vorgehensmodells kann hierbei von Organisation zu Organisation variieren. Ein organisationsspezifisches Vorgehensmodell mit hohem Abstraktionsgrad kann beispielsweise Vorgaben für einen Projekttyp 'Softwareentwicklungsprojekt' definieren, der alle Projekte einer Organisation zur Softwareentwicklung repräsentiert, ohne Berücksichtigung der Art des zu entwickelnden Softwaresystems. Ein organisationsspezifisches Vorgehensmodell mit geringerem Abstraktionsgrad könnte dagegen zusätzlich zwischen dem Projekttyp 'Client-Server Entwicklungsprojekt' und dem Projekttyp 'Entwicklungsprojekt am Mainframe' unterscheiden. Welcher Grad der Abstraktion gewählt wird, hängt von individuellen Gegebenheiten innerhalb der Organisation ab, beispielsweise von Vorkenntnissen und Erfahrungen der Entwickler oder von prinzipiellen Unterschieden bei Projektabläufen.

Ein organisationsspezifisches Vorgehensmodell steht den Projekten der Organisation zur Anwendung zur Verfügung. Mit Initialisierung eines konkreten Projekts wird das organisationsspezifische Vorgehensmodell auf die Eigenschaften dieses Projekts zugeschnitten. Ergebnis ist ein für das Projekt möglichst gut angepasstes (zugeschnittenes) *projektspezifisches Vorgehensmodell*.

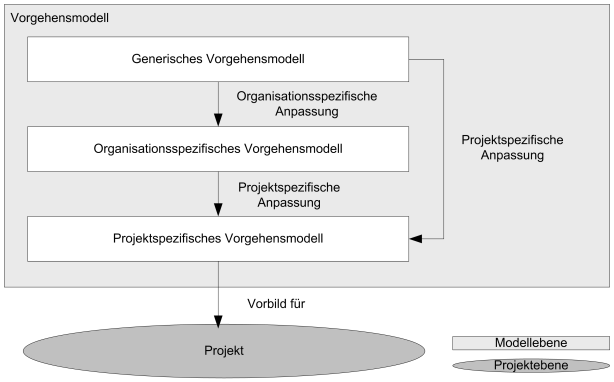

Abbildung 1.1: Anpassungs- und Instanziierungsprozess für Vorgehensmodellen

Abbildung 1.1 beschreibt den Anpassungsprozess ausgehend von einem generischen Vorgehensmodell hin zu einem projektspezifischen Vorgehensmodell. Ein generisches Vorgehensmodell wird im Rahmen der organisationsspezifischen Anpassung auf eine konkrete Organisation angepasst, ein organisationsspezifisches Vorgehensmodell wird im Rahmen der projektspezifischen Anpassung auf ein konkretes Projekt angepasst. Möglich ist auch die projektspezifische Anpassung direkt auf Basis des generischen Vorgehensmodells. Das resultierende projektspezifische Vorgehensmodell dient als Modell für konkrete Projekte.

Generische, organisationsspezifische und projektspezifische Vorgehensmodelle sind Modelle mit unterschiedlicher Zielrichtung. Ein generisches Vorgehensmodell modelliert die Projektabwicklung für eine Menge von Organisationen mit ähnlicher Projektlandschaft, ein organisationsspezifisches Vorgehensmodell modelliert die Projektabwicklung für alle Projekttypen innerhalb einer Organisation, ein projektspezifisches Vorgehensmodell modelliert die Durchführung für eine Menge ähnlicher Pro-

jekte innerhalb einer Organisation. Seine Anwendung findet ein Vorgehensmodell immer im Rahmen eines konkreten Projekts.

1.2 Problemstellung

Damit ein Vorgehensmodell in einem konkreten Projekt sinnvoll anwendbar ist, ist eine Anpassung an den jeweiligen Projektkontext erforderlich. Die Anpassung kann entsprechend Abbildung 1.1 in einem zweistufigen Prozess ausgehend von einem generischen über ein organisationsspezifisches hin zu einem projektspezifischen Vorgehensmodell erfolgen oder alternativ in einem einstufigen Prozess der projektspezifischen Anpassung aus einem generischen Vorgehensmodell. Die Techniken zur organisationsspezifische Anpassung und projektspezifische Anpassung unterscheiden sich dabei jedoch grundsätzlich.

Im Rahmen der *organisationsspezifischen Anpassung* wird das generische Vorgehensmodell substantiell verändert. Dabei können vier Arten von Anpassungen unterschieden werden:

- Addition: Es werden neue Elemente im Vorgehensmodell eingeführt.
- Modifikation: Elemente eines Vorgehensmodells werden in ihren Inhalten verändert.
- Detaillierung: Elemente eines Vorgehensmodells werden in ihren Inhalten erweitert/verfeinert.
- Reduktion: Elemente eines Vorgehensmodells werden entfernt.

Ziel der *projektspezifischen Anpassung* ist dagegen die Submodellbildung. Es wird ein in sich vollständiges und konsistentes Submodell des (organisationsspezifischen oder generischen) Vorgehensmodells gebildet. Das Submodell entspricht selbst wieder einem vollständigen Vorgehensmodell für einen spezifischen Typ von Projekten. Nicht alle Vorgehensmodelle eignen sich in gleicher Weise für eine Anpassung. Zu unterscheiden sind hier insbesondere Vorgehensmodelle, die explizit um Methoden erweitert wurden und methodenneutrale Vorgehensmodelle.

Wird zu einem Vorgehensmodell eine Methodik angegeben, ist diese nahtlos in das Vorgehensmodell integriert. Wir bezeichnen diese Form von Vorgehensmodellen im Folgenden als methodenspezifische Vorgehensmodelle. Ziel der methodenspezifischen Vorgehensmodelle ist eine möglichst gute Projektunterstützung durch eine speziell auf das Vorgehensmodell angepasste Methodik. Methodenspezifische Vorgehensmodelle sind auf die Unterstützung von Projekten mit bestimmten Charakteristiken ausgerichtet, wie beispielsweise die Durchführung von Softwareentwicklungsprojekten nach dem objektorientierten Paradigma. Diese Spezialisierung verbessert einerseits die Anwendbarkeit des Vorgehensmodells, schränkt andererseits jedoch den Anwendungsbereich des Vorgehensmodells stark ein.

Methodenspezifische Vorgehensmodelle weisen zusätzlich noch ein weiteres Problem auf: Da die Methodik auf die verschiedenen Elemente (speziell Produkte und Aktivitäten) im Vorgehensmodell

verteilt ist, entsteht eine Abhängigkeitsstruktur im Vorgehensmodell mit hohem Vernetzungsgrad. Dies führt in der Regel zu stark monolithischen Strukturen, die eine flexible Anpassung des Vorgehensmodells auf unterschiedliche Kontexte erschweren, wenn nicht unmöglich machen. Eine Reduktion oder Modifikation von Elementen, wie sie im Rahmen der organisationsspezifischen Anpassung erforderlich ist, sowie eine Submodellbildung für die projektspezifische Anpassung führen schnell zu Inkonsistenzen und Lücken in der vom Vorgehensmodell vorgegebenen Methodik: Elemente, die entfernt werden, fehlen gegebenenfalls an anderer Stelle, Änderungen an Elementen verursachen Seiteneffekte und führen zu unvorhergesehenen Inkonsistenzen mit den Inhalten anderer Elemente. Typische methodenspezifische Vorgehensmodells sind beispielsweise alle Vorgehensmodelle, deren Wurzeln im Unified Process [69] liegen (z.B. [77], [126], [119]).

Methodenneutrale Vorgehensmodelle haben dagegen die Projektorganisation im Fokus und bleiben hinsichtlich der anzuwendenden Methodik offen. Sie sind gerade auf Grund ihrer Methodenunabhängigkeit für eine Vielzahl teilweise auch stark unterschiedlicher Projekte anwendbar und weisen eine hohe Flexibilität hinsichtlich ihrer Anpassbarkeit auf. Kennzeichen und in gewisser Weise Nachteil der methodenneutralen Vorgehensmodelle ist dagegen die mangelnde methodische Unterstützung für Projekte. Die Wahl geeigneter Methoden für ein Projekt erfolgt unabhängig vom Vorgehensmodell und liegt in den Händen des jeweiligen Projektleiters. Eine freie Methodenwahl bietet zwar maximale Flexibilität für die Methodenauswahl in den Projekten, es ist jedoch nicht gewährleistet, dass die im Rahmen der gewählte Methoden erarbeiteten Ergebnisse mit den vom Vorgehensmodell geforderten Ergebnissen zusammenpassen. Dies führt nicht selten zur (weitgehend als sinnlos erachteten) Nachdokumentation der geforderten Vorgehensmodellergebnisse parallel zur Erarbeitung der tatsächlichen Methodenergebnisse. Auch ist durch fehlende Vorgaben kein einheitliches methodisches Vorgehen in den Projekten möglich. Als methodenneutrale Vorgehensmodelle können beispielsweise die verschiedenen V-Modell Varianten ([39], [79]) oder der Open Prozess [53] eingeordnet werden.

Eine Organisation, die sich entschließt ein Vorgehensmodell einzuführen, steht somit vor dem Problem, ob sie ein methodenspezifisches Vorgehensmodell wählt und damit hinsichtlich der Anpassbarkeit stark eingeschränkt ist, oder ob sie ein methodenneutrales Vorgehensmodell wählt und gegebenenfalls erheblichen Aufwand in eine methodische Anpassung investieren muss. Keines der bekannten Vorgehensmodelle bietet bisher hinsichtlich der genannten Problemstellung eine befriedigende Lösung. Für Organisationen bleibt letztlich die Wahl des kleineren Übels: mangelnde Flexibilität bei der Anpassung versus mangelnde Methodikunterstützung.

1.3 Beitrag der Arbeit

Beitrag dieser Arbeit ist die Konzeption und prototypische Erweiterung von Vorgehens-Metamodellen um das Metamodell einer Methodenbibliothek mit einem Metamodell der Integration. So erweiterte Vorgehens-Metamodelle - im Weiteren Integrations-Metamodell genannt -

erlauben die Entwicklung von Vorgehensmodellen - im Weiteren als Integrationsmodelle bezeichnet -, mit dynamisch integrierbaren Methoden. Als dynamisch integrierbar bezeichnen wir in diesem Zusammenhang die alternative Zuordnung von Methoden zu Produkten und Aktivitäten im Vorgehensmodell, mit der Möglichkeit zur projektspezifischen Methodenauswahl. Damit unterstützen die aus dem Integrations-Metamodell abgeleiteten Integrationsmodelle einerseits die volle Flexibilität der methodenneutralen Vorgehensmodelle hinsichtlich Anpassbarkeit auf unterschiedliche Einsatzkontexte und können andererseits ideal um Methoden ergänzt werden. Sie kombinieren somit in geeigneter Form die Vorteile von methodenneutralen Vorgehensmodellen mit den Vorteilen methodenspezifischer Vorgehensmodelle.

Lösungsidee ist die explizite Trennung der Konzepte Methode und Vorgehensmodell, die Formalisierung von Methode und Schnittstelle und die Einführung eines vom Vorgehensmodell unabhängigen Anpassungsprozesses für Methoden. Methoden und Vorgehensmodell werden erst mit Ende des Anpassungsprozesses zusammengeführt. Damit bleibt die Flexibilität des Vorgehensmodells bei der Anpassung erhalten, das angepasste Vorgehensmodell kann jedoch beliebig methodisch erweitert werden.

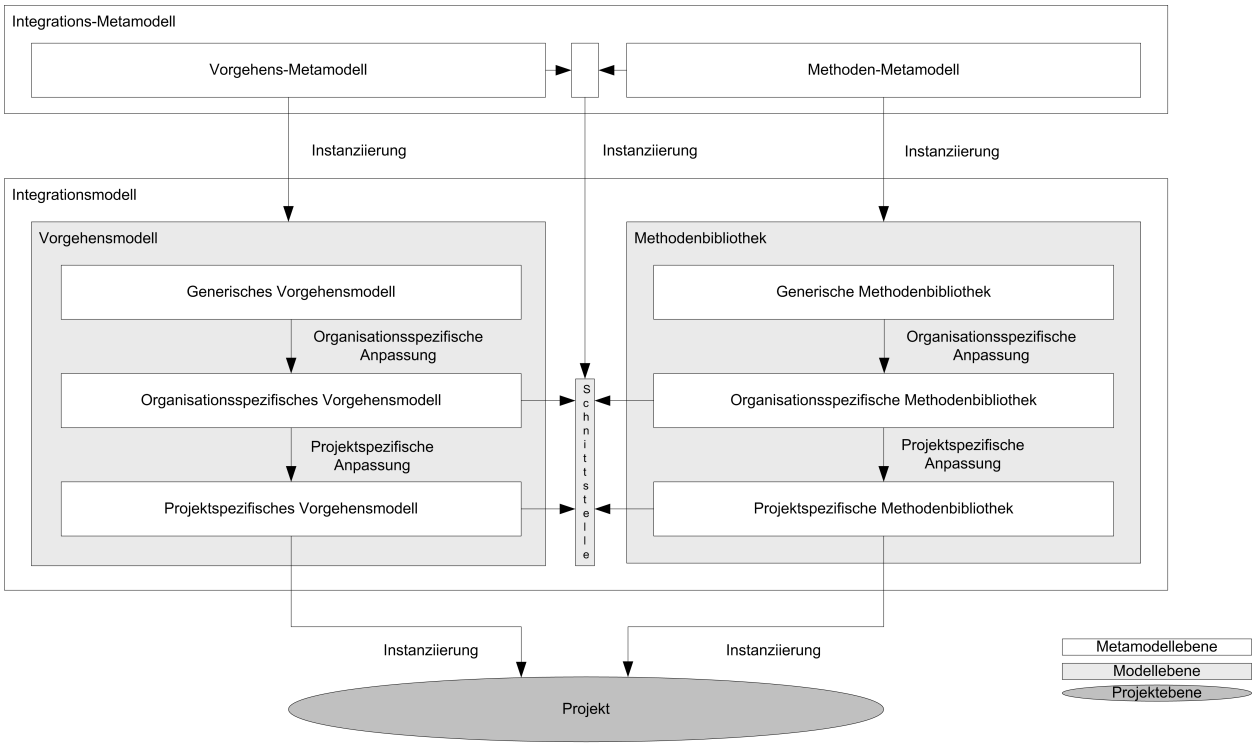

Abbildung 1.2: Anpassungs- und Instanziierungsprozess für Integrationsmodellen auf Basis des Integrations-Metamodells

Abbildung 1.2 stellt Integration-Metamodell und Integrationsmodell eingebettet in den erweiterten Anpassungs- und Instanziierungsprozess dar. Das Integrations-Metamodell umfasst drei Teil-Metamodelle: ein Metamodell zur Entwicklung eines methodenneutralen Vorgehensmodells, ein

Metamodell zur Entwicklung einer Methodenbibliothek und ein Metamodell für die geeignete Integration von Methoden der Bibliothek in das Vorgehensmodell.

Instanzen eines Integrations-Metamodells sind Integrationsmodelle bestehend aus einem methodenneutralen Vorgehensmodell, einer Methodenbibliothek und einer expliziten Zuordnung von Methodenergebnissen zu Elementen im Vorgehensmodell. Wie im normalen Anpassungsprozess (vgl. Abbildung 1.1) durchläuft das Vorgehensmodell im Integrationsmodell die zwei Stufen der organisationsspezifischen und projektspezifischen Anpassung. Parallel zur organisationsspezifischen Anpassung des Vorgehensmodells wird die Methodenbibliothek angepasst, d.h. Methoden werden bei Bedarf entwickelt, inhaltlich überarbeitet oder ergänzt. Außerdem erfolgt eine Zuordnung der Methoden zum Vorgehensmodell.

Ergebnis des ersten Anpassungsschritts ist ein organisationsspezifisch angepasstes Integrationsmodell, mit getrenntem Methoden- und Vorgehensmodellanteil jedoch einer klaren, gegebenenfalls alternativen Zuordnung von Methoden zum Vorgehensmodell. Im Rahmen der projektspezifischen Anpassung passt der Projektleiter das Vorgehensmodell an die Bedürfnisse seines Projekts an und wählt aus den zugeordneten Methoden die für seine Projektsituation geeigneten aus. Ergebnis ist ein projektspezifisch angepasstes Vorgehensmodell, erweitert um eine Menge von Methoden. Vorteil dieses Ansatzes ist die starke Entkopplung von Vorgehensmodell und Methoden im Integrationsmodell. Dies ermöglicht letztlich die hohe Flexibilität bei der Anpassung.

Aus wissenschaftlicher Sicht beschäftigt sich diese Arbeit mit der Fragestellung, was unter einer Methode zu verstehen ist und wie das Konzept einer Methode formal beschrieben werden kann. Im Gegensatz zu Vorgehensmodellen gibt es bisher nur wenig Arbeiten, die sich mit der Frage beschäftigen, was eine Methode konkret kennzeichnet, welche Konzepte als Methode bezeichnet werden können und wie Methoden sich zu verwandten Konzepten abgrenzen. Ein zentraler Beitrag dieser Arbeit ist daher die Entwicklung einer allgemeinen Charakterisierung von Methoden sowie die Ableitung einer Methodenontologie mit einer umfassenden Methodendefinition. Ontologie und Definition bildet die Basis für die Entwicklung eines formalen Methoden-Metamodells, das inhaltliche Eigenschaften von Methoden untersucht und als Vorbild für das semi-formale Metamodell der Methodenbibliothek in Integrations-Metamodellen dient. Ein weiteres Ergebnis ist die formale Erfassung der Integration von Methoden in Vorgehensmodelle. So wird untersucht und formal beschrieben, in welcher Form Vorgehensmodelle Anforderungen an potentiell zu integrierende Methoden formulieren und wie sichergestellt werden kann, dass Methoden diese Anforderungen tatsächlich erfüllen.

1.4 Einordnung und Abgrenzung

Diese Arbeit kann im Bereich der Detaillierung bzw. Erweiterung von Vorgehensmodellen durch Methoden eingeordnet werden. Ziel ist die Verbesserung der Anwendbarkeit des Vorgehensmodells

durch eine gute methodische Unterstützung unter Erhaltung der Flexibilität bei der Anpassung. Das Problem, Methoden und Vorgehensmodell geeignet zu verbinden ist nicht neu und hat letztlich zu zwei Lösungsansätzen geführt: Entwicklungsmethoden auf der einen Seite und Vorgehensmodelle mit unterschiedlichen Ansätzen zur Methodenintegration auf der anderen. Entwicklungsmethoden stellen die Methodik zu Systementwurf und -entwicklung in den Vordergrund. Projektspezifische Managementaspekte werden dabei nur am Rande berücksichtigt. Die Methodik ist eingebettet in ein rudimentäres Vorgehensmodell, das einen groben Rahmen für die Erarbeitung der methodischen Ergebnisse definiert. Entwicklungsmethoden, wie beispielsweise OOAD [20], Structured Analysis [33], Object-oriented Analysis [29] und Object-oriented Design [30], sind gerade auf Grund ihrer engen Kopplung mit einer spezifischen Methodik in der Regel nicht als organisationsweite Vorgehensmodelle einsetzbar. Sie werden vielmehr zur Unterstützung konkreter Projekte verwendet, beispielsweise im Rahmen eines übergeordneten Vorgehensmodells. Sie sind nicht bzw. nur minimal an unterschiedliche Kontexte anpassbar.

Vorgehensmodelle als Modelle der Projektabwicklung fokussieren dagegen auf die Unterstützung der Projektorganisation. Zur Integration der Methoden bieten Vorgehensmodelle unterschiedliche Ansätze an. Beispielsweise ermöglichen einige Vorgehensmodelle durch einen modularen Aufbau die Austauschbarkeit von Teilmodellen. So unterstützen der Rational Unified Process [77] und der OpenUp Prozess [119] ein 'Plug-In Konzept'. Einzelne Teilmodelle im Vorgehensmodell (Plug-Ins) können ausgetauscht werden und damit gegebenenfalls eine neue Methodik eingeführt werden. Plug-Ins modellieren jedoch nicht ausschließlich Methodik sondern sind für sich genommen eigenständige Teilmodelle eines Vorgehensmodell. Der Plug-In Ansatz erlaubt zwar eine dynamische Integration von Methoden, die Vorgehensmodelle bleiben jedoch unflexibel hinsichtlich der Anpassung.

Andere Vorgehensmodelle geben keine explizite Methodik vor, sie geben jedoch Hinweise auf geeignete Methoden, wobei die Anwendung nicht verbindlich vorgegeben wird. Eine entsprechende Variante der losen Kopplung von Vorgehensmodell und Methoden findet sich beispielsweise im OPEN Process Framework [44]. Das Metamodell des OPF unterstützt explizit die Zuordnung von Notationen (*Languages*) zu Produkten (*Work Products*) und von Techniken (*Techniques*) zu Aktivitäten (*Work Units*). Diese Form der Unterstützung eignet sich für einfache methodische Hinweise, eine durchgängige methodische Unterstützung ist jedoch nicht möglich.

Eine ähnliche Strategie verfolgt das V-Modell XT [79], [118]. Hier werden über *Methodenreferenzen* Hinweise auf Methodentypen gegeben, die die Durchführung einzelner *Aktivitäten* geeignet unterstützen. Im Rahmen der organisationsspezifischen Anpassung werden den Aktivitäten konkrete Methoden zugeordnet. Diese Methodenbeschreibung ist jedoch hochgradig unstrukturiert. Auch fehlt ein direkter Bezug der Methodenergebnisse zu Produkten bzw. Themen im V-Modell.

Einen alternativen Ansatz zur Integration von Methodik in ein Vorgehensmodell unterstützen Gnatz et al. mit ihrem Konzept für einen 'lebendigen' Prozess [50]. Der Ansatz erlaubt die dynamische Zuordnung von *Process Patterns* zu *Process Artefacts*. Process Patterns sind Prozessbeschreibungen

in einem informellen, jedoch fest strukturierten Pattern Format. Ein Pattern bietet eine methodische Beschreibung zur Durchführung eines Process Artefacts. Bei Process Patterns handelt es sich bisher jedoch um einen rein akademischen Ansatz, der für eine Anwendung in der Industrie noch nicht ausreichend konkret ist.

Für einen detaillierten Vergleich der hier kurz vorgestellten Ansätze mit dem in dieser Arbeit entwickelten Ansatz zur Erweiterung von Vorgehens-Metamodellen zu Integrations-Metamodellen wird auf Kapitel 6.8 verwiesen. Dort werden explizit die Unterschiede in den Metamodellen der Ansätze und die jeweiligen Auswirkungen auf die Flexibilität hinsichtlich der Anpassung bei der Integration von Methoden dargestellt. Insbesondere wird gezeigt, in welcher Form der hier entwickelte Ansatz sich von den bestehenden Ansätzen abgrenzt.

1.5 Aufbau der Arbeit

Diese Arbeit beschäftigt sich im weitesten Sinn mit Modellen und der Modellierung von Prozessen. Kapitel 2 führt in die Grundlagen der Modellierung ein und definiert die Begriffe *Modell* und *Metamodell*. Des Weiteren wird die MOF Metadatenarchitektur als ein häufig genutzter Ansatz zur Entwicklung von Metamodellen vorgestellt. So dient die MOF auch in dieser Arbeit als Grundlage der Metamodellentwicklung. Zur Beschreibung der Modelle auf den verschiedenen Ebenen der Metadatenarchitektur wird jedoch nicht die MOF Modellierungssprache selbst verwendet. Vielmehr greift diese Arbeit auf eine von Gnatz entwickelte alternative graphische Modellierungssprache zurück (vgl. [49]), die auf der Modellierungssprache der Meta Object Facility aufsetzt, diese jedoch hinsichtlich der Semantik zur Modellinstanziierung erweitert und konkretisiert. Diese Sprache wird mit leichten Änderungen an der konkreten Syntax ebenfalls in diesem Kapitel vorgestellt.

Kapitel 3 führt allgemein in das Thema Vorgehensmodelle ein. Der Modellcharakter von Vorgehensmodellen wird motiviert und es werden die wesentlichen Konzepte eines Vorgehensmodells vorgestellt. Vorgehensmodelle treten in unterschiedlichen Formen auf. Diese werden diskutiert und zu verwandten Konzepten abgegrenzt. Im Weiteren wird der Einsatz von Vorgehensmodellen in Organisationen beispielsweise zur Prozessverbesserung und zur Prozessplanung motiviert. Neben Vorgehensmodellen gibt es eine Reihe weiterer Modelle mit dem Ziel der Prozessverbesserung. Zu nennen sind hier insbesondere Qualitätsmanagementmodelle, Reifegradmodelle oder Assessmentmodelle. Diese Modelltypen werden vorgestellt und ihre Anwendung zur Prozessverbesserung im Zusammenhang gezeigt.

Eine intensive Auseinandersetzung mit dem Thema Methoden erfolgt in Kapitel 4. Anhand repräsentativer Beispielmethoden werden allgemeine Eigenschaften von Methoden identifiziert. Kernkonzepte und Einflussfaktoren von Methoden werden im Detail diskutiert und darauf aufbauend eine Methodenontologie sowie eine allgemeine Methodendefinition abgeleitet.

Die informelle Diskussion zu Methoden dient als Grundlage für eine formale Spezifikation des Methodenkonzepts in Kapitel 5. Neben einer Formalisierung von Methoden beschäftigt sich dieses Kapitel insbesondere mit der Abgrenzung von Vorgehensmodellen und Methoden sowie einer formalen Spezifikation der Schnittstelle. Es wird gezeigt, wie Methoden in Vorgehensmodelle eingebettet sind, in welcher Form Vorgehensmodelle ihre Anforderungen an potentiell integrierbare Methoden formulieren und wie sichergestellt wird, dass Methoden diese Anforderungen tatsächlich erfüllen.

Basierend auf der formalen Beschreibung der Methodenintegration wird in Kapitel 6 ein Metamodell entwickelt, das produktorientierte Vorgehens-Metamodellen zu Integration-Metamodellen erweitert. Das resultierende Metamodell erlaubt die Ableitung von Vorgehensmodellen mit dynamisch integrierbaren Methoden. Das Metamodell umfasst einerseits das Metamodell einer Methodenbibliothek zur Entwicklung und Verwaltung von Methoden, andererseits modelliert es eine allgemeine Schnittstelle zur Zuordnung der Methoden zu Produkten und Aktivitäten im Vorgehensmodell. Zur Sicherstellung der Konsistenz werden jeweils die relevanten Konsistenzbedingungen definiert. Das Metamodell der Methodenbibliothek wird anhand einer Beispielbibliothek mit verschiedenen Methoden evaluiert.

Integrationsmodelle als Instanzen des Integrations-Metamodells durchlaufen ähnlich zu Vorgehensmodellen einen Lebenszyklus. In ihrem Lebenszyklus durchlaufen sie unterschiedliche Phasen von der Anforderungsanalyse über Entwurf und Entwicklung bis hin zu Anwendung und Analyse. Ein entsprechendes Lebenszyklusmodell, das insbesondere die Trennung von Standard und organisationsspezifischen Varianten berücksichtigt, wird in Kapitel 7 vorgestellt. Wie das Modell zeigt, muss zusätzlich zum normalen Lebenszyklus eines Vorgehensmodells im Lebenszyklus von Integrationsmodellen auch die Entwicklung, Anpassung und Zuordnung der Methoden berücksichtigt werden.

Die Struktur der Schnittstelle zwischen Methodenbibliothek und Vorgehensmodell hängt stark von der Produkt- und Aktivitätsstruktur des gewählten Vorgehens-Metamodells im Integrations-Metamodell ab. Die allgemeine Schnittstelle muss zur konkreten Anwendung auf die jeweiligen strukturellen Anforderungen des Vorgehens-Metamodells angepasst werden. Kapitel 8 zeigt abschließend am Beispiel des V-Modells XT wie eine entsprechende Anpassung aussehen könnte. Das Metamodell der Schnittstelle wird auf die Struktur des V-Modells angepasst, der Mechanismus zur Methodenauswahl wird in das Tailoringmodell des V-Modells integriert. Abschließend wird gezeigt, wie eine konkrete Umsetzung des hier entwickelten Ansatzes auf Basis der für das V-Modell verwendeten Technologien und Werkzeuge aussehen könnte.

Kapitel 9 fasst die Ergebnisse der Arbeit noch einmal im Überblick zusammen und zeigt Vor- aber auch Nachteile des hier erarbeiteten Ansatzes zur Methodenintegration. Es werden insbesondere auch Anwendungsszenarien für die Ergebnisse vorgestellt, die über das Ziel der hier adressierten Methodenintegration in Vorgehensmodelle hinausgehen.

Kapitel 2

Grundlagen der Modellierung

Ziel dieser Arbeit ist die Entwicklung eines Metamodells für Vorgehensmodelle mit der Möglichkeit zur flexiblen Auswahl und Integration von Methoden. Dieses Kapitel geht auf die Grundlagen der Modellierung im Allgemeinen und die Entwicklung von Metamodellen auf Basis einer Metadatenarchitektur im Besonderen ein.

Das Kapitel ist wie folgt aufgebaut: Abschnitt 2.1 beschäftigt sich mit den Grundlagen der Modellierung. Die Begriffe Modell und Metamodell werden eingeführt und definiert und die Rolle von Metamodellen als Modellierungssprachen für Modelle gezeigt. Zur Entwicklung des Metamodells stützt sich diese Arbeit auf die Meta Object Facility (MOF) der OMG. Die MOF ist die Spezifikation einer Metadatenarchitektur, die die Entwicklung, Verwaltung und Transformation domänenspezifischer Modelle unterstützt. In Abschnitt 2.2 werden die Grundprinzipien der MOF vorgestellt. Kern der MOF sind eine Metadatenarchitektur mit beliebig vielen Metaebenen, sowie eine UML-nahe Notation zur Modellentwicklung auf den verschiedenen Metaebenen der Architektur. Wie gezeigt wird, weist die MOF Notation Defizite hinsichtlich der Modellbeziehungen zwischen den Metaebenen auf. So gibt es keine klare Semantik der Modellinstanziierung über mehrere Metaebenen hinweg. Zur Lösung des Problems wird in dieser Arbeit eine alternative Sprache verwendet, die diesen Defiziten geeignet begegnet. Diese Sprache wird in Abschnitt 2.3 vorgestellt. Die Ergebnisse des Kapitels werden in Abschnitt 2.4 zusammengefasst.

2.1 Modelle und Modellbildung

Modelle und Modellbildung spielen in der Informatik eine zentrale Rolle. Nach [108] sind Modelle Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können. Modelle sind Abstraktionen. Sie erfassen nicht alle Eigenschaften des Originals sondern nur solche, die den Modellbeschaffern oder den Modellnutzern relevant erscheinen. Ziel der Modellbildung ist es, einen Gegenstandsbereich (das Original) in einem Modell abzubilden. Die Modellbildung erfolgt durch Äquivalenzklassenbildung und Strukturierung (vgl. [49]).

Zur Entwicklung eines Modells braucht es eine geeignete Modellierungssprache. Eine Sprache ist allgemein ein System von Zeichen und Regeln zur Verwendung dieser Zeichen (vgl. [109]). Die Zeichen geben das Alphabet der Sprache vor. Das Alphabet einer graphischen Modellierungssprache entspricht der Menge graphischer Konstrukte, die zur Darstellung von Modellen verwendet werden können. Durch Anwendung von Regeln über den graphischen Konstrukten entstehen gültige Modelle. Für graphische Modellierungssprachen fassen wir daher den Sprachbegriff enger. Eine graphische Modellierungssprache beschreibt die zur Modellbildung notwendigen Modellelemente, sowie Regeln für gültige Verknüpfungen der Modellelemente.

Ist eine Modellierungssprache selbst Gegenstand der Modellbildung, spricht man von einem Metamodell. Ein Metamodell ist das Modell einer Modellierungssprache eines Modells, auch Objektmodell genannt. Dieses Konzept der Hierarchie von Sprachebenen wird auch als Metaisierung von Sprachen bezeichnet und stammt aus der Sprachstufentheorie. Es kann rekursiv über beliebig viele Metaebenen fortgeführt werden. So definiert ein Metamodell die Sprache eines Objektmodells, kann jedoch selbst wieder Objektmodell zu einem Metametamodell sein (vgl. [109]).

Metamodelle definieren eine Sprache zur Modellbildung. Ein anhand eines Metamodells entwickeltes Modell ist wohlgeformt, wenn es der abstrakten Syntax des Metamodells vollständig entspricht, d.h. wenn der Aufbau des Modells vollständig den von der Modellierungssprache vorgegebenen Verknüpfungsregeln folgt. Wohlgeformtheit berücksichtigt daher den syntaktischen Aufbau. Die Konsistenz der entwickelten Modelle wird durch eine Menge von Konsistenzbedingungen sichergestellt. Konsistenz eines Modells bedeutet somit Widerspruchsfreiheit. Ein konsistentes Modell ist in sich stimmig und enthält keine Widersprüche. Ein Modell ist dann konsistent, wenn es den Konsistenzbedingungen zur Modellbildung genügt.

Im folgenden Abschnitt wird eine der aktuell bekanntesten Metadatenarchitekturen vorgestellt, die dieses Prinzip der Metaisierung zur Entwicklung von Modellierungssprachen unterstützt, die Meta Object Facility.

2.2 Die Meta Object Facility

Die Meta Object Facility (MOF) ist die Spezifikation eines Frameworks zur Entwicklung objektorientierter Metamodelle. Entwickelt wurde die Meta Object Facility von der Object Management Group ([127]), die auch für die Weiterentwicklung der Spezifikation verantwortlich ist. Zentraler Bestandteil der MOF ist das Konzept einer Metadatenarchitektur sowie eine einfache objektorientierte graphische Modellierungssprache zur Entwicklung von Metamodellen, das MOF Model. Abbildung 2.1 zeigt die Metadatenarchitektur der MOF sowie die beispielhafte Einordnung von Modelltypen auf den verschiedenen Metaebenen.

Die MOF geht von einer vier-Ebenen Architektur aus, wobei bei Bedarf die Erweiterung um beliebig viele Zwischenebenen möglich ist. Modelle einer Ebene sind Instanzen von Modellen der darüber

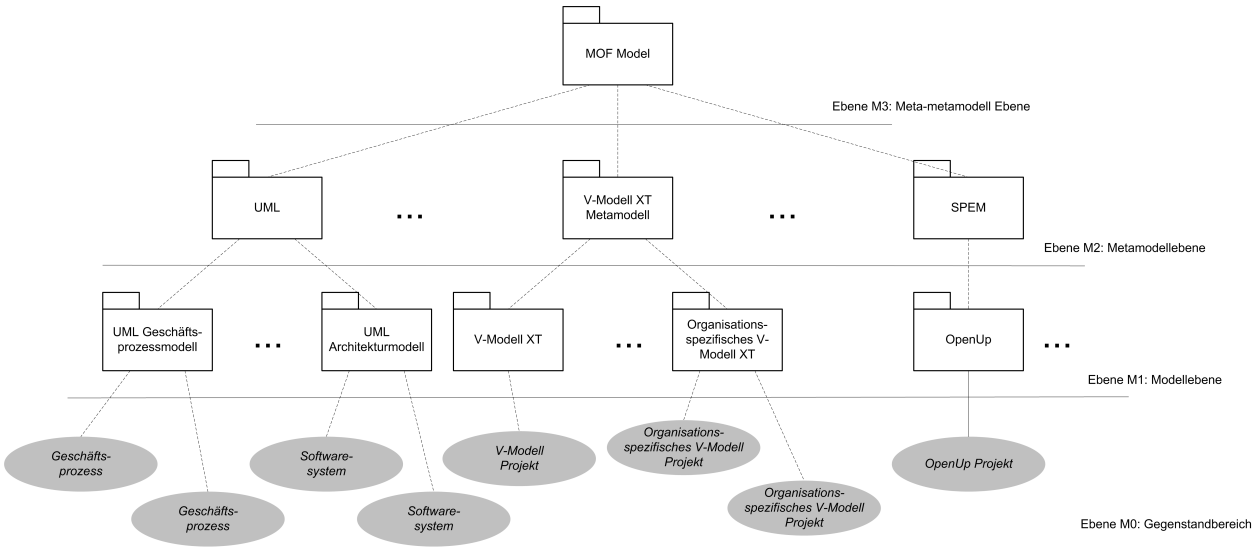

Abbildung 2.1: Die MOF Schichtenarchitektur

liegenden Ebene. Gleichzeitig sind sie Metamodell für Modelle der darunter liegenden Ebene. Die Ebenen werden beginnend mit der untersten Ebene M0, dem Gegenstandsbereich, aufsteigend bis zur Ebene M3 nummeriert. Das MOF Model ist auf der obersten Metaebene M3 der MOF Metadatenarchitektur angesiedelt. Es definiert eine Basissprache für die Entwicklung von Metamodellen auf Ebene M2 und wird auch als Metametamodell bezeichnet. Die Metamodelle auf Ebene M2 der MOF Metadatenarchitektur sind Instanzen des MOF Models, d.h. die in den Metamodellen verwendeten Konzepte sind Instanzen der Konzepte des MOF Models. Jedes Metamodell auf Ebene M2 der Metadatenarchitektur definiert eine Sprache für eine Menge von Anwendungsmodellen auf der Ebene M1. Die Anwendungsmodelle sind jeweils Instanzen ihres Metamodells. Anwendungsmodelle modellieren selbst wieder eine Sprache zur Beschreibung konkreter Objekte oder Prozesse ihres Gegenstandsbereichs.

In Abbildung 2.1 sind mehrere Beispiele von Modellhierarchien dargestellt. So liegt beispielsweise auf Ebene M2 die UML 2.0 [86] als Instanz des MOF Models und gleichzeitig als Metamodell von UML Anwendungsmodellen der Ebene M1. Anwendungsmodelle der UML sind alle Arten von Modelle, die mit Hilfe der UML Notation erstellt werden, dies können beispielsweise Architekturmodelle oder Geschäftsprozessmodell sein. Die Beschreibung einer konkreten Umsetzung des Architekturmodells oder des Geschäftsprozessmodells wäre Element der Ebene M0, beispielsweise eine Systemdokumentation oder eine Prozessdokumentation.

Ein weiteres Beispiel eines Metamodell-Standards auf Ebene M2 ist das Software Process Engineering Metamodell (SPEM) [88] der OMG. Die SPEM Spezifikation definiert eine Metamodell zur Entwicklung von Vorgehensmodellen. Instanz des SPEM ist beispielsweise der OpenUp Prozess [119].

Ein Vorteil der MOF ist die Möglichkeit zur Entwicklung eigener, proprietärer Metamodelle. Diese Technik wurde beispielsweise bei der Entwicklung des V-Modells XT angewendet. Das V-Modell XT Metamodell ist ein proprietäres Metamodell und ist wie das SPEM auf Ebene M2 der Metadatenarchitektur angesiedelt. Instanzen des Metamodells sind das V-Modell XT selbst, aber auch alle Varianten des V-Modells wie beispielsweise organisationsspezifische Anpassungen. Die Dokumentation eines konkreten Projekts, das nach dem V-Modell XT oder einer seiner Variante durchgeführt wird, ist Instanz des V-Modells.

Das MOF Model ist im Wesentlichen eine minimale objektorientierte Modellierungssprache. Während frühere Versionen der MOF (bis 1.4) ein eigenständiges MOF Model definierten, stützt sich die aktuelle Version (Version 2.0) weitgehend auf die UML 2.0. Die UML 2.0 Spezifikation umfasst zwei Teile: UML 2.0 Infrastructure [85] und UML 2.0 Superstructure [85]. Ziel der Aufteilung ist die Möglichkeit zur Wiederverwendung von Teilen der UML. Die UML 2.0 Infrastructure definiert den Kern der UML 2.0. Sie umfasst die Basiselemente der Sprache und kann von anderen Spezifikationen wie beispielsweise der MOF wiederverwendet werden. Die UML 2.0 Superstructure erweitert die Basiskonzepte der UML Infrastructure und unterstützt beispielsweise die Definition weiterer UML Modellelemente. Das MOF Model selbst unterscheidet zwischen einem Basismodell (Essential MOF (EMOF)) zur Entwicklung einfacher Metamodelle, sowie einer erweiterten Version, dem Complete MOF Model (CMOF). CMOF bietet einen größeren Sprachumfang an und ist auch zur Entwicklung komplexere Metamodelle, wie beispielsweise zur Definition der UML 2.0, geeignet.

Ein Problem der Metaisierung ist die Beziehung von Modelle über mehrere Metaebenen hinweg. Die MOF geht hier implizit von einer Instanziierungsbeziehung aus: Ein Objektmodell einer Ebene M_{i-1} ist Instanz seines Metamodells der Ebene M_i. Klassen und Assoziationen des Modells der Ebene M_i werden im Modell der Ebene M_{i-1} durch Objekte und Links instanziiert. Attribute der Klassen werden in den Objekten mit konkreten Werten belegt. Gleichzeitig repräsentiert das Metamodell auf der Ebene M_i das Objektmodell eines Modells der Ebene M_{i+1}. Die von der MOF zur Modellbildung verwendete UML kennt jedoch nur ein einstufiges Instanziierungsmodell. Eine Klasse der UML beschreibt ausschließlich die Semantik ihrer unmittelbaren Instanzen. Eine Instanziierung der Instanzen ist dagegen nicht vorgesehen. Das Instanziierungsmodell der UML erlaubt somit keine Aussagen über die Beziehung zwischen einem Modell auf der Ebene M_{i+1} und einem Modell der Ebene M_{i-1}. Eine ausführliche Diskussion zu der hier angesprochenen Problematik findet sich in Abschnitt 2.3.

Während die MOF keine Aussagen hinsichtlich der Instanziierungssemantik macht, gibt es eine Reihe von Autoren, die hier Lösungsansätze in Form von Modellierungssprachen oder Pattern anbieten. An dieser Stelle wird nicht näher auf diese Ansätze eingegangen, sondern auf die entsprechende Literatur verwiesen (vgl. [7], [3]). Die Diskussion dient jedoch als Motivation für die Entscheidung, in dieser Arbeit nicht die MOF selbst als Modellierungssprache zu verwenden, sondern auf eine alternative Sprache zurückzugreifen, die sich in ihrer Syntax an der UML orientiert, jedoch eine

geeignete Semantik hinsichtlich der Instanziierung über mehrere Modellierungsebenen hinweg definiert. Im Folgenden wird diese Sprache, soweit für diese Arbeit erforderlich vorgestellt. An manchen Stellen wurden dabei Ergänzungen vorgenommen.

2.3 Angepasste Modellierungssprache

Als Modellierungstechnik für Metamodelle und Modelle wird in diese Arbeit eine von Gnatz [49] entwickelte graphischen Modellierungssprache verwendet. Die Sprache setzt weitgehend auf den Konzepten des MOF Models auf. Gegenüber dem MOF Model bietet die Sprache von Gnatz mehrere Vorteile: sie stützt sich wie das MOF Modell in der konkreten Syntax vollständig auf die UML ab, ist jedoch in den verfügbaren Konzepten direkt auf die Bedürfnisse der Vorgehensmodellierung zugeschnitten. Zusätzlich unterstützt die Sprache eine geeignete Semantik für die Instanziierungsbeziehung der Modelle zwischen den Metaebenen.

Bei der Definition der Sprache verwendet Gnatz das in der Informatik üblicherweise verwendete dreistufige Muster zur Sprachdefinition (siehe auch Abbildung 2.2) mit den Elementen:

- Abstrakte Syntax: Festlegung der Sprachkonzepte.
- Semantik: Abbildung der Sprachkonzepte auf eine semantische Domäne.
- Konkrete Syntax: Notation zur Darstellung der abstrakten Syntax.

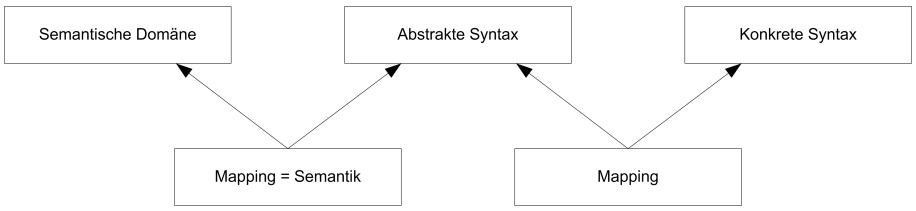

Abbildung 2.2: Muster zur Sprachdefinition

Zur Formulierung von Konsistenzbedingungen verwendet Gnatz eine intuitive Sprache, die auf Konstrukten der Prädikatenlogik aufsetzt. Bei der Spezifikationssprache handelt es sich um eine pseudoformale Sprache. So ist sie weder vollständig formal definiert, noch ist der Sprachumfang festgelegt. Um hier klare Regeln zur Spezifikation der Konsistenzbedingungen zu haben, verwendet diese Arbeit die im Sprachumfang der UML enthaltene Spezifikationssprache, die Objekt Constraint Language (OCL) [86]. In den folgenden Abschnitten werden die Konzepte der Modellierungssprache soweit vorgestellt, wie sie Erweiterungen betreffen bzw. zum Verständnis dieser Arbeit notwendig sind. Für eine vollständige Beschreibung wird auf [49] verwiesen.

Abstrakte Syntax und Semantik

Die abstrakte Syntax der Modellierungssprache definiert die Konzepte: Klasse, Assoziation und Attribut. Jedes dieser Konzepte hat einen eindeutigen Namen als Identifikator. Eine Klasse ist eine Typbeschreibung für eine Menge von Objekten. Sie kann beliebig viele Attribute definieren. Ein Attribut ist ein Platzhalter für einen konkreten Wert. Einem Attribut ist ein eindeutiger Name und ein Datentyp zugeordnet. Wir beschränken den Raum der zulässigen Datentypen auf die Grundtypen: Integer, String und Boolean.

Assoziationen sind Konstrukte zur Beschreibung von Beziehungen zwischen Modellelementen. Eine Assoziation hängt jeweils über ein Assoziationsende an einer Klasse. Eine Assoziation verbindet immer genau zwei Klassen und hat dementsprechend genau zwei Assoziationsenden. Einem Assoziationsende ist ein eindeutiger Name zugeordnet. Des Weiteren definiert ein Assoziationsende die Multiplizität der Assoziation auf Instanzebene. Ihren Wertebereich erhält die Multiplizität über zwei Attribute zur Definition der unteren Grenze bzw. der oberen Grenze. Abbildung 2.3 beschreibt die vollständige abstrakte Syntax der Modellierungssprache. Wie bei der Definition des MOF Models wird auch die Sprache von Gnatz zur Definition der Sprache selbst verwendet.

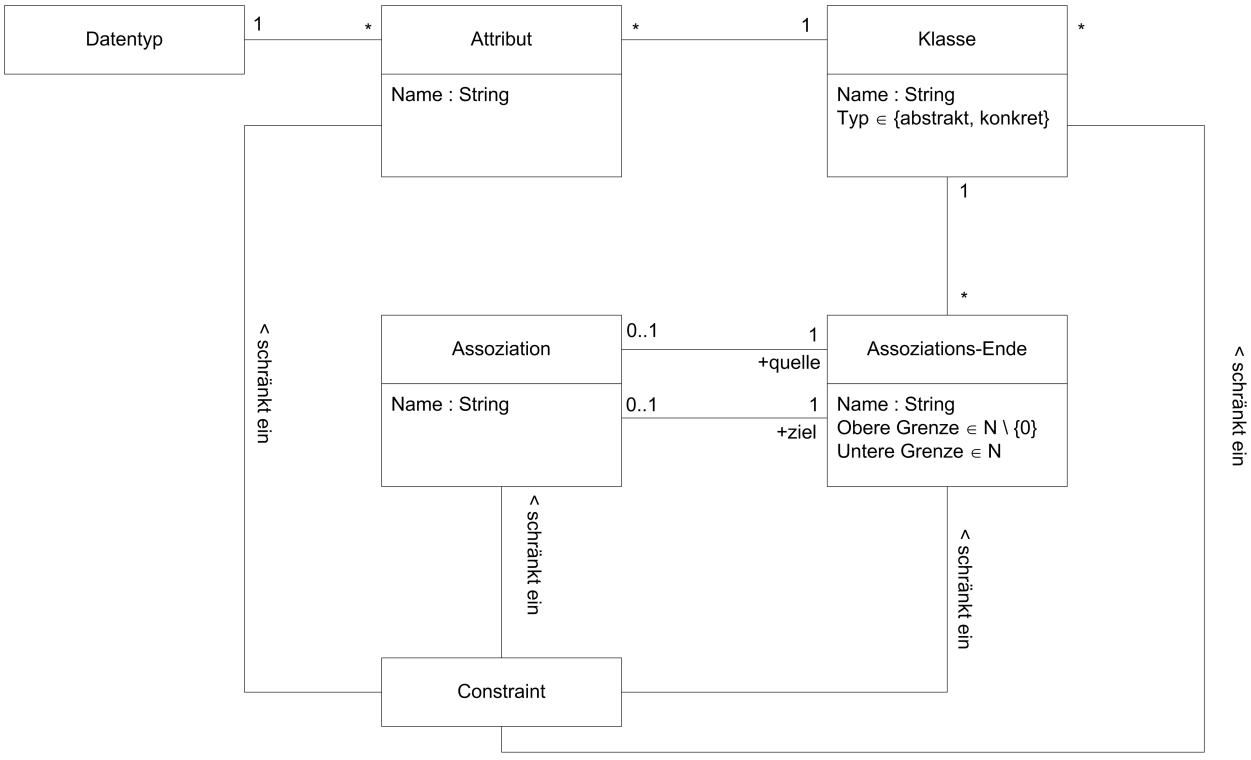

Abbildung 2.3: Abstrakte Syntax der Modellierungssprache (Quelle Gnatz, eigene Darstellung)

Für wohlgeformte Modelle gilt entsprechend der abstrakten Syntax immer, dass Assoziationen mit genau zwei Assoziationsenden verknüpft sind. Ein Assoziationsende entsprecht der Quelle der Assoziation, ein Assoziationsende dem Ziel. Für diese Beziehung muss als zusätzliche Wohlgeformt-

heitsregel gelten, dass Ziel und Quelle der Assoziation nicht identisch sein dürfen. Wir formulieren dies durch folgende Bedingung:

```
context Assoziation inv:
    self.quelle <> self.ziel
```

Die Semantik der abstrakten Syntax wird über ihre Instanzen definiert. Die semantische Domäne der Sprache basiert auf den abstrakten Elementen Objekt und Link. Klassen werden durch Objekte, Assoziationen durch Links, Attribute durch ihre Wertebelegung interpretiert. Die Semantik wird unverändert aus [49] übernommen und hier nicht weiter betrachtet.

Konkrete Syntax

Eine abstrakte Syntax einer Sprache dient vorrangig zur Identifizierung der zu beschreibenden Elemente einer Domäne. Die konkrete Syntax stützt sich auf die abstrakte Syntax und bildet die abstrakten Elemente auf konkrete Beschreibungselemente ab. Abbildung 2.4 zeigt die Abbildung von Elementen der abstrakten Syntax auf Sprachkonstrukte der UML. Die konkrete Syntax orientiert sich an den Sprachkonstrukten der UML.

Die Sprache orientiert sich weitgehend an der Notation der UML Klassendiagramme. Kernelemente sind UML Klassen. Klassen können Attribute zugeordnet werden, diese haben jeweils einen spezifischen Datentypen. Klassen können als abstrakt gekennzeichnet werden. In diesem Fall dürfen keine Instanzen der Klasse auf unteren Modellebenen angelegt werden.

Assoziationen der abstrakten Syntax werden in der konkreten Syntax auf UML Assoziationen und UML Aggregationen abgebildet. Eine Aggregation ist eine Assoziationen mit dem Namen 'enthält'. Die Assoziationsenden werden als *Teil* und *Ganzes* benannt. Zu den Assoziationsenden werden die entsprechenden Multiplizitäten explizit angegeben. Die Multiplizität gibt die Anzahl möglicher Instanzen des Assoziationsendes und damit die Anzahl möglicher Instanzen der zugeordneten Klasse an. Zulässige Werte sind entsprechend der üblichen UML Notation:

- **0..1**: es darf höchstens eine Instanz der Klasse existieren
- **1**: es muss genau eine Instanz existieren
- **n..***: es müssen mindestens n Instanzen existieren, mit $n > 0$
- ***: es dürfen beliebig viele Instanzen existieren

Für Aggregationen können bei Bedarf gleichwertige verkürzende Darstellung (Aggregation II und Aggregation III) verwendet werden, die anzeigen, dass ein Element weitere Elemente enthalten kann. Aggregation II und Aggregation III unterscheiden sich hinsichtlich ihrer Multiplizitäten. Während Aggregation II die Multiplizität * unterstützt, kennzeichnet Aggregation III die Multiplizität 0..1.

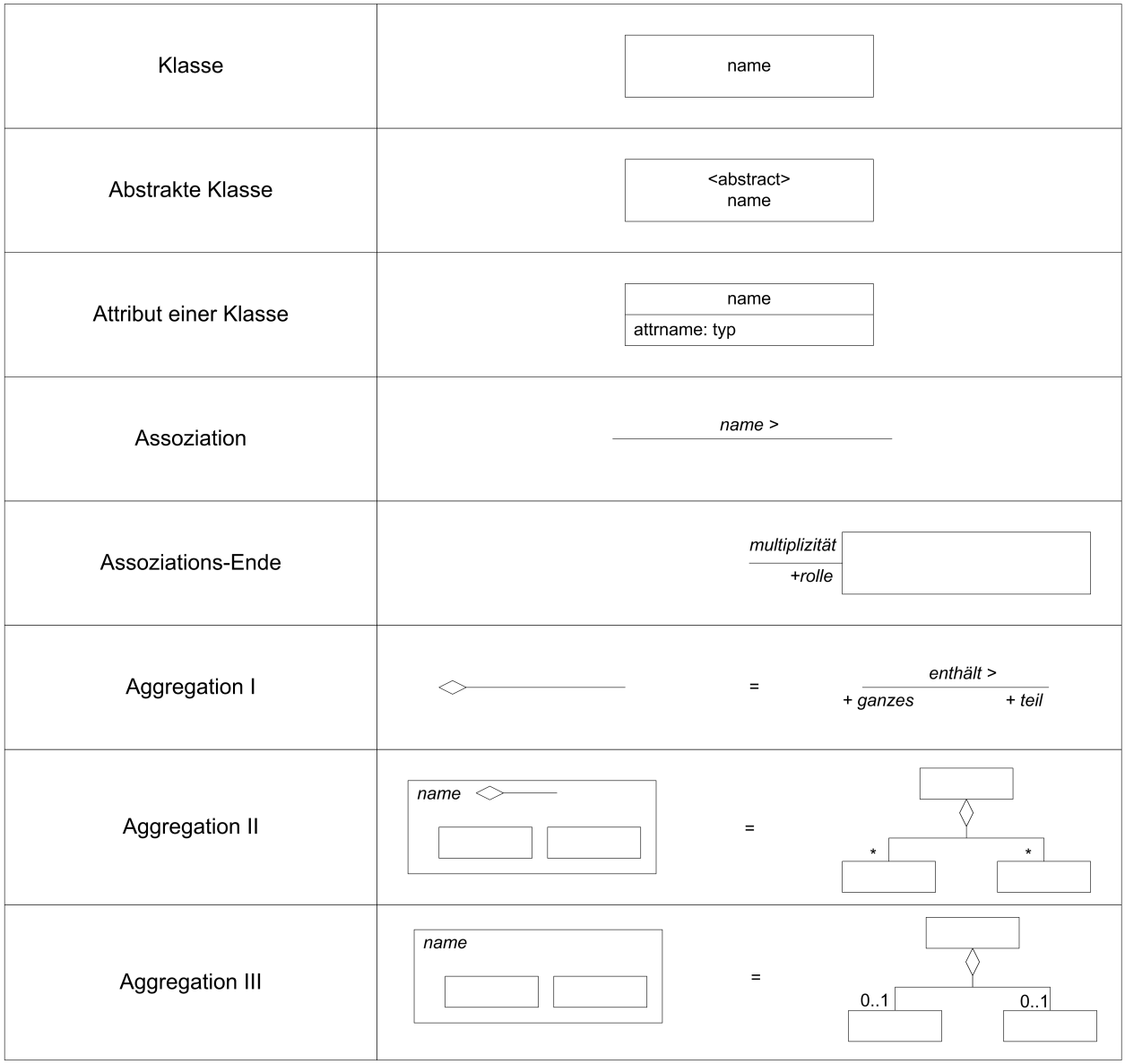

Abbildung 2.4: Angepasste konkrete Syntax

Eine Besonderheit der konkreten Syntax ist die Leserichtung von Assoziationen. Die Sprache unterstützt ausschließlich ungerichtete Assoziationen. Die Assoziationen sind somit in zwei Richtungen navigierbar. Dies ist insbesondere von Bedeutung für die Definition von Konsistenzbedingungen. Die Sprache unterstützt keine drei- und mehrwertigen Assoziationen.

Da graphische Sprachen, und im Besonderen auch die UML, in der Regel nicht ausdrucksstark genug sind, um alle Aspekte der Modelle eindeutig zu beschreiben, werden die Modelle um formal definierte Konsistenzbedingungen ergänzt. Als Sprache wird hier die Object Constraint Language [87] verwendet. Diese an die Prädikatenlogik angelehnte Spezifikationssprache ist Teil der UML 2.0 Spezifikation und unterstützt die Navigation durch die Modelle sowie die formale Beschreibung von Bedingungen über den Modellen.

Generalisierung und Spezialisierung

Die MOF Metadatenarchitektur geht implizit von einer Instanziierungsbeziehung zwischen den Modellen auf unterschiedlichen Metaebenen aus. Die Semantik der Instanziierungsbeziehungen ist jedoch nicht eindeutig definiert, sondern orientiert sich am üblichen objektorientierten Instanziierungsmodell der UML (vgl. Abschnitt 2.2). Die objektorientierte Instanziierung sieht ein einstufiges Modell vor. Eine Klasse kann durch Objekte instanziiert werden, die Objekte erben die Eigenschaften der Klasse. Ein Objekt kann jedoch selbst nicht mehr instanziiert werden. Bei einer Modellentwicklung auf Basis einer mehrstufigen Metadatenarchitektur, erlaubt dieser Ansatz keine Aussagen über die Beziehung zweier Modelle, die nicht auf unmittelbar aufeinander folgenden Metaebenen angesiedelt sind.

Die Modellierungssprache von Gnatz formalisiert dagegen explizit die Instanziierungsbeziehung zwischen Modellen über das Konzept der Spezialisierung bzw. Generalisierung. Danach ist ein Modell eine Spezialisierung seines Metamodells, das Metamodell ist dagegen eine Generalisierung des Modells. Die in der Sprache verwendete Semantik von Spezialisierung und Generalisierung orientiert sich an dem normalen Verständnis der Vererbung in objektorientierten Sprachen. Insbesondere erlaubt der Vererbungsmechanismus eine mehrstufige Spezialisierung: Ein Metamodell ist Spezialisierung eines Metametamodells und kann zu einem Modell weiter spezialisiert werden. Vorteil dieser Form der Modellierung ist, dass ein Spezialisierungsmodell im Gegensatz zum einstufigen Instanziierungsmodell prinzipiell nicht die Anzahl der Spezialisierungsstufen vorgibt.

Die Semantik der Spezialisierung über Modellebenen hinweg wird durch eine Abbildung der abstrakten Syntax auf sich selbst definiert. Ein Modellelement der Ebene M_i ist eine Spezialisierung eines Modellelements der Ebene M_{i+1}. Das spezialisierte Element erbt alle Eigenschaften seiner Generalisierung, bleibt jedoch selbst Typ und kann weiter spezialisiert werden. Bei der Spezialisierung müssen Vorgaben hinsichtlich Zuordnung und Multiplizitäten der Assoziationen des übergeordneten Modells eingehalten werden. Abbildung 2.5 zeigt am Beispiel eines einfachen Vorgehens-Metamodells eine gültige Spezialisierungskette bis hin zur Gegenstandsebene. Die Multiplizitäten im Metamodell legen z.B. fest, dass es zu einem Produkt immer genau eine verantwortliche Person geben muss. Eine Rolle darf jedoch für mehrere Produkte verantwortlich sein. Eine gültige Spezialisierung des Metamodells wäre beispielsweise die Zuordnung einer Rolle Projektleiter als verantwortliche Rolle für die Erstellung von Projektplan und Projektstatusberichten. Die Spezialisierung dieses Minimodells erfolgt in der Gegenstandsebene, in einem konkreten Projekt. Hier wird die Rolle Projektleiter im Modell mit einer konkreten Person, Herrn Müller, besetzt. Herr Müller ist verantwortlich für die Erstellung des Projektplans im Projekt sowie für die Erstellung der verschiedenen Projektstatusberichte, die im Projektverlauf zu erstellen sind.

Einen Sonderfall der Spezialisierung stellen Attribute dar. Für Attribute ohne Wertzuweisung gilt die Spezialisierungssemantik aller anderen Modellelemente. Den Attributen kann jedoch auf auf jeder Spezialisierungsebene ein Wert zugewiesen werden. Ist dies der Fall, darf der Wert bei weiteren

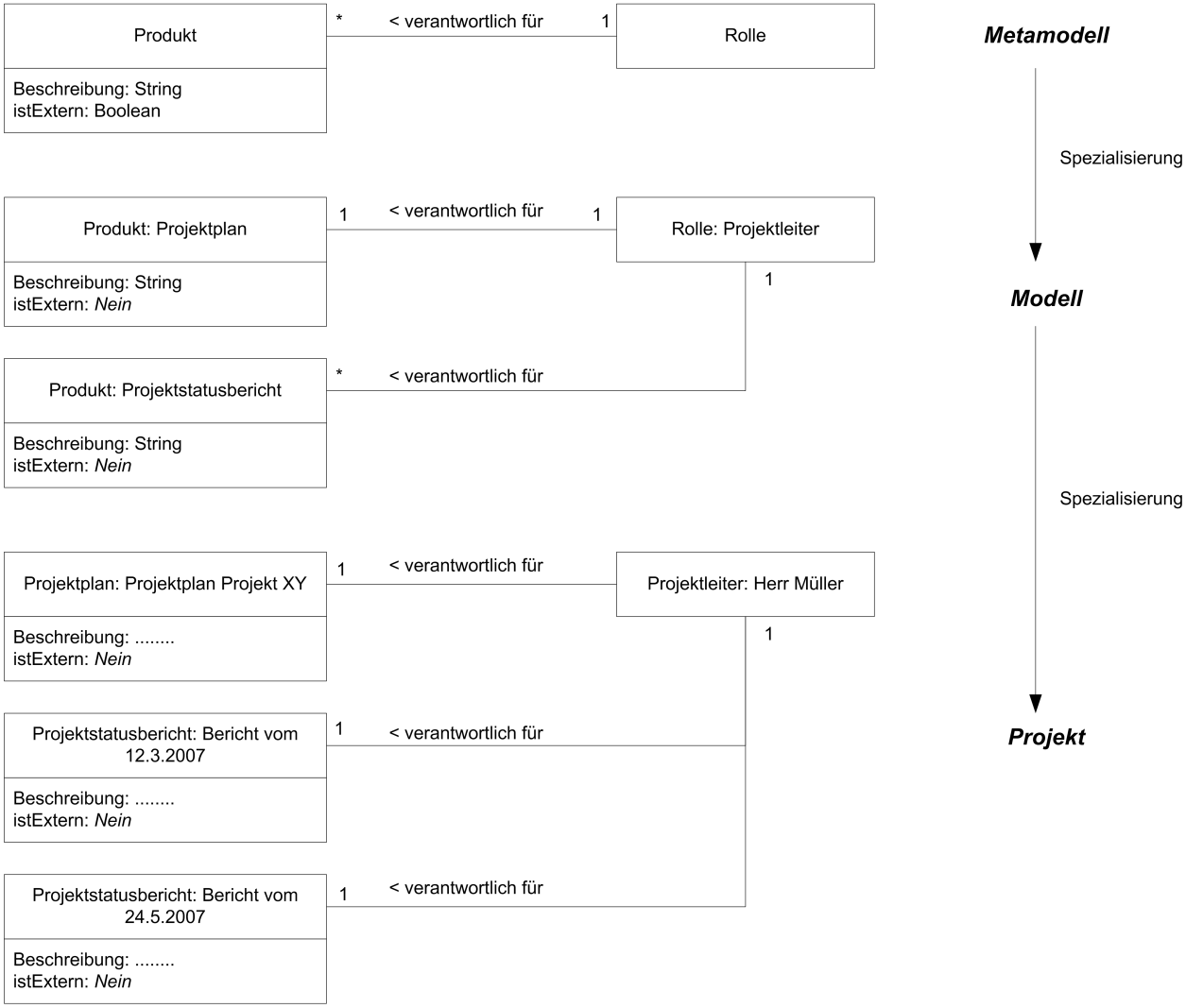

Abbildung 2.5: Generalisierung und Spezialisierung am Beispiel

Spezialisierungen jedoch nicht mehr überschrieben werden. Im Beispiel wurden der Klasse Produkt in einem (einfachen) Vorgehens-Metamodell die Attribute *istExtern* und *Beschreibung* zugeordnet. *IstExtern* kennzeichnet, ob ein Produkt dem Projekt von außen zur Verfügung zu stellen ist, oder ob es im Rahmen des Projekts erstellt wird. Das Attribut kennzeichnet eine Eigenschaft, die bereits einem Produkt auf Modellebene zugeordnet werden kann. Im Beispiel wurde das Attribut sowohl für den Projektplan wie auch für den Projektstatusbericht auf Modellebene mit dem konkreten Wert *Nein* belegt. Dieser Wert wird auf alle Produktinstanzen in einem Projekt weiter vererbt und darf dort nicht mehr verändert werden. Das Attribut *Beschreibung* bezieht sich dagegen auf einzelne Produktinstanzen in einem Projekt und kann nicht auf Modellebene nicht gesetzt. Erst in einem konkreten Projekt wird für jede Instanz eine individuelle Beschreibung erstellt.

2.4 Zusammenfassung

In diesem Kapitel wurden Modelle und Metamodelle allgemein eingeführt und die Rolle von Metamodellen zur Entwicklung von Modellen motiviert. Als ein bekannter Ansatz zur Metamodellierung wurde die MOF Spezifikation vorgestellt. Die MOF definiert eine Metadatenarchitektur sowie eine Modellierungssprache zur Entwicklung der Modelle auf den verschiedenen Ebenen der Metadatenarchitektur. Die Modellierungssprache der MOF, das MOF Model, nutzt als Sprachkonstrukte eine eingeschränkte Form der UML 2.0, die UML Infrastructure Specification. Diese Sprache weist jedoch Defizite hinsichtlich der Semantik zur Instanziierung von Modellen über mehrere Ebenen auf. Aus diesem Grund wurde für diese Arbeit eine alternative Sprache herangezogen, die eine durchgängige Lösung für das Instanziierungsproblem anbietet. Bei dieser Sprache handelt es sich um eine auf der UML basierende graphische Modellierungssprache, die speziell auf die Anforderungen von Vorgehens-Metamodellen angepasst wurde. Die Sprache wird in den folgenden Kapiteln zur Modellierung aller Metamodelle und ihrer Modelle herangezogen. Sie beschränkt sich auf eine minimale Menge von Konzepten und bleibt so einfach und übersichtlich.

Kapitel 3

Vorgehensmodelle

Vorgehensmodelle beschreiben die quantifizierbare, wiederholbare und qualitätsgesicherte Abwicklung von Softwareentwicklungsprojekten. Sie dienen den Projekten als Leitfaden und Richtschnur und unterstützen so eine geordnete Projektdurchführung. Mit dem Einsatz eines Vorgehensmodells streben Organisationen eine Standardisierung und Vereinheitlichung ihrer Prozesse an. Die Prüfung der tatsächlichen Prozessqualität kann dann beispielsweise auf der Basis von Qualitätsmanagementmodellen erfolgen. In diesem Kapitel werden Vorgehensmodelle mit ihren Kernkonzepten eingeführt und verschiedene Entwicklungsansätze vorgestellt. Abschließend werden Vorgehensmodelle zu Qualitätsmanagementmodellen und anderen verwandten Modellen abgegrenzt.

Das Kapitel ist wie folgt aufgebaut: In Abschnitt 3.1 wird der Modellcharakter von Vorgehensmodellen motiviert und es werden die zentralen Modellelemente vorgestellt. Grundlegende Begriffe, die im Zusammenhang mit Vorgehensmodellen eine Rolle spielen, wie Projekt, Prozess und Projektplan, werden in Abschnitt 3.2 eingeführt und definiert. In der Praxis verbreitete Vorgehensmodelle werden in Abschnitt 3.3 vorgestellt und ihre wesentlichen Unterschiede werden kurz erläutert. Eng mit Vorgehensmodellen verwandt sind Phasenmodelle. Obwohl häufig als Vorgehensmodelle bezeichnet, decken Phasenmodelle nur einen Teilbereich eines Vorgehensmodells ab. Abschnitt 3.4 stellt die wichtigsten Phasenmodelle mit ihren charakteristischen Eigenschaften vor. Die Rolle von Vorgehensmodellen in Organisationen, sowie Gründe für die Einführung eines Vorgehensmodells werden in Abschnitt 3.5 diskutiert. Abschnitt 3.6 schließt mit einem Vergleich der Entwicklungsansätze für Vorgehensmodelle den Themenbereich der Vorgehensmodelle ab.

Vorgehensmodelle stehen in engem Zusammenhang mit Qualitätsmanagementmodellen. Deren Aufgabe ist die Sicherstellung und Messung der Qualität eines Vorgehensmodells und seiner Anwendung. Die bekanntesten Qualitätsmanagementmodelle sowie verwandte Modelltypen werden in Abschnitt 3.7 vorgestellt. Abschnitt 3.8 fasst das Kapitel zusammen und zeigt abschließend, wie die in diesem Kapitel eingeführten Modelltypen sich gegenseitig geeignet ergänzen und gemeinsam zur Prozessverbesserung in einer Organisation beitragen.

3.1 Vorgehensmodelle als Modelle der Projektdurchführung

Ein Vorgehensmodell beschreibt auf abstrakter Ebene ein standardisiertes und einheitliches Vorgehen zur Projektdurchführung. Es ist Abbild konkreter Projekte und Modell für die Durchführung weiterer Projekte. Als Modell blendet ein Vorgehensmodell projektspezifische Details aus - beispielsweise die gewählte Technologie, Vorkenntnisse des Projektteams oder spezielle Prozesse des Kunden - und konzentriert sich statt dessen auf die Beschreibung allgemein gültiger Vorgaben und Richtlinien für die Projektdurchführung. Dazu legt ein Vorgehensmodell fest **wer** in einem Projekt **was** zu tun hat, **wie** die Aufgabe durchzuführen ist und **wann** die Ergebnisse vorzulegen sind. Ein Vorgehensmodell benennt eine Menge von Rollen (wer), die verantwortlich sind für die Erstellung von Produkten (was). Das Vorgehensmodell macht außerdem über Aktivitäten Angaben zur Erstellung der Produkte (wie) und benennt Meilensteine zu denen die Produkte vorzulegen sind (wann). Beispielsweise benennt das Vorgehensmodell den *Projektleiter* als verantwortliche Rolle für das Produkt *Projektplan*. Der *Projektplan* muss zu jedem Meilenstein in einem Projekt vorgelegt werden. Die Aktivität *Projekt planen* macht Vorgaben dazu, wie der *Projektplan* zu erstellen ist. Dementsprechend identifizieren wir folgende Kernkonzepte, die ein Vorgehensmodell mindestens bereitstellen muss: Rollen, Produkte, Aktivitäten, Phasen mit Meilensteinen. Wir bezeichnen im Weiteren die Menge aller Rollen in einem Vorgehensmodell als das *Rollenmodell*, die Menge der Produkte in einem Vorgehensmodell als das *Produktmodell* und die Menge der Aktivitäten als *Aktivitätsmodell*. Meilensteine und Phasen als Rückgrad der Projektplanung fassen wir im Folgenden unter dem Begriff *Ablaufmodell* zusammen. Neben den der vier Teilmodellen kann ein Vorgehensmodell beliebige weitere Konzepte zur Verfügung stellen, wie beispielsweise Methoden, Werkzeuge, Vorlagen, Checklisten oder Vorgaben zu Standards. Abbildung 3.1 stellt schematisch den Aufbau eines Vorgehensmodells aus Teilmodellen und Ergänzungen vor. Im Folgenden werden die Teilmodelle näher erläutert.

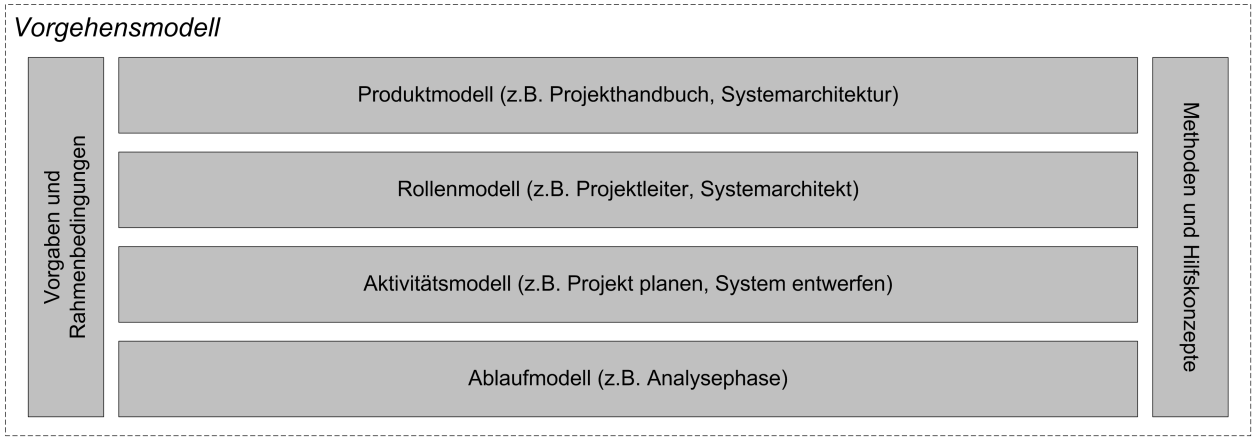

Abbildung 3.1: Teilmodelle von Vorgehensmodellen

Das *Rollenmodell* umfasst alle in einem Vorgehensmodell definierten Rollen. Eine Rolle ist charakterisiert durch eine Beschreibung der in einem Projekt durchzuführenden Aufgaben mit dem dafür notwendigen Fähigkeitsprofil, sowie der Menge der zu verantwortenden Produkte. So ist beispielsweise die Rolle Projektleiter üblicherweise verantwortlich für die Erstellung eines Produkts Projektplan. Damit erfüllt er die Aufgabe zur Planung und Steuerung eines Projekts. Eine Person, die diese Rolle in einem Projekt ausfüllt, muss hierzu die entsprechenden Fähigkeiten, beispielsweise die Beherrschung von Planungsmethoden oder die Fähigkeit zur Kommunikation und Führung mitbringen. Weitere Rollen in einem Projekt sind beispielsweise der Systemarchitekt der Entwickler oder der Qualitätsverantwortliche. In einem Projekt kann eine Rolle durch mehrere Personen ausgefüllt werden. Ebenso können mehrere Personen dieselbe Rolle in einem Projekt ausfüllen.

Das *Produktmodell* umfasst alle in einem Vorgehensmodell definierten Produkte. Ein Produkt legt den Typ der in einem realen Projekt zu erarbeitenden Ergebnissen fest. Dazu macht es konkrete Vorgaben zu Form und insbesondere zu Inhalt der Ergebnisse. In einem Projekt werden Instanzen des Produkts erstellt. Je nach Produkt kann die Erstellung mehrerer Instanzen innerhalb des Projekts erforderlich sein. So kann beispielsweise das Produkt *Systemspezifikation* in einem konkreten Projekt mehrfach instanziiert werden: für das *System* in seiner Gesamtheit sowie für einzelne *Systemkomponenten*. Die Systemspezifikation schreibt vor, welche Inhalte für eine Spezifikation zu erarbeiten sind. Die Instanzen gestalten diese Inhalte für das System oder eine konkrete Systemkomponente aus. Der Begriff eines Produkts ist hier allgemein zu verstehen. Bei einem Produkt kann es sich um jede Form eines Ergebnistyps in einem Projekt handeln. Ein Produkt kann beispielsweise ein Dokument, ein Bericht, ein Protokoll, ein Systemelemente, oder das System selbst sein. Instanzen der Produkte sind Gegenstand der in einem Projekt durchzuführenden Qualitätssicherungsmaßnahmen und damit Grundlage der Projektüberwachung. Produkte können je nach Vorgehensmodell Hierarchien bilden.

Das *Aktivitätsmodell* umfasst alle in einem Vorgehensmodell definierten Aktivitäten. Eine Aktivität ist die Beschreibung eines konkret durchzuführenden Prozessschritts im Projekt. Ziel der Aktivität ist die Erstellung oder Modifikation eines oder mehrerer Produkte. Durchgeführt wird eine Aktivität von einer Rolle mit entsprechendem Fähigkeitsprofil. Instanzen von Aktivitäten sind die im Projektplan einzuplanenden Arbeitsschritte. Wie bei Produkten, können auch bei Aktivitäten mehrere Instanzen einer Aktivität im Projekt auftreten, d.h. eine Aktivität wird mehrfach im Projektplan eingeplant und wird entsprechend mehrfach in einem Projekt durchgeführt. Beispielsweise ist es Aufgabe des Projektleiters, zu jedem Meilenstein in einem Projekt die Aktivität zur Erstellung und Pflege des Projektplans durchzuführen. Wie Produkte können Aktivitäten abhängig vom Vorgehensmodell Hierarchien bilden.

Das *Ablaufmodell* definiert die übergeordnete zeitliche und inhaltliche Organisation der durchzuführenden Aktivitäten und der zu erstellenden Produkte. Hierzu stützt sich das Ablaufmodell auf die Konzepte Phase und Meilenstein. Phasen kennzeichnen Ausschnitte aus dem Lebenszyklus eines

Softwaresystems. Beispiele sind etwa Analysephase, Entwurfsphase oder Implementierungsphase. Meilensteine markieren Phasenübergänge und unterstützen die Projektsteuerung durch Angaben zu konkreten Phaseneingangs- und Phasenausgangskriterien. Meilensteine dienen vorrangig als Qualitätsmesspunkte in Projekten. Zu einem Meilensteinen wird der aktuelle Projektstatus anhand der Qualität der erarbeiteten Produkte geprüft und es wird über den Projektfortschritt entschieden. Im Vorgehensmodell gibt es eine eindeutige Zuordnung von Aktivitäten zu Phasen und von Produkten zu Meilensteinen.

Die vier genannten Teilmodelle bilden die Basis eines jeden Vorgehensmodells. Zusätzlich kann ein Vorgehensmodell beliebige weitere Konzepte enthalten. Welche dies sind, variiert von Vorgehensmodell zu Vorgehensmodell. Eine Standardisierung fehlt zum aktuellen Zeitpunkt. So bieten einige Vorgehensmodelle als Unterstützung für die Durchführung ihrer Aktivitäten und zur Erarbeitung der Produkte eine konkrete *Methodik* an. Zur Umsetzung der Methodik stehen *Hilfskonzepte* wie Checklisten, Werkzeuge oder Vorlagen zur Verfügung. Neben Methoden und Hilfskonzepten definieren Vorgehensmodelle *Vorgaben und Rahmenbedingungen* zur Projektdurchführung. Hierbei kann es sich beispielsweise um Angaben zu einzuhaltenden externen Standards (z.B. ISO Standards), um interne Richtlinien (z.B. Vorgaben zur Vertragsgestaltung), Styleguides (z.B. Programmierrichtlinien) oder Handbücher (z.B. Handbuch zum Qualitätsmanagement) handeln. Die hier verwendeten Namen der Kernkonzepte können ebenfalls von Vorgehensmodell zu Vorgehensmodell variieren, Jedes Vorgehensmodell stellt jedoch mindestens Konzepte zur Verfügung, die in irgendeiner Form den hier genannten Konzepten entsprechen.

3.2 Grundlegende Begriffe und Definitionen

Nachdem der prinzipielle Aufbau von Vorgehensmodellen vorgestellt wurde, werden in diesem Kapitel Begriffe definiert, die im Zusammenhang mit Vorgehensmodellen eine zentrale Rolle spielen. Diese sind neben dem Vorgehensmodell selbst, der Prozess, das Projekt und der Projektplan.

Ein *Prozess* ist nach [57] eine Folge von Schritten, die zur Erreichung eines gegebenen Zwecks ausgeführt werden. Als ein Beispiel eines Prozesses kann die Durchführung eines Projekts genannt werden. Ein Prozess kann unterteilt werden in einzelne *Prozessschritte*. Diese können atomar oder hierarchisch aus weiteren Prozessschritten aufgebaut sein.

Ein *Projekt* ist nach DIN 69901 ein einmaliges Vorhaben, bei dem innerhalb einer definierten Zeitspanne ein definiertes Ziel erreicht werden soll. Es ist gekennzeichnet durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit, wie z.B.: Zielvorgaben, zeitliche, finanzielle, personelle oder andere Bedingungen, Abgrenzungen gegenüber anderen Vorhaben und projektspezifische Organisation. Als Vorhaben ist ein Projekt eine abstrakte, nur schwer greifbare Größe. Konkrete Repräsentation eines Projekts ist ein eindeutiger *Projektidentifikator*, beispielsweise der Projektname.

Ein *Vorgehensmodell*[1] ist das Modell einer Projektdurchführung. Es beschreibt auf allgemeiner Ebene die strukturierte und qualitätsgesicherte Durchführung von Projekten. Damit hat ein Vorgehensmodell Vorbildcharakter und dient als Leitfaden für die Durchführung konkreter Projekte.

Projekte können unterschiedlicher Natur sein hinsichtlich ihres definierten Ziels. So werden in einer Organisation neben typischen Entwicklungsprojekten beispielsweise auch Beratungsprojekte oder Analyseprojekte durchgeführt. Die Klasse der hier betrachteten Vorgehensmodelle beschränkt sich jedoch auf die Beschreibung einer bestimmten Klasse von Projekten, den Softwareentwicklungsprojekten. In Anlehnung an die Definition eines Projekts definieren wir ein *Softwareentwicklungsprojekt* als ein Projekt mit dem Ziel der Erstellung eines Softwaresystems in vorgegebener Zeit, unter Einhaltung des vorgegebenen Budgets und unter Sicherstellung der geforderten Leistung in angemessener Qualität.

Ein Vorgehensmodell ist eine abstrakte Beschreibung der Projektdurchführung. Ein *Projektplan* beschreibt dagegen den tatsächlichen Prozess der Projektdurchführung. Ein Projektplan dokumentiert zu jedem Zeitpunkt der Projektdurchführung den aktuellen Projektstatus. Er gibt an welche Ergebnisse bereits erarbeitet wurden, welche Prozessschritte noch durchzuführen sind und wer die Verantwortung für ihre Durchführung trägt. Ein Projektplan ist somit einerseits selbst ein im Rahmen der Projektdurchführung zu erstellendes Ergebnis, andererseits entspricht er einem Modell eines konkreten Prozesses der Projektdurchführung. In der Literatur wird häufig der Projektplan als Instanz eines Vorgehensmodells bezeichnet (vgl. [8] und [49]), eine Sichtweise, die kritisch zu sehen ist, da Projektpläne keine vollständigen Prozessbeschreibung der Projektdurchführung liefern.

3.3 Vorgehensmodelle in der Praxis

In diesem Abschnitt werden nun einige der bekannteren Vorgehensmodelle, die sich in der Industrie durchsetzen konnten, vorgestellt. Bei den hier genannten Vorgehensmodellen handelt es sich um die heute wichtigsten frei oder kommerziell verfügbaren Vorgehensmodelle. Daneben gibt es eine wesentlich größere Anzahl an organisationsspezifischen Vorgehensmodellen, die auf die Belange einer konkreten Organisation zugeschnitten sind und nicht der Allgemeinheit zur Verfügung stehen. Weiterhin können die Modelle prinzipiell danach unterschieden werden, ob sie methodenneutral konzipiert wurden oder bereits eine eigene Methodik angeben.

Methodenneutrale Vorgehensmodelle

Ein in Deutschland verbreitetes Vorgehensmodell ist das V-Modell des Bundes in seinen verschiedenen Ausprägungen. Das V-Modell dient seit über 15 Jahren als Standard für die Durchführung von

[1] Im englischsprachigen Raum wird üblicherweise der allgemeinere Begriff 'process model' verwendet. Als 'process model' kann jedoch jede Form von Prozessbeschreibung betrachtet werden, wie beispielsweise auch die Beschreibung von Geschäftsprozesse. Eine explizite Unterscheidung zwischen Prozessmodell und Vorgehensmodell, wie im Deutschen, existiert hier nicht.

IT-Vorhaben. Kennzeichnend und Namensgeber für das V-Modell ist die V Form des zu Grunde liegenden V-Modells von Rook [99]. Erste Version des V-Modells des Bundes war das V-Modell 92, welches als eines der ersten Vorgehensmodelle die Durchführung von IT Projekten für den Bund vereinheitlichte und standardisierte. Fünf Jahre später wurde das V-Modell 92 durch eine überarbeitete Version abgelöst, das V-Modell 97 [39]. Im Jahre 2005 wurde schließlich das V-Modell XT [118] als Nachfolger des V-Modell 97 veröffentlicht und stellt den aktuellen Standard dar. Alle Varianten des V-Modells wurden bewusst methodenneutral gehalten, um die breite Einsetzbarkeit der Modelle zu gewährleisten.

Ein weiteres methodenneutrales Vorgehensmodell ist der OPEN Prozess [53]. Der OPEN Prozess ist ein frei verfügbares Vorgehensmodell für die Entwicklung von objektorientierten, komponentenbasierten Softwaresystemen. OPEN steht für 'Object oriented Process Environment and Notation'. Im Zentrum des OPEN Prozesses stehen Aktivitäten zur Beschreibung von Teilprozessen. Die Aktivitäten werden zu Tasks verfeinert. Der OPEN Prozess definiert auch Ergebnistypen. Der Fokus liegt jedoch auf den dynamischen Elementen. Der Prozess wurde und wird vom OPEN Consortium [129] entwickelt und gepflegt. Während die OPEN Spezifikation selbst noch methodenneutral ist, gibt es bereits Erweiterungen für eine Anwendung des OPEN Prozesses mit der UML [55]. Der OPEN Prozess wurde bereits mit der Option entwickelt, relativ einfach methodenähnliche Konzepte zu integrieren, gibt diese jedoch nicht verpflichtend vor.

Die ISO Standards ISO/IEC 12207 [62] und ISO/IEC 15288 [63] definieren ebenfalls methodenneutrale Vorgehensmodelle. Beide Standards beschreiben die wesentlichen Elemente eines Vorgehensmodells, wobei ISO 15288 die Entwicklung von Systemen (Hardware und Software) unterstützt, ISO 12207 sich dagegen auf den Softwareentwicklungsprozess beschränkt. Die ISO Standards stellen einerseits vollwertige Vorgehensmodelle dar, sind jedoch andererseits nicht unbedingt zum unmittelbaren Einsatz gedacht. So handelt es sich etwa bei ISO 12207 um ein Referenzmodell für das Reifegradmodell Spice (vgl. Abschnitt 3.8).

Vorgehensmodelle mit Methoden

Neben methodenneutralen Vorgehensmodellen gibt es eine Reihe von Vorgehensmodellen, die bereits eine konkrete Methodik vorgeben. Hier sind insbesondere der Unified Process [69] und seine Varianten zu nennen: der Rational Unified Process RUP [77] und der Object Engineering Process OEP [91]. Eine neuere Variante ist der OpenUp Prozess [119], eine leichtgewichtige Variante des Rational Unified Process. Der Unified Process wurde von den Vätern der UML, Booch, Rumbaugh und Jacobson, entwickelt. Dementsprechend unterstützen alle Varianten die Softwareentwicklung nach dem objektorientierten Paradigma. Aus Ablaufsicht sieht der Unified Process ein iteratives Vorgehen vor.

Auch das Microsoft Solution Framework (MSF) [124] kann als leichtgewichtiges Vorgehensmodell mit integrierter Methodik eingeordnet werden. Beim MSF handelt es sich um ein stark werkzeug-

zentriertes Vorgehensmodell. Es fokussiert die Durchführung von Softwareentwicklungsprojekten im Microsoft-Umfeld. So bindet der Prozess unterschiedliche Microsoft-Werkzeuge ein, beispielsweise zum Konfigurationsmanagement, zur Projektplanung oder zur Entwicklung. Eine Anwendung des MSF außerhalb des Microsoft-Umfelds ist nicht vorgesehen und auch nicht realistisch.

Einen Spezialfall innerhalb der Vorgehensmodelle stellen agile Methoden dar. Sie sind nach der hier zu Grunde gelegten Definition als vollwertige Vorgehensmodelle einzuordnen, da sie ein durchgängiges Phasenmodell kennen und konkrete Vorgaben für die inhaltliche Projektdurchführung machen. So kennen auch agile Methoden Produkte, Aktivitäten und Rollen. Im Gegensatz zu Entwicklungsmethoden haben sie den gesamten Prozess der Projektdurchführung im Fokus. Trotz des Namens geben agile Methoden keine direkte Methodik vor, sondern definieren eine Menge von Prinzipien und Techniken, die im Rahmen der Projektdurchführung anzuwenden sind. Grundlage der agilen Methoden sind zwölf im so genannten agilen Manifest vorgegebene Prinzipien [123], die den Begriff der Agilität in einem Projekt charakterisieren. Die Prinzipien decken alle Bereiche der Projektdurchführung ab wie beispielsweise das Projektmanagement, die Releaseplanung oder den Entwicklungsprozess. Repräsentative Vertreter der agilen Methoden sind beispielsweise eXtreme Programming ([12], [122], [12]), Scrum [104], [128] oder Crystal Clear [31].

3.4 Abgrenzung zu Phasenmodellen

Phasenmodelle, häufig auch als Lifecycle-Modelle bezeichnet, konzentrieren sich im Gegensatz zu Vorgehensmodellen ausschließlich auf die Modellierung des übergeordneten Projektablaufs. Sie sind im Prinzip unabhängig einsetzbar, finden jedoch häufig als Teil eines Vorgehensmodells Verwendung. Eine Phase kennzeichnet dabei einen spezifischen Abschnitt im Lebenszyklus eines Systems. Typische Phasen der Systementwicklung sind Analyse, Design, Implementierung, Test und Auslieferung, wobei Bezeichnung und Anzahl der Phasen von Modell zu Modell variieren können.

Historisch können Phasenmodelle als Vorgänger heutiger Vorgehensmodelle gesehen werden. In ihren Anfängen wurde Softwareentwicklung noch als hochgradig kreative und künstlerische Tätigkeit gesehen, die stark von den Fähigkeiten des einzelnen Entwicklers abhing. Mit steigenden Fähigkeiten von Computern und Programmiersprachen stiegen jedoch die Anforderungen an die Programme und damit die Komplexität der Entwicklungsaufgaben. Die bis dahin praktizierte intuitive Vorgehensweise (auch als 'code and fix model' bezeichnet [18]) stieß an ihre Grenzen. Es entstanden die ersten Phasenmodelle (auch Lifecycle Modelle oder Lebenszyklusmodelle genannt (siehe [83], [103]) als Vorgänger der heutigen Vorgehensmodelle. Phasenmodelle definieren auf der Basis von Meilensteinen und Phasen einen Ablaufrahmen für die Projektdurchführung. Sie legen im Wesentlichen fest, welche Phasen in einem Projekt zu durchlaufen sind, in welcher Reihenfolge (z.B. sequentiell, überlappend, nebenläufig) die Phasen durchlaufen werden können, welche Themenbereiche innerhalb dieser Phasen zu bearbeiten sind und welche Übergangskriterien von einer Phase zur nächsten gelten. Übergangskriterien umfassen Abschlusskriterien für eine Phase, Entscheidungs-

kriterien für die Auswahl der folgenden Phase und Eingangskriterien für die nächste Phase ([18]). Erste streng sequentielle Phasenmodelle, so genannte *stagewise models*, entstanden bereits in den 50er Jahren (vgl. [14]), erwiesen sich jedoch durch die starre Phasenabfolge in ihrer Handhabung als zu unflexibel und fanden wenig Anklang in der Praxis.

Mit dem *Wasserfallmodell* von Royce [100] wurde 1970 die streng sequentielle Abfolge der Phasen flexibilisiert. Neu hinzu kamen insbesondere Feedbackschleifen, die bei Bedarf Rücksprünge auf frühere Phasen erlaubten. Bis heute liegt vielen Vorgehensmodellen das originalen Wasserfallmodell oder eine seiner Varianten zu Grunde. Eine Variante des Wasserfallmodells ist beispielsweise das V-Modell von Rook [99]. Im V-Modell wird jeder Phase im Wasserfallmodell eine Phase zur Verifikation und Validierung der Phasenergebnisse gegenübergestellt. So kennt das V-Modell explizit ein Phase zum Modultest, zum Integrations- und Systemtest, zum Akzeptanztest und zum Betriebstest. Kernproblem des Wasserfallmodells und seiner Varianten ist der sequentielle Ablauf der Projektdurchführung. Die Anforderungen werden in einer frühen Phase einmalig festgelegt und das System dementsprechend entwickelt. Der Anwender erhält erst relativ spät eine lauffähige Version seiner Anforderungen. Dieses Vorgehen erfordert stabile Anforderungen und ausreichend Erfahrung des Anwenders bei deren Formulierung. Diese Voraussetzungen sind jedoch häufig nicht erfüllbar. Die Erfahrung hat gezeigt, dass Anforderungen während der Projektlaufzeit häufig noch starken Änderungen unterworfen sind. Für diese Fälle braucht es flexiblere Modelle wie beispielsweise das Spiralmodell.

Mit dem *Spiralmodell* von Boehm [18] wurde 1986 das iterative Vorgehen in das Zentrum der Softwareentwicklung gestellt. Die Entwicklung eines Systems erfolgt in kurzen Zyklen. In jedem Zyklus werden alle Phasen durchlaufen und es wird ein relevanter und aussagekräftiger Bestandteil des Systems entwickelt. Dieser dient als Grundlage für die Planung des nächsten Zyklus. Mit seinem Ansatz verfolgte Boehm das Ziel einer risikogesteuerten Projektdurchführung. Die Einführung kurzer Zyklen im Entwicklungsprozess erlaubt die wiederholte Neujustierung im Projektverlauf mit einer Neubewertung der Risiken. Das Spiralmodell wird in der Literatur teilweise nicht als eigenständiges Vorgehensmodell, sondern als Metamodell bezeichnet ([8]). Konkrete Ausprägungen sind beispielsweise das evolutionäre Modell und das inkrementelle Modell. Jedes dieser Modelle folgt vollständig oder in Teilen der iterativen Vorgehensweise des Spiralmodells, konzentriert sich jedoch auf eine spezifische Problemstellung.

Im *evolutionären Modell* werden System und Anforderungen parallel entwickelt und umgesetzt. Das Modell sieht die Formulierung der Kernanforderungen zu Beginn eines Projekts vor. Diese werden in einer Iteration umgesetzt und dem Anwender zur Prüfung vorgelegt. In einem folgenden Zyklus werden Änderungswünsche übernommen, weitere Anforderungen formuliert und das bestehende Minimalsystem (Nullversion) entsprechend erweitert. Diese Vorgehens wird durchgeführt, bis keine weiteren Anforderungen offen sind. Problem des evolutionären Modells ist das Fehlen eines verbindlichen Anforderungsrahmens für das Gesamtprojekt. Dies erlaubt keine stabile Aufwandschätzung

und Planung über einen Zyklus hinaus und kann in jedem Zyklus zu Änderungen an der Architektur führen. Das evolutionäre Modell wird in der Literatur auch als evolutionäres Prototyping [16] bezeichnet: mit der initialen Iteration wird ein Prototyp entwickelt, der mit jeder Iteration evolutionär verfeinert und erweitert wird.

Das *inkrementelle Modell* adressiert das Probleme der instabilen Anforderungen. Hier werden zu Beginn eines Projekts alle Anforderungen erfasst und dokumentiert. Die Umsetzung erfolgt dann ähnlich zum evolutionären Ansatz: ausgehend von einer minimalen Auswahl an Mussanforderungen an das System, werden in jedem Zyklus weitere Anforderungen hinzugenommen und umgesetzt. Bestehende Anforderungen werden (wenn möglich) nicht mehr geändert. Aus diesem Grund wird hier nicht von Iterationen, sondern von Inkrementen gesprochen. Mit jedem Inkrement wird eine weitere Auswahl der ursprünglich ermittelten Anforderungen übernommen und umgesetzt. Vorteil des inkrementellen Modells ist die im Vergleich zum evolutionären Modell zuverlässigere Planung und Schätzung zu Beginn eines Projekts.

3.5 Motive für den Einsatz eines Vorgehensmodells

Viele Firmen bzw. Organisationen, die in irgendeiner Form Softwareentwicklung betreiben, haben die Notwendigkeit für eine Standardisierung ihrer Projektabwicklung sowie zur Festlegung organisationsweiter Vorgaben und Richtlinien erkannt. Die Motive für den Einsatz eines Vorgehensmodells können jedoch von Organisation zu Organisation variieren. Sie hängen davon ab, welche Ziele eine Organisation hinsichtlich ihrer Prozessreife anstrebt. War in den Anfängen der Softwareentwicklung vor allem eine Verbesserung der Testprozesse und damit eine Verbesserung der Systemqualität vorrangiges Ziel, treten heute vermehrt Planbarkeit, Messbarkeit und Vergleichbarkeit der Prozesse selbst in den Vordergrund. Nachgelagerte Ziele, die damit erreicht werden sollen, sind beispielsweise Wettbewerbsvorteile und verbesserte Marktpositionierung durch die Erfüllung Internationaler Standards bzw. den Nachweis einer offiziell vorgegebenen Prozessreife. Im Folgenden werden Gründe und Motive für den Einsatz eines Vorgehensmodells in Organisationen diskutiert. Damit werden implizit die vielfältigen Aufgaben, die ein Vorgehensmodell in einer Organisation erfüllt, charakterisiert.

Einheitliche Projektplanung und -organisation

Ein Vorgehensmodell definiert über das Ablaufmodell auf grobgranularer Ebene eine einheitliche Struktur zur Projektplanung. Es legt die zu durchlaufenden Phasen mit den entsprechenden Aktivitäten fest und definiert relevante Meilensteine zur Überwachung des Projektstatus. Instanz eines Ablaufmodells im Vorgehensmodell ist ein Meilensteinplan bzw. initialer Projektplan für ein konkretes Projekt. Der initiale Projektplan legt alle im Projekt zu durchlaufenden Instanzen von Phasen und Meilensteinen fest und bringt sie in eine sinnvolle zeitliche Reihenfolge. Zu den Instanzen der

Phasen werden Instanzen der Aktivitäten eingeplant, so dass sichergestellt ist, dass die erwarteten Ergebnisse zu den konkreten Meilensteinen vorliegen.

Der Einsatz eines Vorgehensmodells bietet einer Organisation die Möglichkeit zur Vereinheitlichung der Projektplanung und damit zur Vereinheitlichung der gesamten Projektabwicklung. Phasen und Meilensteine sind im Vorgehensmodell vorgegeben, ebenso der Prozess an einem Meilenstein zum Übergang von einer Phase in die Nächste. Durchzuführende Aktivitäten, notwendige Rollen und zu erstellende Produkte sind bekannt. Die für ein konkretes Projekt gültige Zusammenstellung der Instanzen ist im Projektplan dokumentiert.

Auch wenn Projektpläne sich auf Grund projektspezifischer Vorgaben und Rahmenbedingungen im Detail unterscheiden können, wird so mit einem Vorgehensmodell die Projektplanung und damit die Projektabwicklung weitgehend vereinheitlicht. Ein Vorgehensmodell erfüllt damit mehrere Aufgaben. Es unterstützt Projektleiter bei ihrer Aufgabe der Projektplanung und gibt insbesondere unerfahrenen Projektleitern einen fest definierten Rahmen vor. Für das Management bietet eine einheitliche Projektabwicklung nach den Vorgaben eines Vorgehensmodells eine bessere Vergleichbarkeit und Bewertbarkeit der Projekte. In Kombination mit einem durchgängigen Qualitätsmanagementprozess erlaubt ein Vorgehensmodell eine Qualitätsbewertung von Projekten zu unterschiedlichen Zeitpunkten im Projektverlauf.

Förderung des Prozessdenkens

Ein Vorgehensmodell unterstützt maßgeblich die Verbesserung der Prozessqualität in Organisationen durch Förderung des 'Prozessdenkens'. Denken in Prozessen ist hier im Gegensatz zu intuitivem Vorgehen einzuordnen. Ein Vorgehensmodell definiert die standardmäßig anzuwendenden Prozesse. Den Mitarbeitern einer Organisation sind die Prozesse und die einzelnen Prozessschritte bekannt. Insbesondere wenden die Projektmitarbeiter die definierten Prozesse korrekt an. Auch wenn im Einzelfall intuitives Vorgehen schneller zum Ziel führen würde, wird akzeptiert und anerkannt, dass die Einhaltung der vorgegebenen Prozesse langfristig höheren Nutzen bringt. So bringt beispielsweise ein geordneter Prozess zum Änderungsmanagement die Erfassung und Bearbeitung von Änderungswünschen der Anwender in einen definierten Prozess mit klaren Regeln und Entscheidungsprozessen, der keine Berücksichtigung undokumentierter Änderungswünsche zulässt. Dies wirkt sich unmittelbar auf die Stabilität der Anforderungen, auf die Planbarkeit der Entwicklung und damit auf die Planbarkeit des gesamten Projektes aus.

Unterstützt wird Prozessdenken durch ein einheitliches Prozessvokabular, wir es im Vorgehensmodell definiert ist. Ein Vorgehensmodell gibt durch Namen für Produkte, Aktivitäten, Rollen, Phasen und Meilensteine ein in der gesamten Organisation einheitliches Vokabular vor. Dies fördert ein einheitliches Prozessverständnis in der Organisation. Es erleichtert die Verständigung innerhalb eines Projekts, aber auch über Projektgrenzen hinweg. Es beugt Missverständnissen vor und unterstützt

zu einem gewissen Grad eine Identifikation des Teams mit dem Vorgehensmodell und dem von ihm vorgegebenen Prozessen.

Konstruktive Qualitätssicherung

Ein Vorgehensmodell ist eine essentielle Maßnahme zur Qualitätssicherung in einer Organisation. Im Rahmen des Qualitätsmanagement können zwei Arten von Qualitätssicherungsmaßnahmen unterschieden werden: konstruktive und analytische Qualitätssicherungsmaßnahmen. Konstruktive Maßnahmen sind Maßnahmen, die vorausschauend in einer Organisation eingesetzt werden. Ihr Einsatz wirkt sich positiv auf die Qualität des zukünftig zu erstellenden Produktes aus. Typische konstruktive Maßnahmen sind beispielsweise Schulungen der Mitarbeiter, Einsatz von Werkzeugen und Methoden oder Vorgaben von Standards. Analytische Qualitätssicherungsmaßnahmen fokussieren dagegen die Ermittlung der Qualität eines bereits erstellten Produkts. Im Rahmen der analytischen Qualitätssicherung wird nach Erstellung eines Produkts mit geeigneten Maßnahmen, z.B. Test oder Review, dessen Qualität ermittelt und ggf. werden Maßnahmen zur Qualitätsverbesserung eingeleitet.

Ein Vorgehensmodell ist eine Maßnahme der konstruktiven Qualitätssicherung. Sein Einsatz wirkt sich maßgeblich auf die Prozessqualität bei der Durchführung von Projekten aus und beeinflusst so die Qualität des zu entwickelnden Softwaresystems. Durch organisatorischen und inhaltlichen Vorgaben unterstützt ein Vorgehensmodell klare Organisations- und Ablaufstrukturen in Projekten. Es erlaubt zu definierten Zeitpunkten die Überprüfung des Projektstatus hinsichtlich Qualität, Zeit und Budget und erhöht so die Transparenz der Projektdurchführung. Des Weiteren unterstützt ein Vorgehensmodell durch explizite Vorgaben an die zu erarbeitenden Zwischenergebnisse sowie deren Bewertung hinsichtlich Qualität und Vollständigkeit.

Kontinuierliche Prozessverbesserung

Ein Vorgehensmodell beschreibt allgemein und wiederholbar alle in einem Projekt durchzuführenden Aktivitäten und macht Vorgaben für die zu erarbeitenden Produkte. Es kann so als objektiver Maßstab zur Bewertung der Qualität der Projektergebnisse und damit zur Bewertung der Qualität der Projektabwicklung in ihrer Gesamtheit herangezogen werden. Dies bildet die Grundlage für die Ermittlung der Prozessreife einer Organisation. Die Prozessreife macht einerseits Aussagen über die Qualität der in einer Organisation definierten Prozesse selbst, andererseits wird zur Ermittlung der Prozessreife auch die Anwendung der definierten Prozesse in den Projekten bewertet.

Als Maß für die Prozessreife einer Organisation dienen Reifegradmodelle. Reifegradmodelle definieren mehrere Stufen der Prozessreife. Jede Stufe stellt bestimmte Qualitätsanforderungen an Prozessdefinition und Prozessdurchführung. Bekannte Reifegradmodelle sind beispielsweise Spice oder CMMI (vgl. Abschnitt 3.7). Der Prozessreifegrad erlaubt eine objektive und vergleichbare Aussage über die Prozessqualität von Organisationen. Je nach Reifegradmodell wird der Prozessreifegrad

für einzelne Teilprozesse oder für eine vorgegebene Menge von Teilprozessen festgelegt. Der Prozessreifegrad einer Organisation gibt somit Aufschluss über die Qualität der Projektdurchführung.

Prozessreifegradmodelle werden aus unterschiedlichen Gründen eingesetzt. Einerseits unterstützen sie Bestrebungen einer Organisation zur Verbesserung der internen Prozesse durch Angabe eines Zielrahmens. Einer Organisation steht mit einem Reifegradmodell ein objektiver Maßstab für den Soll-Ist Vergleich ihrer aktuellen Prozesse zur Verfügung. Gleichzeitig spielen Prozessreifegradmodelle in der Industrie eine wachsende Rolle. So fordern beispielsweise mehr und mehr Auftraggeberorganisationen bei Ausschreibungen einen spezifischen Prozessreifegrad ihrer Zulieferfirmen.

3.6 Beschreibungsansätze für Vorgehensmodelle

Ein Aspekt, der für diese Arbeit von großer Bedeutung ist, ist die Beschreibung von Vorgehensmodellen. Ein Vorgehensmodell muss, damit es anwendbar ist, in einer geeigneten Sprache dokumentiert werden. Eine häufig gewählte Beschreibungsform für Vorgehensmodelle ist reines Textformat. Zur Verbesserung der Darstellung werden strukturelle Hilfsmittel wie Tabellen oder Listen verwendet, die Texte werden bei Bedarf um informelle Graphiken ergänzt, die zur Veranschaulichung der Inhalte dienen. Vor allem in den Anfängen der Vorgehensmodellentwicklung war die Verwendung informeller Beschreibungssprachen üblich. Vorteil einer informellen Beschreibung von Vorgehensmodellen ist deren Verständlichkeit. Auch Personen mit geringen Vorkenntnissen können in Projekten den Beschreibungen der Vorgehensmodelle einfach folgen und diese umsetzen. Eine frühe Form informeller Vorgehensmodelle sind so genannte *Projekthandbücher* (vgl. [13] oder [41]). Idee war die Formulierung der Vorgaben an die Durchführung von Projekten in einem verpflichtend anzuwendenden Projekthandbuch. Mit ihrer Kapitelstruktur sowie ergänzenden Erläuterungen forderten Projekthandbücher einen festen Rahmen für die in Projekten verpflichtend zu erstellende Dokumentation.

Später entstand eine erweiterte Form informeller Vorgehensmodelle, bestehend aus einem Konglomerat an Dokumenten, Foliensätzen, Checklisten, Vorlagen und/oder Handbücher. Diese wurden ergänzt um eine informelle Beschreibung zur Anwendung im Projekt. Diese Form von Vorgehensmodellen findet sich bis heute in vielen Organisationen. Sie sind historisch über Jahre gewachsen und wurden bei Bedarf ergänzt oder überarbeitet. Informelle Vorgehensmodells sind gut verständlich und intuitiv anwendbar, sie werfen jedoch bezüglich ihrer Pflege und Weiterentwicklung eine Reihe von Probleme auf. Kernprobleme ist insbesondere die Erhaltung der Konsistenz (Widerspruchsfreiheit) im Vorgehensmodell.

Inkonsistenzen in Vorgehensmodellen entstehen insbesondere bei inhaltlichen Änderungen, Erweiterungen oder Anpassungen. Ursache für die Entstehung der Inkonsistenzen sind die auf Grund des informellen Charakters des Vorgehensmodells häufig nicht zu vermeidenden Redundanzen in den unterschiedlichen Texten und Dokumenten. Bei jeder Änderung am Vorgehensmodell müssen

redundante Textstellen identifiziert und entsprechend angepasst werden. Dazu steht häufig nur rudimentäre Werkzeughilfe zur Verfügung. Der Pflege- und Weiterentwicklungsprozess für informelle Vorgehensmodelle ist dementsprechend aufwändig und zeitintensiv. Dies führt nicht selten dazu, dass informelle Vorgehensmodelle nicht regelmäßig überarbeitet und aktualisiert werden und schnell altern.

Neben informellen Beschreibungsansätzen für Vorgehensmodelle fanden sich vor allem in den 80er und 90er Jahren eine Reihe formale Techniken. Wegweisend war hier ein Artikel von Osterweil 'Software Processes are Software too' [93]. In diesem Artikel schlägt Osterweil die Verwendung von Programmiersprachen zur Beschreibung von Prozessen vor. In einer Zeit (der Artikel erschien 1987), in der vor allem informelle Prozessbeschreibungen üblich waren, kann dieser Vorschlag als ein erster Versuch in Richtung Formalisierung von Vorgehensmodellen auch im Hinblick auf Eindeutigkeit der Prozessbeschreibung und Werkzeugunterstützung und eingeordnet werden. Ähnliche Arbeiten aus dieser Zeit stammen von Fernström und Ohlsson [43] sowie Tailor et al. [110]. Ein bereits fortgeschrittener formaler Ansatz ist die Sprache APPL/A (vgl. Sutton [71]). APPL/A steht für 'Ada Process Programming Language based on Aspen' und ist eine Anpassung der Programmiersprache ADA auf die Programmierung von Softwareprozessen.

Kernproblem formaler Beschreibungssprachen ist die Projekten inhärente Parallelität der Aufgaben, die sich naturgemäß schwer mit typischerweise sequentiellen Programmiersprachen nachbilden lässt. Ein weiteres Problem ist die Starrheit dieser Sprachen. Projekte sind lebendige Gebilde, die Änderungen und Anpassungen unterworfen sind. Diese Dynamik lässt sich nur schwer mit formalen Programmiersprachen nachbilden. Letztlich konnte sich diese Form der Modellierung von Vorgehensmodellen in der Praxis nicht durchsetzen. Die meisten modernen Vorgehensmodelle gehen heute einen Mittelweg zwischen vollständiger Formalisierung und durchgängig informeller Beschreibung. Mittel der Wahl ist die Verwendung von Vorgehens-Metamodellen als Sprache für die Beschreibung der Strukturen eines Vorgehensmodells. Die Inhalte werden dagegen weiterhin in freiem Text dokumentiert. Diese Teilformalisierung erlaubt insbesondere die Verwendung von Werkzeugen für strukturelle Modellanpassungen sowie unterschiedliche Darstellungs- und Generierungsansätze.

Ein Vorgehens-Metamodell definiert die Konzepte und Beziehungen, aus denen Vorgehensmodelle aufgebaut sein dürfen. Bis heute konnte sich kein einheitlicher Standard eines Vorgehens-Metamodells durchsetzen, auch wenn die Object Management Group mit dem SPEM einen entsprechenden Versuch startete. Jedes Vorgehens-Metamodell kennt seine individuellen Konzepte sowie seine 'Philosophie' der Verknüpfungen über den Konzepten, die sich im Metamodell widerspiegelt. Auch wenn sich die Vorgehens-Metamodelle im allgemeinen im Detail stark unterscheiden, können hinsichtlich ihrer Philosophie prinzipiell zwei Arten unterschieden werden: aktivitätsorientierte und produktorientierte Vorgehens-Metamodelle. Instanzen dieser Vorgehens-Metamodelle können dementsprechend in aktivitätsorientierte bzw. produktorientierte Vorgehensmodelle einge-

teilt werden. Im Folgenden werden die kennzeichnenden Unterschiede der beiden Metamodellklassen diskutiert und entsprechende Vorgehensmodelle vorgestellt.

Aktivitätorientierte Metamodelle

Aktivitätsorientierte Vorgehensmodelle stellen Aktivitäten und die Reihenfolge ihrer Ausführung in den Vordergrund. Zwischen Aktivitäten und Produkten existieren üblicherweise zwei Assoziationen: (1) Aktivitäten *nutzen* Produkte als Eingaben und (2) Aktivitäten *produzieren* weitere Produkte als Ausgaben. Produkte, die von einer Aktivität erzeugt wurden, können als Eingabeprodukte für weitere Aktivitäten dienen. Rollen führen Aktivitäten durch um Produkte zu erzeugen. Durch diese Abhängigkeitsstruktur ist die Durchführungsreihenfolge der Aktivitäten weitgehend vorgegeben. Eine Aktivität kann durchgeführt werden, sobald alle von ihr benötigten Produkte erstellt wurden und vorliegen. Abbildung 3.2 stellt die Idee der aktivitätsorientierten Modellierung in Form eines UML Klassendiagramms dar.

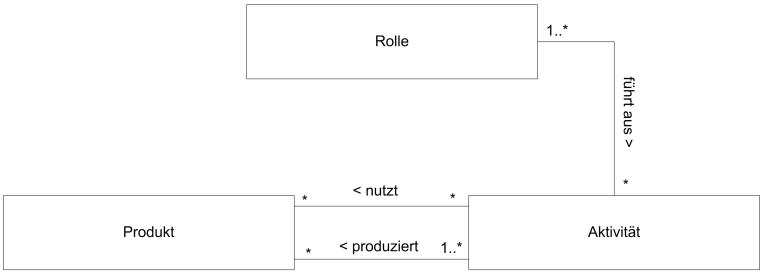

Abbildung 3.2: Aktivitätsorientierter Ansatz zur Vorgehensmodellierung

Eine Aktivität kann eine Menge von Produkten nutzen um eine Menge weiterer Produkte zu erzeugen. Die Mulitplizitäten zwischen Produkten und Aktivitäten sind dabei nicht beschränkt. Unterstützt das Metamodell zusätzlich eine Hierarchisierung von Aktivitäten, kann bei dieser Form der Vorgehensmodellierung in letzter Konsequenz eine Phase im Vorgehensmodell als eine hierarchisch aus weiteren Aktivitäten bestehende Aktivität betrachtet werden.

Bekanntester Vertreter eines aktivitätsorientierten Vorgehens-Metamodells ist das Software Process Engineering Metamodell (SPEM) [88] der OMG. Das SPEM wurde auf der Grundlage existierender Vorgehensmodelle entwickelt. Ziel war dabei nicht so sehr die Entwicklung neuer Ideen, sondern vielmehr die Zusammenführung existierender Ansätze in einem Standard. SPEM konforme Vorgehensmodelle sind nach Angaben der OMG (vgl. [88]) unter anderem der Rational Unified Process [77] (ebenso die Minimalvariante des RUP, der OpenUp Prozess [119]), DMR Macroscope, die IBM Gobal Services Method [56] und Fujitsu SDEM. Weitere aktivitätsorientierte Vorgehensmodelle, die sich jedoch nicht am SPEM orientieren, sind beispielsweise der OPEN Process [53] oder das V-Modell 97 [39].

Der aktivitätsorientierte Ansatz spiegelt die Dynamik der von ihm modellierten Prozesse wider und kann so eine detailgetreue Prozessbeschreibung liefern. Es ist jedoch nicht Aufgabe eines Vorge-

hensmodells existierende Prozesse zu beschreiben, sondern vielmehr unterstützender Leitfaden für neue Prozesse zu sein. Dies erfordert eine gewisse Flexibilität des Vorgehensmodells hinsichtlich Anpassung und Skalierung. Hier liegen die Schwächen des aktivitätsorientierten Ansatzes. Durch die starken Abhängigkeiten zwischen Aktivitäten und Produkten führen Anpassungen (beispielsweise das Löschen einzelner Aktivitäten oder Produkte) an den Vorgehensmodellen schnell zu Inkonsistenzen. Diese Form der Anpassung ist jedoch notwendiger Bestandteil der organisations- und projektspezifischen Anpassung eines Vorgehensmodells. Viele aktivitätsorientierte Vorgehensmodelle neigen jedoch auf Grund ihrer Abhängigkeiten zu einer monolithischen Struktur und bieten nur geringe Flexibilität hinsichtlich der Anpassung.

Produktorientierte Metamodelle

Produktorientierte Ansätze zur Vorgehensmodellierung, Chroust spricht auch von resultatsorientierten Ansätzen (vgl. [28]), gehen hier einen alternativen Weg. Sie stellen die Beschreibung der Ergebnisse in den Vordergrund. Die Bandbreite reicht von stark produktorientierten Modellen, die keinerlei Angaben mehr zu Aktivitäten machen, bis hin zu abgeschwächten Formen der Modellierung, die zwar Aktivitäten definieren, diese jedoch als einfache Ergänzung der Produkte betrachten. Abbildung 3.3 stellt die dem produktorientierten Ansatz zu Grund liegende Idee vor.

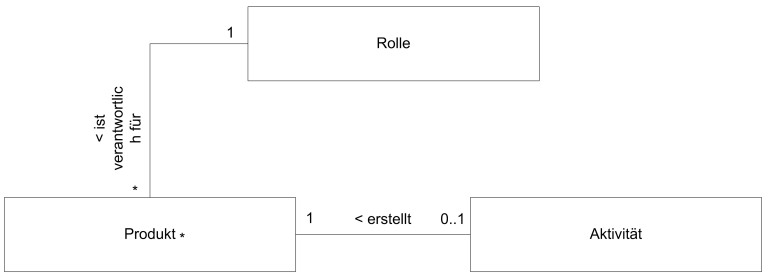

Abbildung 3.3: Produktorientierter Ansatz zur Vorgehensmodellierung

Die dominante Stellung der Produkte im Vorgehensmodell wird durch die Zuordnung verantwortlicher Rollen zum Produkt hervorgehoben. Während bei aktivitätsorientierten Ansätzen mehrere Rollen gleichberechtigt an der Erstellung eines Produkts beteiligt sein können, ordnet der produktorientierte Ansatz jedem Produkt genau eine verantwortliche Rolle zu. Eine weitere Zuordnung mitwirkender Rollen ist möglich. Unterstützt wird dieser Ansatz durch eine 1:0..1 Beziehung zwischen Produkten und Aktivitäten, die festlegt, dass ein Produkt von höchstens einer Aktivität erstellt wird (abgeschwächte Ansätze können auch eine 1..n:0..1 Beziehung zwischen Produkten und Aktivitäten erlauben) Diese Multiplizität verstärkt die untergeordnete Stellung der Aktivitäten gegenüber den Produkten. Ziel einer Aktivität ist die Erstellung eines Produkts, eine Rolle verantwortet die Erstellung des Produkts. Ein Beispiel eines produktorientierten Vorgehensmodells ist das V-Modell XT [118].

Der produktorientierte Ansatz stellt die Wichtigkeit der Prozessergebnisse, nicht die Art der Prozessdurchführung in den Vordergrund. Eine Reihenfolge der Aktivitäten und damit eine Reihenfolge der Produkterstellung wird nicht explizit vorgegeben. Eine entsprechende Planung bleibt dem Projekt überlassen. Einschränkungen hinsichtlich der Reihenfolge ergeben sich jedoch durch eine Zuordnung der Produkte zu Meilensteinen als Übergänge zwischen zwei Phasen. Das V-Modell XT kennt zusätzlich das Konzept der Produktabhängigkeiten, die inhaltliche Abhängigkeiten zwischen Produkten und ihrer Erstellungsreihenfolge explizit modellieren. Das Vorhandensein einer Menge von Produkten in einer vorgegebenen Qualität zu einem Meilenstein definiert das Übergangskriterium zur nachfolgenden Phase.

Produktorientierte Vorgehensmodelle haben den Charakter eines Prozessrahmens, nicht so sehr einer Prozessbeschreibung. Das Vorgehensmodell gibt eine grobe Richtung für die Projektabwicklung vor und definiert die erwarteten Ergebnisse. In welcher Reihenfolge in einem Projekt die Ergebnisse erarbeitet werden, ist dagegen nicht starr vorgegeben. Der produktorientierte Modellierungsansatz führt zu einer weniger starren Abhängigkeitsstruktur wie der aktivitätsorientierte Ansatz. Damit sind produktorientierte Vorgehensmodelle flexibler auf unterschiedliche Projekt- und Organisationskontexte anpassbar.

3.7 Qualitätsmanagementmodelle

Ein Qualitätsmanagementmodell definiert Anforderungen an konkrete Qualitätsmanagementsysteme in Organisationen. Die Anforderungen beziehen sich sowohl auf die Definition als auch auf die Umsetzung. Ein Qualitätsmanagementsystem ist das Zusammenwirken von Menschen, Prozessen, Methoden und Werkzeugen mit dem Ziel, Produkte und Dienstleistungen des Unternehmens ständig zu verbessern [38]. Qualitätsmanagementmodelle verfolgen ähnlich zu Vorgehensmodellen das Ziel, die Qualität von Produkten und Prozessen in Organisationen zu standardisieren und messbar zu machen. Sie unterscheiden sich von Vorgehensmodellen jedoch hinsichtlich der von ihnen verwendeten Strategie. Während Vorgehensmodelle konkrete Vorgaben an Produkte und Prozesse definieren und so der Organisation einen Leitfaden an die Hand geben wie Qualität erreicht werden kann, dienen Qualitätsmanagementmodelle vorrangig als Maßstab für die Qualität. Hierzu definieren Qualitätsmanagementmodelle ein Referenzmodell. Die Messung der Qualität erfolgt durch Vergleich der organisationsspezifischen Prozesse und Produkte zum jeweiligen Referenzmodell. Um eine unabhängige Vergleichbarkeit von Organisationen bezüglich ihrer Qualitätsvorgaben zu gewährleisten, unterstützen viele Qualitätsmanagementmodelle zusätzlich ein Assessment und/oder Zertifizierungsverfahren.

Das PDCA Modell

Eines der ersten Qualitätsmanagementmodelle überhaupt ist das von Deming entwickelte PDCA (Plan-Do-Check-Act) Modell [34], auch unter dem Namen Deming Modell bekannt. Das Modell hat eine kontinuierliche Qualitätsverbesserung zum Ziel. Qualitätsmaßnahmen werden zunächst geplant, dann nach Plan eingeführt, gemessen und bei Abweichungen korrigiert. Dieser Ablauf wird in Zyklen weitergeführt (siehe Abbildung 3.4). Das PDCA Modell ist vor allem für viele prozessorientierte Qualitätsmanagement- und Prozessverbesserungsmodelle Vorbild.

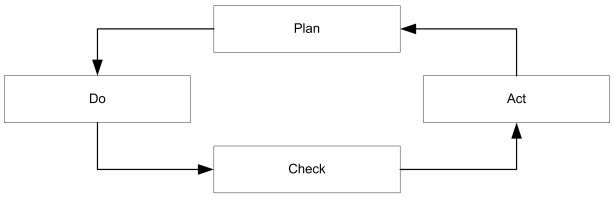

Abbildung 3.4: Das PDCA Modell nach Deming

ISO 9000:2000

Die ISO 9000 Familie ist eine Sammlung von Normen, Leitfäden und Definitionen, die in ihrer Gesamtheit die ISO 9000 Qualitätsmanagementnorm definieren. Unternehmen können sich hinsichtlich dieser Norm zertifizieren lassen. Im Zentrum der Norm steht ISO/IEC 9001:2000 [61], das Referenzmodell der ISO Norm. Das Referenzmodell definiert die Anforderungen an Unternehmen, die

ihr Qualitätsmanagementsystem nach ISO 9000 aufbauen und zertifizieren lassen wollen. Kernforderung ist die Dokumentation des QM Systems in einem QM Handbuch. Im Handbuch werden die grundsätzliche Ausrichtung der Organisation hinsichtlich Qualität, die Qualitätsziele und die Qualitätsmaßnahmen dokumentiert. Das QM Handbuch dient als Vorgabe für die Qualitätssicherung in den Projekte.

Im Rahmen der Zertifizierung wird die Vollständigkeit und Angemessenheit der Dokumentation, sowie die Umsetzung in der Organisation geprüft. Grundidee der ISO Norm sind acht Prinzipien, an denen sich die Anforderungen ausrichten. Diese behandeln Themen wie beispielsweise *Kundenorientierung*, *Prozessorientierung*, *Einbeziehung der Mitarbeiter* und *Ständige Verbesserung*. Die Norm ist branchenunabhängig. Eine ISO Zertifizierung ist in allen Bereichen der Wirtschaft möglich - von Softwarefirmen über Verkehrsbetriebe und Kliniken bis hin zu Handwerksbetrieben. Hilfestellung zur Einführung und Anwendung der ISO 9001 Norm bietet ISO/IEC 9004:2000 [60]. Eine allgemeine Einführung in die ISO Norm sowie ein Glossar und Definitionen finden sich in ISO/IEC 9000:2000 [59]. Hilfestellung für die Anwendung der ISO Norm in Softwarefirmen bietet der Zusatz ISO/IEC 9000:2000 Teil 3. Die ISO 9000 Norm ist vor allem in Europa die bekannteste Norm für ein Qualitätsmanagementsystem.

TQM und EFQM

TQM (Total Quality Managment) ist ein ganzheitliches Qualitätsmanagementkonzept, das in seinen Grundkonzepten auf dem von Deming entwickelt PDCA Ansatz beruht und bereits in den 50er Jahren erfolgreich in der Japanischen Autoindustrie etabliert wurde. TQM ist nach Jurow und Barnard [72]:

> A system of continuous improvement employing participative management and centered on the needs of customers.

Im Zentrum des TQM steht somit die Qualitätsverbesserung der Prozesse, ausgerichtet an den Kundenbedürfnissen. Deming fasste die Kernelemente seines Vorgehens zusammen und veröffentlichte 1986 in [34] 14 Schritte, mit denen TQM in einer Organisation eingeführt werden kann. Der Erfolg in Japan veranlasste die Amerikanische Regierung, TQM auch für amerikanische Firmen nutzbar zu machen. Um dies zu forcieren, wurde 1987 der Baldridge National Quality Award [117], benannt nach dem damaligen United States Secretary of Commerce, Malcolm Baldrige, ins Leben gerufen und seitdem jährlich vergeben.

Der Baldridge National Quality Award ist ein Verfahren zum Nachweis eines erfolgreichen TQM in einer Qrganisation. Der Preis wird vom National Institute of Standards and Technology ausschließlich an amerikanische Firmen vergeben. Grundlage sind eine Reihe von Kriterien zur Bewertung der Qualität. Für die Erfüllung eines Kriteriums werden Punkte vergeben. Durchgeführt wird die

Bewertung in Form von Self Assessments. Erreicht eine Firma einen bestimmten Prozentsatz, qualifiziert sie sich für die Ausscheidung um den Preis. Die Gewinner werden öffentlich bekannt gegeben.

Europäisches Gegenstück zum Baldridge National Quality Award ist der European Quality Award (bzw. der Ludwig Erhard Preis in Deutschland) vergeben von der European Foundation for Quality Management (EFQM) [120]. Der European Quality Award richtet sich am EFQM Modell aus, einer auf europäische Verhältnisse angepassten TQM Variante.

Das Bewertungssystem des EFQM Modells beruht auf 9 Kriterien. Diese unterscheiden sich nach Befähigerkriterien und Ergebniskriterien. Befähigerkriterien bewerten das Vorgehen einer Organisation, Ergebniskriterien bewerten die Ergebnisse bzw. was die Organisation erzielt. Die Kriterien werden entsprechend einem Punktesystem bewertet. Wie beim Baldrige Award wird die Bewertung in Form von Self Assessments durchgeführt. Die Gewinner werden von der EFQM ermittelt und öffentlich bekannt gegeben. Obwohl eine Reihe von Unternehmen EFQM unterstützen und sich am Self Assesment Verfahren beteiligen, konnte sich das Modell bisher in Deutschland nicht gegen die wesentlich bekanntere ISO 9000 Norm durchsetzen.

CMMI, IDEAL und SCAMPI

Das CMMI (Capability Maturity Model Integration) [111] ist ein Qualitätsmanagement- und Prozessreifegradmodell. Als Qualitätsmanagementmodell unterstützt es die Etablierung qualitativ hochwertiger Prozesse in einer Organisation durch Vorgabe eines Referenzmodells. Als Reifegradmodell gibt es ein Maß für die Bewertung der Prozessreife einer Organisation vor. Die Prozessreife entspricht der Fähigkeit einer Organisation, qualitativ hochwertige Prozesse zu definieren und zu leben.

CMMI ist eine Weiterentwicklung des 1986 vom Department of Defence (DoD) in Auftrag gegebenen Capability Maturity Model (CMM). Ziel des DoD war es, bei der Vergabe von Projekten eine gewisse Sicherheit zu haben, dass der Auftragnehmer die vertraglich festgelegten Liefervereinbarungen tatsächlich einhält. Wurde das Modell ursprünglich nur unter Zwang von den Zulieferern eingesetzt, so erkannte bald eine Reihe von Unternehmen den Nutzen einer Prozessverbesserung und begannen, das Modell freiwillig einzusetzen [76].

Entwickelt wurde CMMI am Software Enginieering Institute der Carnegy Mellon University (SEI) im Auftrag des DoD. Im Gegensatz zu ISO 9000 ist CMMI spezifisch auf den Softwareentwicklungsprozess ausgerichtet. Hierzu definiert CMMI eine Reihe von Prozessgebieten (z.B. Projektplanung, Risikomanagement, Anforderungsmanagement) und ordnet diesen verschiedene Ziele zu. Als Hilfestellung nennt CMMI eine Reihe von Praktiken, die bei der Erreichung der Ziele unterstützen.

Zur Einführung eines CMMI konformen Prozesses kann das ebenfalls vom SEI entwickelte IDEAL Modell [42] verwendet werden. Das IDEAL Modell ist ein Vorgehensmodell zur Einführung CMMI konformer Prozesse in einer Organisation. Das Modell definiert entsprechend seinem Namen (IDE-

AL = Initiating, Diagnosing, Establishing, Acting & Learning) fünf Phasen. Zu jeder Phase gibt das Modell neben dem Sinn und Zweck auch Ziele, Eingangskriterien, Ausgangskriterien und die in der Phase durchzuführenden Tasks an.

Zur Bewertung der Prozessreife definiert CMMI zwei unterschiedliche Modelle: ein stufenförmiges Modell (Staged Representation) und ein durchgängiges Modell (Continuous Representation). Das stufenförmige Modell betrachtet alle Prozesse einer Organisation im Querschnitt. Das Bewertungsschema definiert fünf Stufen der Prozessreife. Eine Organisation muss für alle Prozesse die entsprechenden Kriterien einer Stufe erfüllen um den entsprechenden Reifegrad zu bekommen. Im durchgängigen Modell werden dagegen nur einzelne Prozesse betrachtet und hinsichtlich ihrer Prozessreife eingestuft. Die Prozessreife einer Organisation wird im Rahmen einer Begutachtung (Appraisal) durchgeführt.

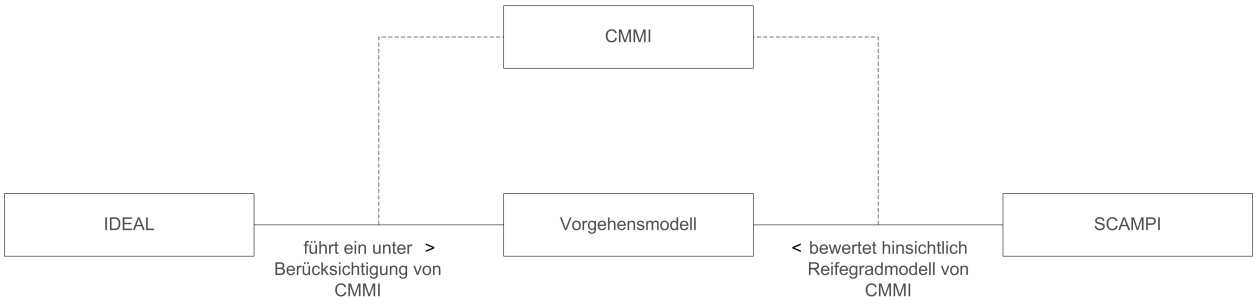

Abbildung 3.5: Die Modelle des CMMI

Als Verfahren zur Begutachtung und zur Bewertung der Prozessreife und CMMI Konformität unterstützt das SEI ein eigenes Assessmentverfahren unter dem Namen SCAMPI (Standard CMMI Appraisal Method for Process Improvement) [81]. Abbildung 3.5 zeigt die Modelle rund um CMMI im Zusammenhang.

SPICE

Ein weitere Reifegradmodell ist der ISO/IEC 15504 Standard, besser bekannt unter dem Namen SPICE (Software Process Improvement and Capability Determination). SPICE wurde im Rahmen einer Initiative zur Entwicklung eines internationalen Standards zur Prozessverbesserung entwickelt. SPICE ist wie CMMI ein Qualitätsmanagement- und Reifegradmodell, geht jedoch im Gegensatz zu CMMI ausschließlich von einer kontinuierlichen (durchgängigen) Reifeskala aus.

SPICE ist eine Sammlung von Dokumenten, die in in ihrer Gesamtheit einen Rahmen für die Bewertung von Lieferanten und der eigenen Organisation bieten. 1998 wurden die Dokumente als ISO/IEC 15504:1998 in den Standardisierungsprozess der ISO aufgenommen. Anfang 2000 wurde der Standard überarbeitet und aktualisiert.

Aktuell umfasst SPICE fünf Teile: Part 1 (Concepts and Vocabulary [65]) enthält allgemeine Informationen zum Modell. Vorgaben und Hilfestellung für die Durchführung von Assessments liefern

Part 2 (Performing an assessment [65]) und Part 3 (Guidance on performing an assessment [66]). Part 4 beschäftigt sich mit der Prozesseinführung und Prozessbewertung (Guidance on use for process improvement and process capability determination [64]). Eine Zusammenfassung der Standards und beispielhafte Anwendung findet sich schließlich in Part 5 (An exemplar Process Assessment Model [67]). Umgangssprachlich wird SPICE häufig mit Part 5 des Standards gleichgesetzt. SPICE orientiert sich in seiner Bewertung an Referenzmodellen. Diese legen den Bereich für die Anwendung des Assessmentverfahrens fest. Als Grundlage für das Assessment von Softwareentwicklungsprozessen verwendet SPICE beispielsweise den ISO/IEC 12207 Standard [62]; andere Referenzmodelle sind jedoch auch möglich.

Weitere Modelle

Die vorgestellten Qualitätsmanagementmodelle sind die heute am weitesten verbreiteten Modelle. Daneben finden sich in der Literatur und Praxis viele weitere, häufig spezifisch auf einen Kontext ausgerichtete Modelle. Zu nennen ist beispielsweise die IT Infrastructure Library (ITIL) [52], ein Qualitätsmodell für den Aufbau und Einsatz von IT-Infrastrukturen. Das Modell befasst sich neben dem Service Level Management auch mit Themen wie Problem- und Änderungsmanagement oder Kapazitätsmanagement. ITIL ist ein britischer Standard und wird vom Office of Government Commerce (OGC) verwaltet.

Ein Qualitätsmanagementmodell, das ursprünglich aus dem Produktionsbereich stammt, heute jedoch auch im Dienstleistungsbereich Verwendung findet, ist das Six Sigma Modell [54]. Das Sigma entspricht dabei der Standardabweichung der Gaußschen Normalverteilung. Aus der Anzahl der Fehler in einem Prozess kann mit Hilfe von Tabellen oder Statistikprogrammen das Sigma-Niveau ermittelt werden. Six Sigma drückt im statistischen Sinne aus, dass unter einer Millionen Fehlermöglichkeiten weniger als 4 Fehlerfälle tatsächlich eintreten.

Andere Modelle konzentrieren sich stärker auf die Methodik zur Prozesseinführung und -verbesserung. Zu nennen ist hier beispielsweise das Agile Deployment Framework [94], ein Ansatz zur Einführung agilen Praktiken in einer Organisation, der sich auf das Quality Improvement Paradigm (IQM) stützt. Des Weiteren finden sich eine Reihe metrikbasierter Ansätze. Diese stützen sich auf der Verwendung von Metriken als Grundlage der Prozessanalyse und -verbesserung, wie beispielsweise die Improvement Methodology von Basili und Rombach [11]. Dieser Ansatz sieht fünf Schritte vor, die einer Erweiterung des PDCA Modells entsprechen. Ziel der Analyse- und Verbesserungsaktivitäten ist dabei vorrangig die Identifizierung, Aufbereitung und Bewertung von Daten als Grundlage der Prozessbewertung und Prozessverbesserung.

Ein einfacher evolutionärer Ansatz zur Prozessverbesserung findet sich in [17]. Der Ansatz konzentriert sich auf die sozioemotionale Seite einer Prozesseinführung. Insbesondere wird in diesem Ansatz den bei einer Prozesseinführung typischerweise auftretenden Ängste, wie Kompentenzverlust, Machtverlust, Angst vor Änderungen, Angst um den Arbeitsplatz gezielt im Rahmen der

Prozesseinführung entgegen gesteuert. Damit adressiert dieser Ansatz ein zentrales Problem der Prozessverbesserung: die mangelnde Akzeptanz der Anwender. Mit diesem für den Erfolg eines Prozessverbesserungsprojekts entscheidenden Aspekt beschäftigt sich beispielsweise auch [113] (The Human Dimension of Software Process) oder [2] (How Things Should Not Be Done: A Real World Horror Story of Software Engineering Process Improvement). Die dort vorgestellten Beispiele zeigen eindrücklich, welche Probleme eine nicht erfolgreiche Prozessverbesserung charakterisieren.

3.8 Zusammenspiel der Modelle

In diesem Kapitel wurden eine Reihe unterschiedlicher Modelle vorgestellt, die im Zusammenhang mit Prozessqualität und Prozessverbesserung innerhalb einer Organisation eine wichtige Rolle spielen. Konkret können folgende Modelltypen unterschieden werden: Qualitätsmanagementmodelle, Prozessreifegradmodelle, Prozesseinführungsmodelle, Prozessbewertungsmodelle und Vorgehensmodelle. Jeder der genannten Modelltypen erfüllt eine festgelegte Aufgabe bezüglich Prozessbewertung und Prozessverbesserung. Gemeinsam unterstützten die Modelle die Etablierung qualitativ hochwertiger Prozesse und ihre kontinuierliche Verbesserung.

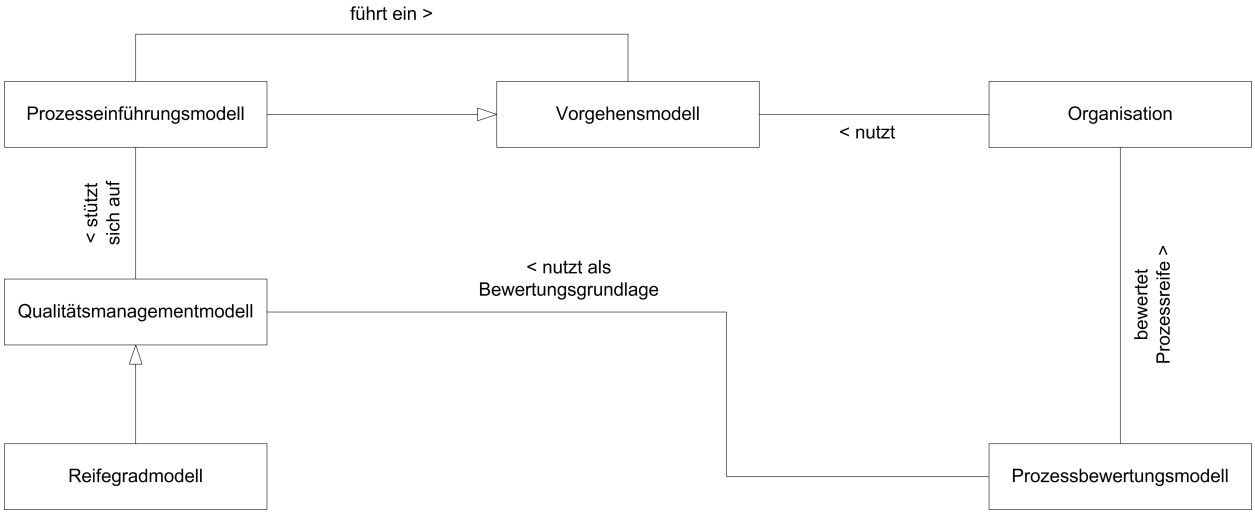

Abbildung 3.6: Zusammenspiel der Modelle

Abbildung 3.6 stellt anhand eines UML Klassendiagramms die Modelle im Zusammenhang dar. Im Zentrum steht eine Organisation, die die Qualität ihrer Prozesse und damit die Qualität ihrer Produkte verbessern möchte. Hierzu nutzt sie ein Vorgehensmodell. Mit Hilfe des Vorgehensmodells kann sie ihre Prozesse explizit definieren und Vorgaben für die Qualität der Ergebnisse festlegen. Die Organisation erhält so eine Basis der Bewertung ihrer Prozessqualität.

Ein Vorgehensmodell muss, damit es sinnvoll anwendbar ist, im Rahmen eines Einführungsprojekts explizit an die Bedürfnisse einer Organisation angepasst werden. Leitfaden für die Durchführung von Einführungsprojekten sind so genannte Prozesseinführungsmodelle. Prozesseinführungsmo-

delle sind spezialisierte Vorgehensmodelle, die nicht die Entwicklung eines Systems sondern die Einführung eines Vorgehensmodells im Fokus haben. Sie modellieren explizit Ablauf und Inhalt von Einführungsprojekten. Prozesseinführungsmodelle können sich auf spezielle Qualitätsmanagementmodelle abstützen, d.h. bei der Einführung des Vorgehensmodells werden die Anforderungen des Qualitätsmanagementmodells berücksichtigt. Ein Qualitätsmanagementmodell definiert Anforderungen an das eingeführte Vorgehensmodell aus qualitativer Sicht. Es macht Vorgaben zur Prozess- und zur Produktqualität.

Eine spezielle Form der Qualitätsmanagementmodelle sind Reifegradmodelle. Sie geben einen nach Prozessreifegraden gestaffelten Qualitätsrahmen für das in einer Organisation eingeführte Vorgehensmodell vor. Das Vorgehensmodell und die Organisation können entsprechend der Reifegrade zertifiziert werden. Zur Ermittlung der tatsächlichen Prozessreife in einer Organisation wird ein Prozessbewertungsmodell eingesetzt. Prozessbewertungsmodelle definieren konkrete Assessmentverfahren. Im Rahmen eines Assessmentverfahrens werden Qualität und Anwendung der im Vorgehensmodell definierten Prozesse innerhalb der Organisation bewertet. Als Grundlage der Bewertung wird ein Referenzmodell als Teil eines Qualitätsmanagementmodells verwendet.

Auch wenn Reifegradmodelle und Qualitätsmanagementmodelle teilweise als Vorgehensmodelle bezeichnet werden, so handelt es sich doch um unterschiedliche Konzepte mit unterschiedlicher Zielrichtung. Im Gegensatz zu Vorgehensmodellen eignen sich Qualitätsmangementmodelle und Reifegradmodelle für den objektiven Vergleich von Organisationen hinsichtlich ihrer Prozessqualität. So können beispielsweise zwei Organisationen nach unterschiedlichen Vorgehensmodellen arbeiten, erfüllen jedoch beide CMMI Level 3. Diese Objektivität ist einer der Gründe, dass immer häufiger bei der Vergabe von Projekten der Prozessreifegrad einer Zulieferfirma von entscheidender Bedeutung ist. Das in einer Organisation konkret verwendete Vorgehensmodell spielt dagegen nur eine untergeordnete Rolle.

Kapitel 4

Methoden

Im Fokus dieser Arbeit stehen sowohl Vorgehensmodelle als auch Methoden. Während jedoch Vorgehensmodelle in Literatur und Forschung bereits gut verstanden sind, gibt es bisher noch kein einheitliches Verständnis darüber, was unter einer Methode zu verstehen ist bzw. wie eine Methode aufgebaut ist. Auch findet häufig keine eindeutige Abgrenzung von Methoden zu verwandten Konzepten wie Heuristiken, Prinzipien oder Verfahren statt. In diesem Kapitel werden nun Methoden systematisch untersucht. Ausgehend von einer Charakterisierung und Abgrenzung der hier betrachteten Methodenmenge, werden die relevanten Konzepte einer Methode identifiziert und beschrieben, danach wird ein für diese Arbeit gültiger Methodenbegriff definiert.

Das Kapitel ist wie folgt aufgebaut: Als empirische Basis stellt Abschnitt 4.1 eine Menge von Beispielmethoden gruppiert nach Teilprozessen in Projekten vor. Abschnitt 4.2 führt auf Basis dieser repräsentativen Methodenmenge eine im Rahmen dieser Arbeit gültige Methodendefinition ein. Die Methodendefinition identifiziert insbesondere die für eine Methode relevanten Konzepte. In den folgenden Abschnitten werden die identifizierten Konzepte im Detail diskutiert und ihre Relevanz für Methoden anhand von Beispielen evaluiert: Abschnitt 4.3 beginnt mit Verfahren und Algorithmen als Basis der Prozessbeschreibung einer Methode. Verfahren können ergänzt werden um Prinzipien und Heuristiken; diese Konzepte werden in Abschnitt 4.4 eingeführt und ihre Rolle in Methoden untersucht. Eine Sprache zur Formulierung der Prozessschritte im Verfahren und ihrer Ergebnisse liefert eine Domänentheorie. Die Dokumentation der Ergebnisse erfolgt anhand von Notationen oder domänenspezifischen Sprachen. Domänentheorien, domänenspezifische Sprachen und Notationen werden in Abschnitt 4.5 diskutiert. Methoden können, auch wenn sie ähnliche Probleme adressieren, große Unterschiede hinsichtlich der Wahl von Domänentheorie und Verfahren aufweisen. Diese Unterschiede sind auf Einflussfaktoren bei der Entwicklung der Methoden zurückzuführen. Solche Einflussfaktoren sind insbesondere Paradigmen und Theorien, die in Abschnitt 4.6 vorgestellt werden. In Abschnitt 4.7 wird anhand der identifizierten Konzepte eine Methodenontologie als erster Schritt hin zu einer Formalisierung von Methoden entwickelt und eine einfache, auf Verfahren und Domänentheorien basierende Klassifikation von Methoden vorgestellt. Eine Abgrenzung von Methoden zu Vorgehensmodellen sowie eine Konkretisierung der Begriffe

methodenspezifisch und *methodenneutral* liefert Abschnitt 4.8. Abschnitt 4.9 fasst die Ergebnisse des Kapitels zusammen.

4.1 Methoden der Softwareentwicklung

Der Begriff der Methode (*μέθοδοσ*) geht in seinen Ursprüngen auf das Altgriechische zurück und bedeutet in etwa: *das Nachgehen* bzw. *der Weg zu etwas hin.* [1] definiert eine Methode als

> '.. ein nach Mittel und Zweck planmäßiges Verfahren, das zu technischer Fertigkeit bei der Lösung theoretischer und praktischer Aufgaben führt'.

Eine Methode beschreibt Schritt für Schritt, welche Tätigkeiten durchzuführen und welche Ergebnisse zu erstellen sind, um ein spezifisches Problem zu lösen.

In dieser Arbeit werden vorrangig Methoden betrachtet, die im Rahmen der Softwareentwicklung Anwendung finden. Konkret sind dies alle Methoden, die in irgendeiner Form die Planung und Durchführung von Entwicklungsprojekten in Organisationen unterstützen. In diesem Abschnitt wird die Domäne der hier betrachteten Methoden anhand von Beispielen näher charakterisiert und eine Methodendefinition abgeleitet.

4.1.1 Methoden im Projekt

Ein Softwareentwicklungsprojekt ist ein Prozess mit dem Ziel der Erstellung eines Softwaresystems (vgl. Abschnitt 3.2). Methoden werden im Rahmen eines Projekts durchgeführt, um für das Projekt relevante Ergebnisse zu erstellen. In diesem Abschnitt werden anhand der zentralen Teilprozesse in Projekten typische Methoden der Softwareentwicklung vorgestellt. Ziel ist dabei eine einführende informelle Charakterisierung der in dieser Arbeit betrachteten Methoden.

Methoden der Anforderungsanalyse

Ziel der Anforderungsanalyse ist die Erfassung, Aufbereitung und Dokumentation der Anforderungen an das im Rahmen eines Projekts zu entwickelnde System. Die Anforderungsanalyse kann unterteilt werden in drei Teilprozesse mit ihren spezifischen Methoden:

- Erfassung der Anforderungen: Im Rahmen der Anforderungserfassung werden in Zusammenarbeit mit den späteren Anwendern bzw. dem Fachbereich die funktionalen und nichtfunktionalen Anforderungen an das System ermittelt und in Kurzform dokumentiert.
- Strukturierung und Abstimmung: Die ermittelten Anforderungen werden nun verfeinert, dokumentiert und mit den Anwendern abgestimmt.

- Prüfung und Bewertung: Die dokumentierten Anforderungen werden abschließend einer Prüfung und einer Bewertung unterzogen, beispielsweise hinsichtlich Vollständigkeit, Notwendigkeit, Plausibilität und Finanzierbarkeit.

Ergebnis des Teilprozesses ist eine konsistente, vollständige und qualitativ gut ausgearbeitete Anforderungsspezifikation, die als Basis der Systementwicklung verwendet werden kann. Der Prozess der Anforderungsanalyse kann punktuell oder durchgängig durch konkrete Methoden unterstützt werden. Typische Methoden, die vor allem im Rahmen der Anforderungserfassung Verwendung finden, sind unter anderem Moderationsmethoden oder Interviewmethoden. Im Rahmen der Strukturierung und Abstimmung der Anforderungen finden dagegen vor allem Methoden zur Anforderungsdokumentation Verwendung. Bekannte Methoden sind hier beispielsweise CRC Cards [4], die Beschreibung als Szenarien oder die Darstellung als Anwendungsfalldiagramme.

Neben den genannten, punktuell einsetzbaren Methoden, existieren Methoden, die den Teilprozesse zur Anforderungsanalyse methodisch abdecken. Zu nennen sind hier beispielsweise die Methoden Structured Analysis [33], Object Oriented Analysis [29] oder die Analysemethode des OEP [91]. In der Praxis setzen sich allerdings verstärkt werkzeugunterstützte Rundumlösungen für die Anforderungdokumentation, -verwaltung und -verfolgung mit durchgängiger Methodik durch (z.B. Telelogic Doors, CaliberRM).

Methoden im Entwicklungsprozess

Die Softwareentwicklung entspricht dem zentralen Teilprozess in einem Entwicklungsprojekt. Der Teilprozess umfasst den gesamten Entwicklungsprozess des im Projekt zu erstellenden Softwaresystems vom Entwurf über die Entwicklung bis hin zur Abnahme. Teilprozesse innerhalb der Softwareentwicklung sind:

- Der Entwurf: Ausgehend von den Anforderungen wird ein Entwurf des zu entwickelnden Systems erstellt. Der Entwurf muss die Umsetzung aller funktionalen und nicht-funktionalen Anforderungen an das zu entwickelnde System geeignet unterstützten.

- Die Softwareentwicklung: Im Rahmen der Softwareentwicklung wird das System entsprechend dem Entwurf implementiert und integriert.

- Aufbau der Entwicklungsumgebung: Dieser Teilprozess wird parallel zu Systementwurf und Softwareentwicklung durchgeführt. Seine Aufgabe ist der Entwurf, die Bereitstellung und die Pflege der Entwicklungsumgebung, sowie die Definition des Entwicklungsprozesses.

Für den Teilprozess der Softwareentwicklung existieren eine Vielzahl an Methoden von unterschiedlicher Granularität und Zielrichtung. Die meisten Methoden konzentrieren sich auf den Teilprozess

des Softwareentwurfs. In dieser Kategorie finden sich beispielsweise die verschiedenen UML basierten Entwurfsmethoden wie Catalysis [40], KobrA [6], UML Components [26] oder der komponentenorientierte Entwurf nach Andresen [5]. Ebenfalls in dieser Kategorie finden sich formale Entwurfsmethoden wie Focus [21] oder die Vienna Development Method VDM [70] und viele mehr. Alle der genannten Methoden decken den Teilprozess des Systementwurfs weitgehend vollständig ab. Daneben gibt es Methoden, die nur punktuell zum Einsatz kommen, beispielsweise die Methode Design by Contract ([82]) zur Spezifikation von Komponentenschnittstellen oder die Entity-Relationship Modellierung [27] zum Datenbankentwurf.

Methoden für den Teilprozess der Softwareentwicklung unterstützen dagegen in irgendeiner Weise die Entwicklung und Pflege des Codes im Projekt. Zu nennen sind hier beispielsweise die verschiedenen Refaktoring Methoden [45], die zur Steigerung der Codequalität eingesetzt werden. Ebenfalls als Methode der Softwareentwicklung kann das aus dem Bereich der agilen Methoden bekannte Pairprogramming (vgl. [31]) genannt werden. Für den Teilprozess der Erstellung und Pflege der Entwicklungsumgebung existieren dagegen nur wenig referenzierbare Methoden.

Methoden des Projektmanagements

Anforderungsanalyse und Entwicklungsprozess stehen im Zentrum eines jeden Entwicklungsprojekts. Querschnittsprozesse dienen zur Unterstützung dieser Prozesse, liefern jedoch keinen unmittelbaren Beitrag zu dem zu entwickelnden Softwaresystem. Typische querschnittliche Teilprozesse betreffen beispielsweise das Projektmanagement und die Qualitätssicherung. Das Projektmanagement ist der zentrale Teilprozess zur Organisation, Planung und Steuerung von Projekten. Aufgabe des Projektmanagements ist es, ein Projekt im Rahmen der gegebenen finanziellen Mittel in der gegebenen Zeit so durchzuführen, dass es als Ergebnis ein Softwaresystem mit der geforderten Leistung liefert. Teilprozesse des Projektmanagements sind:

- Die Projektplanung und -steuerung: Im Rahmen der Projektplanung und -steuerung wird der zeitliche und finanzielle Rahmen zu Beginn eines Projekts festgelegt und zur Projektlaufzeit in regelmäßigen Abständen überwacht. Es wird der Projektaufwand abgeschätzt und es werden Projektrisiken identifiziert und überwacht.

- Die Projektorganisation: Aufgabe der Projektorganisation ist die Definition der in einem Projekt relevanten organisatorischen Aspekte. Dazu zählen beispielsweise die Besetzung der relevanten Rollen, die Festlegung von Eskalationsstrategien und die Organisation der Berichterstattung.

Methoden zum Projektmanagement haben sich bisher vor allem im Bereich der Projektplanung und -steuerung etabliert. Zu nennen sind hier die verschiedenen Schätzmethoden wie Function Point Analyse [58] oder CoCoMo II [19], sowie Planungsmethoden wie die Netzplantechnik [105] und

ihre Varianten. Ebenfalls als Methode der Projekplanung kann die Meilensteintrendanalyse [137] eingeordnet werden. Die Meilensteintrendanalyse ist eine Methode zur Überwachung des Projektfortschritts, um Terminverzögerungen in Projekten frühzeitig erkennen zu können.

Zur Projektorganisation existieren dagegen vorrangig umfassende Projektmanagementmethoden wie beispielsweise Prince2 [92]. Bei dem britischen Projektmanagementstandard Prince2 handelt es sich um eine Projektmanagementmethode, die sowohl den Teilprozess zum Projektmanagement mit Produkten, Aktivitäten, Meilensteinen und Rollen definiert, wie auch die entsprechenden Methoden zur Erarbeitung der Produkte vorgibt. Daneben gibt es weitere, in der Regel firmenspezifische, Projektmanagementmethoden.

Methoden der Qualitätssicherung

Der Teilprozess der Qualitätssicherung ist als Ergänzung zum Projektmanagement zu sehen. Sein Ziel ist die Lieferung des zu entwickelnden Systems nicht nur in vorgegebener Zeit und Budget sondern auch mit der erforderlichen Qualität. Die Qualitätssicherung umfasst folgende Teilprozesse:

- Die Durchführung von Prüfungen: Ein zentraler Aspekt der Qualitätssicherung ist die Durchführung von Prüfungen der Systemelemente. In einem Projekt finden je nach Entwicklungsstadium des Systems Modulprüfungen, Integrationsprüfungen oder Systemprüfungen statt. Weitere Prüfungen betreffen Dokumente und Prozesse. Ein Prüfprozess läuft immer nach einem ähnlichen Schema ab: (1) Vorbereitung der Prüfumgebung, (2) Definition der Prüffälle, (3) Vorbereitung der Prüfdaten, (4) Durchführung der Prüfung, (5) Auswertung des Prüfergebnisses.
- Die Organisation der Qualitätssicherung: Das Vorgehen zur Qualitätssicherung muss in einem Projekt ähnlich zum Projektmanagement organisiert werden. Typische Aufgaben der QS Organisation ist die Festlegung der Qualitätsziele im Projekt, die Planung und Organisation der entsprechenden Maßnahmens sowie die Überwachung ihrer Durchführung.

Kritisch für den Erfolg der Qualitätssicherung im Projekt ist die Definition geeigneter Prüffälle. Die Wahl der Prüffälle muss sicherstellen, dass möglichst viele potentielle Fehler bei einem möglichst geringem Prüfaufwand gefunden werden können. Die meisten Methoden im Bereich der Qualitätssicherung unterstützen daher den Prozessschritt der Prüffallerstellung und Prüfdatenaufbereitung innerhalb des Teilprozesses zur Durchführung von Prüfungen. Die entsprechenden Methoden können grob in drei Bereiche eingeteilt werden:

1. Einige der Methoden unterstützen die methodische Aufbereitung der Testdaten, um mit möglichst wenig Testfällen eine möglichst gute Testabdeckung zu erreichen. Entsprechende Methoden sind beispielsweise der Äquivalenzklassentest oder der Grenzwerttest. Der Äquivalenzklassentest sieht eine Aufteilung der Testdaten in Äquivalenzklassen vor. Für jede Klasse

wird ein Vertreter gewählt, der durch einen Testfall abgedeckt ist. Es wird angenommen, dass im Falle eines Erfolgs alle Daten der Äquivalenzklasse erfolgreich getestet werden können. Die Grenzwertanalyse geht ähnlich vor. Hier werden jedoch Vertreter explizit aus den Randbereichen der Äquivalenzklassen gewählt, da aus der Erfahrung heraus hier die häufigsten Fehler auftreten.

2. Eine andere Klasse von Testmethoden unterstützt die methodische Abdeckung der Abläufe im Code, um mit der Auswahl der Testfälle möglichst viele Variationen zu simulieren. Entsprechende Methoden sind beispielsweise der Anweisungsüberdeckungstest oder der Zweigüberdeckungstest. Beim Anweisungsüberdeckungstest werden alle Anweisungen im Code (Knoten im Kontrollfluss) von mindestens einem Testfall abgedeckt, beim Zweigüberdeckungstest werden auch alle bedingten Abläufe (Zweige im Kontrollfluss) innerhalb der Anweisungen von mindestens einem Testfall abgedeckt.

3. Eine weitere Gruppe von Methoden adressiert die Auswahl relevanter Testfälle. In dieser Kategorie können beispielsweise Black-box Test, White-box Test oder Error Guessing eingeordnet werden. Der Black-box Test betrachtet nur die Schnittstelle einer Komponente. Die Testfälle decken ausschließlich das an der Schnittstelle sichtbare Verhalten ab. White-box Tests decken dagegen auch Teile der Ablaufstruktur innerhalb einer Komponente ab. Error Guessing ist eine Methode, bei der Testfälle rein empirisch ausgewählt werden. Systemexperten wählen Testfälle, die aus der Erfahrung heraus als kritisch angesehen werden.

Die Organisation der Qualitätssicherung ist ähnlich zur Organisation des Projektmanagement stark abhängig vom jeweiligen Kontext. Zur methodischen Unterstützung werden den Projekten daher auch hier häufig Vorlagen in Form von Qualitätshandbüchern mit Angabe einfacher Methoden zur Erarbeitung der Inhalte zur Verfügung gestellt.

Weitere Methoden in Entwicklungsprojekten

Die meisten der in der Literatur verfügbaren Methoden unterstützen einen der bisher genannten Teilprozesse. In einem Projekt finden sich jedoch viele weiterer Teilprozesse, wie beispielsweise das Konfigurations- und Änderungsmanagement, die Projektvergabe oder der Teilprozess Messung und Analyse zur Verwaltung und Durchführung von Prozessmessungen. Als einige Methodenbeispiele können genannt werden:

- Die Goal Question Metrics Methode [10] zur Ermittlung von Kennzahlen im Rahmen des Teilprozesses Messung und Analyse.

- Die Richtwertmethode [74] zur Bewertung von Angebotskandidaten im Rahmen des organisationsspezifischen Vergabestandards UfAB.

- Die in Bug Tracking Werkzeugen vordefinierten Methoden zur Verwaltung und Verfolgung von Problemen oder Änderungsanträgen.
- Die in Konfigurationsmanagementwerkzeugen integrierten Methoden für eine zustandsbasierte Verwaltung und Versionierung von Projektergebnisse.

Insgesamt existieren jedoch zu diesen, weniger im Vordergrund stehenden Teilprozessen der Softwareentwicklung nur wenige in der Literatur refenzierbare Methoden.

4.1.2 Empirische Methodenbasis

Für diese Arbeit werden beispielhaft einige der genannten Methoden näher untersucht. Die gewählten Methoden dienen als empirische Methodenbasis für diese Arbeit und sind im Folgenden kurz gelistet. Eine ausführliche Beschreibung der Methoden findet sich in Anhang A. Als Beispielmethoden wurden herangezogen:

Function Point Analyse: Die Function Point Analyse (FPA) ist eine Messmethode zur Ermittlung eines objektiven Maßes für die Leistung eines Softwaresystems. Die FPA kann als eigenständige Methode verwendet werden, häufig werden ihre Ergebnisse in weiteren Methoden verwendet, beispielsweise von der Methode CoCoMo II. Die FPA ist in Abschnitt A.1 beschrieben.

CoCoMo II: Die Methode CoCoMo II wurde als Beispiel einer typischen Methode zur Aufwandsschätzung in die Auswahl mit aufgenommen. CoCoMo II stützt sich auf ein allgemeines Schätzmodell, das speziell auf die Schätzung von Informationssystemen ausgerichtet ist, sowie eine Formel zur Berechnung des erforderlichen Aufwands in Personenmonaten. Als objektives Maß für den Umfang können Function Points herangezogen werden. Diese werden in das allgemeinere Maß der Source Lines of Code umgewandelt und dienen als Grundlage der Größe eines Schätzproblems. Die Methode ist in Abschnitt A.2 beschrieben.

Netzplantechnik: Die Netzplantechnik ist der heute wohl bekannteste und am häufigsten eingesetzte Standard zur Projektplanung. Die Netzplantechnik definiert als ISO Standard eine Art Methodenrahmen für Planungsmethoden. Dazu gibt sie die prinzipiellen Konzepte und Vorgehensweisen zur Projektplanung vor. Grundlage der Netzplantechnik ist die Graphentheorie. Die Netzplantechnik selbst ist Ausgangspunkt für eine Menge von spezialisierten Planungsmethoden wie beispielsweise der Critical Path Method (CPM), der Project Evaluation and Review Technique (PERT) oder der Metra Potiential Methode (MPM). Diese Methoden sind in Abschnitt A.3 beschrieben.

Komponentenbasierte Entwicklung nach Andresen: Die wohl größte Klasse der Methoden in der Systementwicklung sind die Entwicklungsmethoden selbst, speziell die verschiedenen UML basierten Entwicklungsmethoden. Als ein Vertreter dieser Methoden wurde die Methode von Andresen zur komponentenbasierten Softwareentwicklung in die Methodenauswahl aufgenommen. Sie kann als stellvertretend für die sehr große Klasse der UML Entwicklungsmethoden betrachtet werden, wie beispielsweise [101], [20], [68], [40], [40], [6], [26]. Die Methode ist in Abschnitt A.4 beschrieben.

Focus: Als ein Vertreter einer formalen Methode wurde die Entwicklungsmethode Focus gewählt. Die Methode unterstützt einen durchgängigen modellbasierten Ansatz zum Systementwurf. Focus kann dabei ähnlich zur Netzplantechnik als generische Methode eingeordnet werden, die als Grundlage für die Entwicklung weiterer spezialisierter Methoden dient. Die Methode ist in Abschnitt A.5 beschrieben.

UfAB: Neben den bisher genannten generischen Methoden, wurde ein Vertreter einer kontextspezifischen Methode in der Auswahl mit aufgenommen, der Ausschreibungsstandard für deutsche Behörden, kurz UfAB genannt. Die Methode beschreibt im Detail die Vorgehensweise zur Durchführung von Vergabeprojekten im Behördenbereich unter Berücksichtigung der Unterschiede von EU-weiten und nationalen Vorgaben für die Vergabe. Die Methode ist in Abschnitt A.6 beschrieben.

Bei der Auswahl der Methoden wurde darauf geachtet, sowohl einfache wie komplexe, sowohl formale wie informelle Methoden mit aufzunehmen um so einen repräsentativen Querschnitt zu erhalten. Ein weiteres Kriterium für die Methodenauswahl war die Referenzierbarkeit der einzelnen Methoden. So liegt für jede der gewählten Methoden eine referenzierbare und allgemein verfügbare Dokumentation vor.

4.2 Methodendefinition

In diesem und in den folgenden Abschnitten wird anhand der betrachteten Methodenmenge eine allgemeine Definition für Methoden abgeleitet und auf Methoden der Softwareentwicklung hin konkretisiert. Es werden die Kernkonzepte einer Methode identifiziert und mit ihren Abhängigkeiten im Detail beschrieben. Abschließend fasst eine Ontologie die erarbeiteten Ergebnisse graphisch zusammen.

Bezeichnend für die Menge der hier betrachteten Methoden ist ihre Heterogenität. So weisen diese Methoden zum Teil große Unterschiede hinsichtlich ihrer inhaltlichen Abdeckung und Breite auf. Eine Entwicklungsmethode wie die Methode von Andresen (vgl. Abschnitt A.4) deckt beispielsweise mehrere Phasen im Lebenszyklus eines Softwaresystems ab. Sie beginnt bei der Erfassung der Anforderungen, geht über den Systementwurf bis hin zum Feindesign und deckt teilweise die Entwicklung mit ab. Für jeden Abschnitt braucht es eine individuell angepasste, methodische Vorgehensweise. Entwicklungsmethoden können somit eher als eine Menge von ideal aufeinander abgestimmten Methoden betrachtet werden. Eine Schätzmethode wie CoCoMo II (vgl. Abschnitt A.2) wird dagegen für einen sehr spezifischen Zweck in einem Projekt eingesetzt - die Aufwandsschätzung - und kommt nur punktuell im Rahmen der Planung zum Einsatz.

Eine Methode, sei sie komplex wie eine Entwicklungsmethode oder einfach wie eine Schätzmethode, entspricht immer einer Prozessbeschreibung zur Lösung eines spezifischen Problems. Die Entwicklungsmethode beschreibt einen methodischen Weg zur Lösung des für Entwicklungspro-

jekte typischen Problems einen für die gegebenen Anforderungen geeigneten Systementwurf zu erstellen. Eine Schätzmethode beschreibt dagegen einen methodischen Weg zur Ermittlung des für die Entwicklung erforderlichen Aufwands. Wir definieren nun in einem ersten Schritt eine Methode wie folgt:

> **Definition:** Eine Methode ist eine dokumentierte Handlungsanweisung zur wiederholbaren und nachvollziehbaren Lösung eines spezifischen Problems bzw. einer spezifischen Fragestellung ihrer Anwendungsdomäne. Eine Methode beschreibt im Detail den Prozess zur Erstellung aller notwendigen (Zwischen-) Ergebnisse, die das Problem lösen bzw. die Fragestellung klären.

Die Definition gibt vor, welche Konzepte mindestens eine Methode auszeichnen - Prozessbeschreibung und Angabe der erforderlichen Ergebnistypen - sie macht jedoch keine Aussagen hinsichtlich der Granularität bzw. des Detaillierungsgrades der Beschreibungen. So kann eine Methode, die dieser Definition entspricht, in ihrer Prozessbeschreibung oberflächlich bleiben oder aber akribisch jeden Prozessschritt definieren. Gleichzeitig schränkt diese Definition die Menge der betrachteten Methoden stark ein. Mit der Forderung nach einer schriftlichen Dokumentation zählen beispielsweise mündliche Überlieferungen oder gewohnheitsmäßige Handlungen nicht als Methoden im Sinne dieser Definition.

In dieser Arbeit wird eine spezielle Menge von Methoden betrachtet, die Methoden der Softwareentwicklung. Für diese Methodenmenge konkretisieren wir unter Berücksichtigung der in Abschnitt 4.1 eingeführten empirischen Methodenbasis die Definition wie folgt:

> **Definition:** Eine Methode der Softwareentwicklung ist eine Methode, die ein Problem bzw. eine Fragestellung innerhalb der Anwendungsdomäne der Durchführung von IT Softwareentwicklungsprojekten löst. Sie
>
> - implementiert je nach Umfang des Problems ein oder mehrere Verfahren zur Problemlösung,
> - stützt sich zur Beschreibung der Verfahren auf geeignete Domänentheorien,
> - bietet zur Dokumentation der Ergebnisse geeignete Notationen mit Abbildung in die Domänentheorie,
> - definiert über Prinzipien einen konzeptionellen Rahmen für das Verfahren,
> - macht über Heuristiken punktuelle Vorschläge zur Durchführung einzelner Prozessschritte im Verfahren.

Die in der Definition eingeführten Konzepte werden in den folgenden Abschnitten im Detail erläutert und ihre Rolle für Methoden anhand von Beispielen nachgewiesen.

4.3 Verfahren, Algorithmen und Methoden

Ein Verfahren ist nach ISO 8402 eine festgelegte Art und Weise, eine Tätigkeit auszuführen. Es beschreibt eine Strategie zur Lösung einer allgemeinen Problem- bzw. Fragestellung. In ähnlicher Weise definiert Chroust ein Verfahren in der Informatik als einen konkreten Weg zur Lösung bestimmter Probleme oder Problemklassen [28]. Die Beschreibung von Verfahren kann in freiem Text, in Form von Pseudocode, graphisch oder als Programmcode erfolgen.

Eine spezielle Form eines Verfahrens ist ein Algorithmus. Nach [106] ist ein Algorithmus ein mechanisch ausführbares Verfahren, mit dem man für jede Instanz einer Problemklasse eine Antwort auf eine allgemeine Frage erhält. Algorithmen sind Verfahren, die spezifische Eigenschaften aufweisen. So gelten für einen Algorithmus die Eigenschaften der:

- Finitheit: Die Algorithmus ist in einem endlichen Text eindeutig beschreibbar und mit endlichen Ressourcen ausführbar.
- Ausführbarkeit: Jede Instruktion des Algorithmus muss tatsächlich ausführbar sein.
- Terminierung: Der Algorithmus muss für jede Art der Eingabe ein Ergebnis liefern.
- Determiniertheit: Der Algorithmus muss unter denselben Eingabeparametern dasselbe Ergebnis liefern.
- Determinismus: Die nächste anzuwendende Instruktion ist zu jedem Ausführungszeitpunkt definiert.

Typische Beispiele für Algorithmen, die unter anderem auch in der Softwareentwicklung Anwendung finden, sind Suchverfahren (z.B. Lineare Suche, Binäre Suche, Hashing), Schedulingverfahren (z.B. Round Robin), Verarbeitungsverfahren (z.B. LIFO, FIFO) oder Sortierverfahren (z.B. Quicksort, Heapsort).

Ein Verfahren legt neben den durchzuführenden Prozessschritten auch die Datentypen der zu erarbeitenden Ergebnisse fest. Wir unterscheiden im Folgenden zwischen allgemeinen und spezifischen Verfahren. Bei allgemeinen Verfahren liegt der Fokus auf dem Prozess der Problemlösung, nicht auf dem Anwendungskontext. So verwenden diese Verfahren Datentypen einer allgemeinen Anwendungsdomäne, wie beispielsweise der Menge der natürlichen Zahlen, oder sie sprechen allgemein von *Objekten* ohne sich auf einen spezifischen *Objekttyp* festzulegen. Spezifische Verfahren sind dagegen in ihren Datentypen auf eine konkrete Anwendungsdomäne hin festgelegt.

Verfahren bilden den Kern von Methoden. Sie beschreiben innerhalb einer Methode den Prozess zur Lösung der gegebenen Problemstellung. Dabei können drei Formen der Umsetzung des Verfahrens in einer Methode identifiziert werden:

1. Die Methode verwendet ein allgemeines Verfahren und adaptiert es auf ihre Anwendungsdomäne. Die Prozessschritte im Verfahren werden auf die Problemstellung der Methode hin konkretisiert, die allgemeinen Datentypen im Verfahren werden durch Informationstypen der Domänentheorie zur Methode ersetzt (vgl. Abschnitt 4.4). Bei allgemeinen Verfahren handelt es sich in der Regel um Algorithmen der Informatik oder der Mathematik. Die Methoden, die entsprechende Verfahren verwenden, sind automatisiert ausführbar und in der Regel vollständig in Werkzeuge integriert. Beispiele sind Methoden zur Suche und Sortierung von Änderungsanträgen in Änderungsmanagementwerkzeugen, Refaktoringmethoden in Entwicklungsumgebungen oder Methoden zur Prüfung der Zyklenfreiheit von Projektplänen in Projektmanagementwerkzeugen.

2. Die Methode definiert selbst ein neues Verfahren oder kombiniert verschiedene allgemeine Verfahren zu einem neuen komplexen Verfahren. In diesem Fall ist das Verfahren bereits auf die Domänentheorie und Problemstellung der Methode angepasst. Beispiele sind unter anderem OOA und OOD als Methoden zu objektorientierter Analyse und Design ([29], [30]), die Netzplantechnik mit den Verfahren zur Vorwärts- und Rückwärtsrechnung sowie zur Ermittlung der Pufferzeiten, oder die Methode CoCoMo II mit dem Verfahren zur Berechnung des Projektaufwands in Personenmonaten (vgl. Abschnitt A.2).

3. Die Methode nutzt ein Verfahren, das von einer anderen Methode definiert wurde: Hat sich ein Verfahren bewährt, wird es häufig von anderen Methoden weiter genutzt. Das Verfahren wird unverändert übernommen. Gegebenenfalls werden Änderungen an der Prozessbeschreibung oder an den Ergebnistypen vorgenommen, die Grundidee des Verfahrens bleibt jedoch erhalten. Beispielsweise basieren viele UML Entwurfsmethoden auf OOA und OOD. Die Planungsmethoden CPM, PERT und MPM (vgl. Abschnitt A.3) verwenden dagegen die von der Netzplantechnik vorgegebenen Berechnungsverfahren.

Neben der Art der Umsetzung des verwendeten Verfahrens können Methoden auch hinsichtlich der Vollständigkeit unterschieden werden, mit der sie ihr Verfahren umsetzen. So kann eine Methode ein Verfahren vollständig umsetzen, d.h. jeder Prozessschritt im Verfahren ist innerhalb der Methode vollständig und eindeutig beschrieben und kann entsprechend ausgeführt werden. Ziel ist im Allgemeinen die automatisierte Ausführbarkeit des Verfahrens im Rahmen der Methodendurchführung.

Eine Methode kann jedoch ein Verfahren auch nur teilweise umsetzen. Zentrale Prozessschritte werden beschrieben, die Details werden offen gelassen. Gründe für eine teilweise Beschreibung sind beispielsweise:

- Vermeidung von Trivialitäten: Insbesondere bei manuell auszuführenden Prozessen reicht häufig eine grobe Prozessbeschreibung. Der Anwender kann die fehlenden Informationen einfach selbst ergänzen.

- Freiraum für kontextspezifische Anpassungen: Das Verfahren einer Methode sollte bei Bedarf kreativ auf einen spezifischen Kontext, beispielsweise spezielle Rahmenbedingungen in einem Projekt oder Änderungen der Projektsituation anpassbar sein. Gezielte Lücken im Verfahren der Methoden geben hier den entsprechenden Freiraum hinsichtlich der Anpassung.

Im Allgemeinen kann davon ausgegangen werden, dass das Verfahren in automatisiert ausführbaren Methoden vollständig, in manuell oder teilmanuell ausführbaren Methoden nur teilweise beschrieben ist. Eine Methode kann dabei je nach Komplexität und Problemstellung eine Menge verschiedener Verfahren kombinieren. Diese können sowohl vollständig als auch teilweise beschrieben sein.

4.4 Ergänzungen zum Verfahren

Eine Methode kann, insbesondere wenn ihr Verfahren nur teilweise beschrieben ist, um Heuristiken und Prinzipien ergänzt werden. Heuristiken machen Vorschläge für die Umsetzung einzelner Prozessschritte im Verfahren, Prinzipien definieren dagegen Rahmenbedingungen für das Verfahren in seiner Gesamtheit.

Heuristiken

Heuristiken sind aus der Erfahrung abgeleitete Vorgehensweisen zur Lösung spezifischer Probleme. Im Gegensatz zu Methoden, die das Ziel haben, weitgehend durchgängige Lösungswege für komplexe anwendungsbezogene Problemstellungen zu beschreiben, beschränken sich Heuristiken auf die Beschreibung von Lösungsstrategien für spezifische Ausschnitte bzw. Aspekte eines Problems. Heuristiken sind selbst keine Methoden, sie können jedoch innerhalb einer Methoden punktuell zur Lösung eines spezifischen Teilproblems herangezogen werden. Insbesondere können einzelne Verfahrensschritte einer Methode um Heuristiken ergänzt werden, die Vorschläge für deren Durchführung machen. Die Heuristiken können so die Abarbeitung des Verfahrensschritts positiv beeinflussen, sie sind jedoch nicht als obligatorische Vorgabe zu sehen. Ziel für den Einsatz von Heuristiken in Methoden ist eine verbesserte Adaption der Methode an kontextspezifische Eigenheiten in Projekten und damit eine Flexibilisierung der Methodendurchführung.

Methoden, die punktuell um Heuristiken ergänzt wurden, werden auch als heuristische Methoden bezeichnet. Als heuristische Methoden können beispielsweise die von Virenscannern implementierten Methoden zur Virensuche eingeordnet werden. Durch den Vergleich von Methodensignaturen und die Ableitung von Schlussfolgerungen können mit Hilfe von Heuristiken Viren anhand von Ähnlichkeiten identifiziert werden. Heuristische Virenerkennung garantiert nicht, dass alle Viren gefunden werden. Je nach Güte der Heuristiken besteht jedoch eine gewisse Wahrscheinlichkeit, dass Viren identifiziert werden können. Da die Signaturen der Viren nicht bekannt sind, ist der Einsatz einer nicht heuristischen Suchmethode in diesem Fall nicht zielführend.

Sehr viel häufiger werden Heuristiken einer Methode als rein informelle Erfahrungswerte mitgegeben. Beispielsweise nennt Rechtin [97] in seinem Buch 'The Art of Systems Architecting' eine Vielzahl an Heuristiken für den Architekturentwurf, die den Architekten bei seinen verschiedenen Aufgaben hinsichtlich Systementwurf unterstützen, etwa beim Entwurf des Komponentenschnitts. Diese können von Softwarearchitekten bei Bedarf an den entsprechenden Stellen im Entwurfsprozess eingesetzt werden. Auch viele Schätzmethoden stützen sich auf heuristische Erfahrungswerte, die beispielsweise anhand von Kennzahlen zu vergangenen Projekten ermittelt wurden.

Prinzipien

Prinzipien sind nach [9] allgemein gültige Grundsätze bzw. Regeln, die als Leitfaden dienen. Prinzipien werden aus der Erfahrung hergeleitet und durch sie bestätigt oder widerlegt. Prinzipien können als Ergänzung des Verfahrens einer Methode um Regeln verwendet werden. Diese geben dem Anwender einer Methode einen über die einzelnen Prozessschritte hinausgehenden festen Rahmen für die Methodendurchführung vor. Wo das Verfahren der Methode wage und unspezifisch bleibt, definieren Prinzipien zusätzliche allgemeine Rahmenbedingungen für die Methodendurchführung. Bleibt eine Methode in ihren Beschreibungen allgemein und abstrakt, können Prinzipien dem Anwender der Methode bei Unklarheiten helfen diese geeignet zu lösen. Typische Beispiele für Entwurfsprinzipien sind das 'bottom-up' Prinzip oder das 'top-down' Prinzip. Diese Prinzipien definieren Regeln für das prinzipielle Vorgehen zum Systementwurf: Ein Entwurf nach dem 'top-down' Prinzip beginnt im allgemeinen/abstrakten - bei den Anforderungen - und geht zum konkreten - zum Code. Ein Entwurf nach dem 'bottom-up' Prinzip beginnt dagegen im Konkreten - beispielsweise mit der Identifizierung bereits fertig gestellter Komponenten - und geht hin zum Allgemeinen/Abstrakten - zur Systemarchitektur. Häufig verwendet wird auch das Prinzip 'teile und herrsche', das eine Problemlösung durch Zerlegung eines Problems in Teilprobleme und Lösung der Teilprobleme fordert.

Die Zuordnung von Prinzipien zu Methoden, die ein durchgängiges Verfahren definieren, kann auch zur Charakterisierung der Methode dienen. So wird im Sprachgebrauch beispielsweise von einer 'top-down' Methode oder einer 'bottom-up' Methode gesprochen, wenn deutlich gemacht werden soll, dass eine Methode nach dem entsprechenden Prinzip vorgeht.

4.5 Domänentheorien und Notationen

Zur Formulierung von Prozessschritten und Ergebnistypen ihrer Verfahren benötigt eine Methode ein auf ihre Anwendungsdomäne abgestimmtes Vokabular. Das Vokabular liefert ihr eine auf ihre Anwendungsdomäne abgestimmte Domänentheorie. Zur Darstellung der erarbeiteten Ergebnisse gibt die Methode domänenspezifische Sprachen und Notationen vor.

Domänentheorien

Eine Domänentheorie ist eine Theorie über eine Anwendungsdomäne (vgl. Theorien in Abschnitt 4.6). Die Domänentheorie definiert die relevanten Konzepte der Anwendungsdomäne mit ihren Beziehungen und Axiomen sowie Konstruktionsregeln über den Konzepten auf Basis der Axiome. Methoden nutzen eine Domänentheorie als Sprache zur Beschreibung ihrer Verfahrensschritte und ihrer Ergebnisse. Die Artefakte der Methode repräsentieren Sichten auf Teilmengen der Informationstypen der Domänentheorie. Die Arbeitsschritte beschreiben gültige Operationen auf den Informationstypen der Artefakte und stützen sich in ihren Beschreibungen auf Axiome und Ableitungsregeln der Domänentheorie.

Domänentheorien werden in der Regel als Teil einer Methode definiert und können bei Bedarf von anderen Methoden mitverwendet werden. Beispielsweise definiert die Methode der Netzplantechnik eine Theorie über der Anwendungsdomäne der Projektplanung. Die Theorie definiert die für die Projektplanung relevanten Typen: Vorgang und Anordnungsbeziehung. Auf Basis dieser Typen beschreibt die Methode das Verfahren zur Erstellung eines Projektplans und definiert als zentralen Ergebnistyp den Netzplan selbst. Der Netzplan entspricht im Wesentlichen einer Vereinigung von Vorgängen und Anordnungsbeziehungen unter Beachtung gültiger Verknüpfungsregeln. Bei Änderungen im Netzplan müssen die Ableitungsregeln der Theorie berücksichtigt werden.

Viele Methoden definieren rein informelle Beschreibungen der Domänentheorien. Die Methoden verwenden Begriffe ihrer Domäne und definieren Konstruktions- und Ableitungsregeln, die aus der Erfahrung heraus als richtig erkannt werden. Dies trifft beispielsweise für die Schätzmethoden CoCoMo II und FPA ebenso zu wie für die Netzplantechnik oder die objektorientierten Entwurfsmethoden OOA und OOD [29], [30]. Daneben gibt es jedoch auch Methoden mit formal spezifizierten Domänentheorien, wie beispielsweise die formalen Entwicklungsmethoden Focus [21] oder die Vienna Development Method [21]. Diese kommen in der Regel aus dem Forschungsbereich und wurden auf eine Anwendung in der Praxis adaptiert. Methoden mit formaler Domänentheorie können im Gegensatz zu Methoden, die aus der Erfahrung heraus entstanden sind, auch als wissenschaftlich fundiert eingeordnet werden.

Notationen und Domänenspezifische Sprachen

Eine Notation entspricht einer Menge von Wörtern über einer Menge von Symbolen. Die syntaktische Struktur der Wörter wird über eine Grammatik festgelegt. Es gibt eine große Zahl unterschiedlicher Notationen. Das Spektrum reicht von formalen Spezifikationssprachen (z.B. Prädikatenlogische Ausdrücke) über graphische Notationen (z.B. UML, Diagramme, E/R Diagramme) bis hin zu freiem Text, gegebenenfalls in Tabellen oder Listen strukturiert.

Wie gezeigt wurde, definieren Domänentheorien die relevanten Informationstypen einer Anwendungsdomäne sowie die Beziehungsstruktur zwischen den Informationstypen. Methoden stützen

sich in der Beschreibung ihres Verfahrens auf die von der Domänentheorie vorgegebenen Informationstypen. Zur Darstellung der im Rahmen der Methodendurchführung erarbeiteten Instanzen zu den Informationstypen, gibt eine Methode Notationen vor. Die Symbole der Notation legen für atomare Informationstypen der Domänentheorie eine visuelle Repräsentation fest, die Grammatik der Notation definiert Regeln für die Bildung gültiger Verknüpfungen über den Informationstypen. Die Methodenergebnisse werden so mit den sprachlichen Mitteln der Notation auf Basis der Domänentheorie dargestellt.

Für eine verbesserte Unterstützung der Methodendurchführung, insbesondere bei einer werkzeugunterstützten Erstellung der Methodenergebnisse, kann eine Methode dem Anwender eine geeignete, auf das zu lösende Problem abgestimmte, domänenspezifische Sprache zur Verfügung stellen. Eine domänenspezifische Sprache (domain specific language) ist nach [35]:

> ... a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain.

Damit eine domänenspezifische Sprache geeignet von einer Methode angewendet werden kann, muss sie eine entsprechende Abstraktion der Anwendungsdomäne zur Methode definieren. Konkret muss die abstrakte Syntax der domänenspezifischen Sprache das axiomatische System der Domänentheorie (vgl. deduktive Theorien in Abschnitt 4.6) der Methode abbilden.

Domänenspezifische Sprachen sind von ihrer Definition her ausführbar. Damit eine Sprache ausführbar ist, benötigt sie eine formale Semantik. Liegt keine formale Domänentheorie vor, muss die Sprache die Theorie formalisieren. Zu vielen der bekannten domänenspezifischen Sprachen gibt es jedoch keine explizit definierte formale Semantik, vielmehr wird die Semantik implizit über ihre Implementierung bzw. den jeweiligen Sprachinterpreter ihrer Implementierung definiert. Das hat zur Folge, dass die Semantik vieler domänenspezifischer Sprachen abhängig ist von ihrer jeweiligen Implementierung und sich in Details leicht unterscheiden kann. Typische Beispiele für domänenspezifische Sprachen mit ausschließlich implementierter formaler Semantik sind die verschiedenen von der OMG definierten UML Profile. Ein UML Profil ist die Spezifikation einer Anpassung der UML auf eine spezifische Anwendungsdomäne. Die OMG bietet beispielsweise Spezifikationen von UML Profilen für CORBA, für Systems Engineering (SysML) oder für Enterprise Application Integration. Alle zur Zeit verfügbaren Profile stehen unter [134] zur Verfügung. Weitere Beispiele für domänenspezifische Sprachen mit ausschließlich implementierter Semantik sind die von der Netzplantechnik vorgeschlagenen Netzplanarten: Ereignis-Knoten-Netzplan, Vorgangs-Pfeil-Netzplan und Vorgangs-Knoten-Netzplan (vgl. Abschnitt A.3).

Für einige domänenspezifische Sprachen gibt es dagegen eine explizit definierte formale Semantik. Dazu zählen beispielsweise Entity Relationship Diagramme zur Modellierung von Datenmodellen. Domänenmodell der Entity Relationship Diagramme ist das bereits 1977 von Chen definierte En-

tity Relationship Modell [27], das auf der formalen Mengentheorie basiert. Eine weitere formale domänenspezifische Sprache ist die Modellierungssprache Autofocus, die als graphische Notation für die formale Domänentheorie der Entwurfsmethode Focus [21] entwickelt wurde.

4.6 Einflussfaktoren

Methoden unterscheiden sich zum Teil gravierend hinsichtlich der von ihnen genutzten Domänentheorie und des von ihnen implementierten Verfahrens. Dies ist auch der Fall bei Methoden, die ähnliche oder sogar die gleiche Fragestellung fokussieren. Zu sehen ist dies insbesondere bei den vielen zum Teil sehr unterschiedlichen Entwurfsmethoden, die im Laufe der Jahre entwickelt wurden. Diese Unterschiede erklären sich insbesondere durch Theorien und Paradigmen, die zum Zeitpunkt der Methodenentwicklung richtungsweisend (modern) waren.

Theorien

Eine Theorie kann allgemein als ein System von Aussagen und Sätzen bezeichnet werden, das zur Beschreibung, Erklärung und Vorhersage von Phänomenen dient. Die Wissenschaftstheorie unterscheidet zwischen empirischen und deduktiven Theorien. Empirische Theorien werden auf der Basis von Beobachtungen und bestätigten Hypothesen gebildet. Zum Nachweis der Richtigkeit der Hypothesen werden Prognosen erstellt. Diese können anhand von Experimenten verifiziert oder falsifiziert werden.

Im Gegensatz zu den empirischen Theorien stützen sich deduktive Theorien nicht auf Beobachtungen sondern entstehen durch ein modellgetriebenes Denken. Kennzeichen einer deduktiven Theorie ist die Formulierung der Theorie in Form eines axiomatischen Systems. Ein axiomatisches System ist ein System von primitiven Axiomen als Prämissen sowie Regeln zur Ableitung von Aussagen über den Axiomen. Ein formales axiomatisches System besteht aus folgenden Elementen vgl. [75]:

- Einem Alphabet Σ von Symbolen zur Erzeugung von Zeichenketten und Aussagen,
- der Menge WF der wohlgeformter Ausdrücke über dem Alphabet mit $WF \subseteq \Sigma^*$ und Σ^* als der Menge aller Zeichenketten über Σ,
- der Menge Ax von Axiomen als Teilmenge von WF, sowie
- einer Menge R von Ableitungsregeln zur Bildung von Aussagen über den Axiomen.

Ein formales axiomatisches System muss zwei zentrale Eigenschaften erfüllen: das System muss konsistent sein - zwei Aussagen, die im System abgeleitet werden können, dürfen sich nicht wider-

sprechen - und das System muss vollständig sein - alle Aussagen, die innerhalb des Systems wahr sind, müssen aus den Axiomen ableitbar sein [1].

Ein axiomatisches System hat selbst keinerlei Wahrheitsanspruch. Es handelt sich um ein rein logisches System von Ausdrücken. Erst durch Interpretation in einer semantischen Domäne, beispielsweise der Mathematik, erhält es eine Bedeutung und wird wahrheitsfähig. Allgemeine Theorien die vor allem in der Informatik eine Rolle spielen, sind beispielsweise die Berechenbarkeitstheorie, die Automatentheorie, die Mengentheorie, die Komplexitätstheorie oder die Graphentheorie.

Theorien können die von einer Methode genutzte Domänentheorie unmittelbar beeinflussen. Eine Domänentheorie entspricht einer deduktiven Theorie mit einer Interpretation in einer konkreten Anwendungsdomäne. Das axiomatische System kann jedoch auch auf eine allgemeine Theorie abgebildet werden und so Erkenntnisse der Theorien für Methoden konkret nutzbar machen. Beispielsweise stützen sich die Domänentheorien vieler Entwurfsmethoden auf die Automatentheorie oder die Mengentheorie, die Domänentheorie der Netzplantechnik stützt sich dagegen auf Ergebnisse der allgemeinen Graphentheorie.

Paradigmen

Thomas Kuhn definiert in seinem Essay 'Die Struktur wissenschaftlicher Revolutionen' [78] ein Paradigma als eine zu einem Zeitpunkt vorherrschende wissenschaftliche Theorie, die das Denken der wissenschaftlichen Forscher beeinflusst. Ein Paradigma ist eine Menge von Erkenntnissen und Sichtweisen, aus denen sich grundlegende Denk- und Handlungsmuster der Wissenschaftler ableiten lassen. Das Paradigma gibt dem Wissenschaftler vor, was ein Problem ist und wie es mit den durch das Paradigma gegebenen Mitteln zu lösen ist. Entscheidend für den Paradigmenbegriff ist die Subjektivität: Paradigmen repräsentieren eine aktuell vorherrschende Denkrichtung und können nach Kuhn die wissenschaftliche Meinung einer Zeit stark beeinflussen. Kuhn führte neben dem Paradigma den Begriff des Paradigmenwechsels ein. Paradigmen bilden für eine gewisse Zeit die theoretische Grundlage für wissenschaftliches Forschen und Handeln. Im Laufe der Zeit treten Widersprüche auf, Probleme können mit den gegebenen Mitteln nicht mehr gelöst werden. Als Folge werden neue Denkmuster und Lösungsstrategien entwickelt - ein Paradigmenwechsel findet statt.

Paradigmen sind Konzepte, die wie Theorien nur mittelbar Einfluss auf Methoden ausüben. Sie geben jedoch eine Begründung bzw. eine Rechtfertigung für die Wahl einer spezifischen Domänentheorie für eine Methode. In der Informatik gab und gibt es seit den Anfängen der Softwareentwicklung in den 60er Jahren immer wieder vorherrschende Paradigmen gefolgt von Paradigmenwechseln. Diese Entwicklung lässt sich sehr gut bei den verschiedenen Entwicklungsmethoden beobachten. So

[1] In seinen Unvollständigkeitssätzen wies Kurt Gödel (1906 - 1978) 1931 allerdings nach, dass ein formales System, das umfangreich genug ist, um damit die Arithmetik der natürlichen Zahlen beschreiben zu können, entweder unvollständig oder inkonsistent ist (vgl. [51], [116]).

kam als Ergebnis der schlechten Qualität von Softwareprogrammen bereits Ende der 60er Jahre, unter anderem ausgelöst durch den legendären Aufsatz von Dijkstra 'Goto Statement considered harmful' [36] das Paradigma der strukturierten Programmierung als das führende Paradigma der Systementwicklung auf. Mit der strukturierten Programmierung fanden erstmals Abstraktionen und Strukturierungselemente Einzug in die Programmiersprachen. Führende Vertreter der strukturierten Programmierung waren Hoare, Dahl und Dijkstra, die 1972 ihre Arbeiten zusammenfassten [32].

Prinzipien und Grundsätze der strukturierten Programmierung wurden später auch auf die Analyse- und Designphase übertragen. Ziel war eine erste Formalisierung der Systembeschreibung sowie des Systemdesigns. Als Vorreiter der strukturierten Analyse und des strukturierten Designs ist insbesondere Tom de Marco 1978 mit 'Structured Analysis and System Specification' [33] zu nennen. Daneben gibt es eine Reihe weiterer Vertreter dieses Paradigmas (vgl. [115], [114], [48], [80]). Das Paradigma der strukturierten Analyse und Design war als eines der ersten Paradigmen, das mehrere Phasen im Systementwicklungsprozess betrachtete, ausgesprochen erfolgreich, wurde jedoch in den 90er Jahren weitgehend durch das Paradigma der Objektorientierung abgelöst.

Die Ursprünge der Objektorientierung sind in den Konzepten der Programmiersprache SIMULA zu finden (siehe auch [46]). Nicht zuletzt deshalb wurde das objektorientierte Paradigma anfangs hauptsächlich in Programmiersprachen wie beispielsweise in C++ oder in der von Bertrand Meyer entwickelten Programmiersprache Eiffel [82] genutzt. Mit der Zeit fand die Idee der Objektorientierung jedoch auch ihren Einzug in die Entwurfsmethoden. Zu nennen sind hier vor allem die Methoden von James Rumbaugh, Grady Booch und Ivar Jacobson ([101], [20], [68]), die als Begründer der UML später auch unter dem Namen 'die drei Amigos' bekannt wurden. Der Siegeszug der Objektorientierung fand schließlich seinen Höhepunkt in der Zusammenführung der verschiedenen Methoden und der Standardisierung der Unified Modeling Language (UML) [86] als einheitlicher objektorientierter Notation. Mit Einführung der UML fand der Wildwuchs bei den objektorientierten Methoden, zumindest hinsichtlich der verwendeten Notation, ein Ende. Heute ist die Objektorientierung als Paradigma zwar noch in Verwendung, es wurde jedoch erkannt, dass damit viele Probleme der Softwareentwicklung nicht befriedigend gelöst werden können. In den letzten Jahren entstanden daher weiterer Paradigmen als Ergänzung wie die komponentenbasierte Systementwicklung ([6], [40]) oder die modellbasierte Entwicklung ([125]).

4.7 Methodenontologie und Klassifikation

Abschließend werden in diesem Abschnitt die wesentlichen Ergebnisse dieses Kapitels in Form einer Methodenontologie (Abbildung 5) zusammengefasst. Die Ontologie stellt alle der genannten Konzepte mit ihren Abhängigkeiten graphisch dar. Als Notation wird die in Abschnitt 2.3 eingeführte Modellierungssprache verwendet. Ziel der Ontologie ist eine Visualisierung der in Abschnitt 4.2 eingeführten Methodendefinition und damit eine erste pseudo-formale Darstellung des Konzepts einer

Methode. Die Ontologie dient insbesondere als Grundlage für das formale Methoden-Metamodell in Kapitel 5.

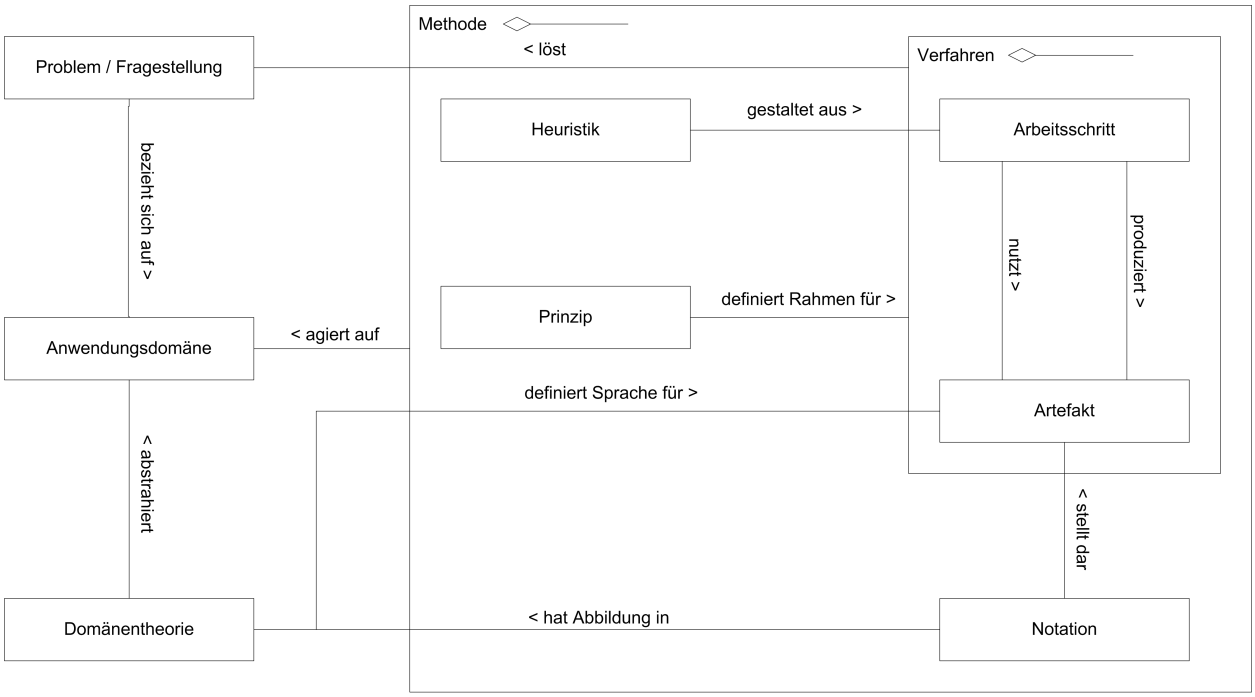

Abbildung 4.1: Eine informelle Methodenontologie

Im Folgenden werden die wesentlichen Ergebnisse noch einmal zusammengefasst. Ziel einer Methode ist die Beschreibung eines Prozesses zur Lösung eines Problems innerhalb einer spezifischen Anwendungsdomäne. Zentrales Konzept einer Methode ist somit das Verfahren zur Problemlösung. Das Verfahren definiert über Arbeitsschritte und Artefakte den zur Problemlösung durchzuführende Prozess. Arbeitsschritte beschreiben einzelne Prozessschritte im Verfahren, Artefakte beschreiben die zu erarbeitenden Ergebnisse unter Berücksichtigung der gegebenen Anwendungsdomäne. Ein Artefakt entspricht einer Sicht auf eine Menge von Informationstypen der Domänentheorie zur Methode. Die Durchführung eines Arbeitsschritts führt zur Erstellung von Instanzen zu den Informationstypen des Artefakts. Dazu kann ein Arbeitsschritt die von anderen Artefakten bereitgestellten Informationen nutzen. Notationen dienen zur Dokumentation der Artefakte und ihrer Informationstypen. Damit eine Notation geeignet für ein Artefakt verwendet werden kann, bedarf es einer Abbildung der abstrakten Syntax der Notation in die Domänentheorie. Domänenspezifische Sprachen sind eine spezielle Form von Notationen, die bereits eine Domänentheorie mitbringen.

Arbeitsschritte beschreiben Prozessschritte im Verfahren. Die Beschreibung muss nicht immer vollständig sein. Methoden können statt dessen mit Hilfe von Heuristiken Vorschläge für die Durchführung einzelner Arbeitsschritten machen. Heuristiken sind jedoch im Gegensatz zu vollständig ausformulierten Arbeitsschritten nicht als verbindlich zu betrachten. Insbesondere geben Heuristiken keine Garantie hinsichtlich des Erfolgs bei ihrer Anwendung. Heuristiken werden

eingesetzt, um Methoden geeignet um Erfahrungswerte zu ergänzen. Sie beziehen sich dabei auf einzelne Arbeitsschritte im Verfahren. Richtlinien für das gesamte Verfahren werden dagegen durch Prinzipien gegeben. Ein Verfahren kann die Prinzipien bereits vollständig umsetzen, in diesem Fall entspricht die Zuordnung des Prinzips einer Charakterisierung der Methode. Eine Methode kann jedoch bei nicht vollständigen Verfahren durch Angabe von Prinzipien dem Anwender der Methode einen Rahmen für die Durchführung des Verfahren vorgeben.

Die Ontologie impliziert, dass alle Methoden von Inhalten und Aufbau her ähnlich aufgebaut sind. In der Praxis weisen Methoden jedoch zum Teil gravierende Unterschiede hinsichtlich ihrer Schwerpunkte bei der Ausarbeitung auf. So können Methoden danach unterschieden werden, welcher ihrer Bereiche (Domänentheorie, Verfahren) besonders gut ausgearbeitet ist. Dementsprechend unterscheiden wir *verfahrensnahe*, *theorienahe* und *anwendungsnahe Methoden*. Abbildung 4.2 stellt die drei Methodenklassen und ihre Beziehung graphisch dar.

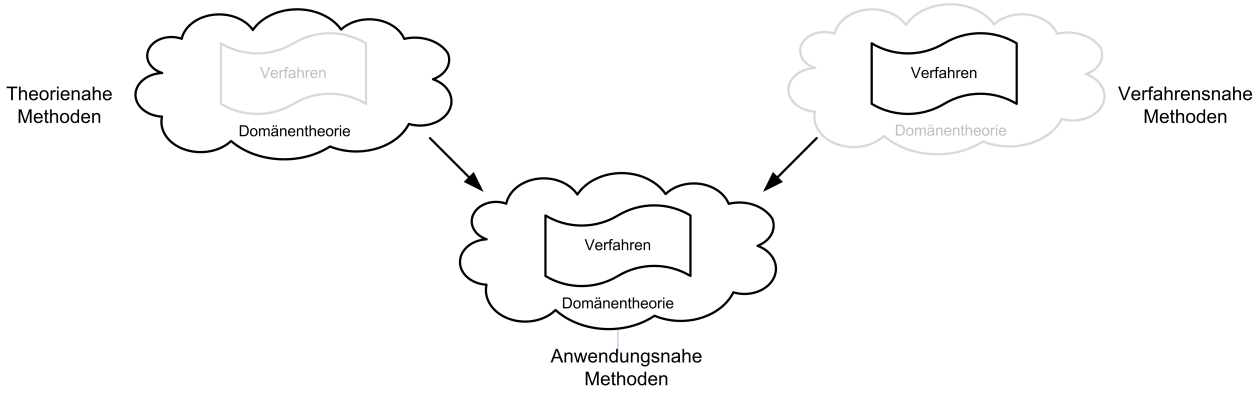

Abbildung 4.2: Methodenklassen und Beziehungen

Theorienahe Methoden konzentrieren sich auf die Definition einer Domänentheorie, häufig mit Angabe von Notationen. Ein rudimentäres, häufig nicht durchgängiges Verfahren wird ebenfalls vorgegeben. Typischer Vertreter dieser Methodenklasse sind die Netzplantechnik mit ihrer Domänentheorie zur Projektplanung und die Entwurfsmethode Focus mit der Domänentheorie zum komponentenbasierten Systementwurf. Eine spezielle Form der theorienahen Methoden sind Methoden, die in Werkzeuge integriert sind. Die Werkzeug implementieren eine entsprechende Domänentheorie, beispielsweise zum Architekturentwurf, zum Änderungsmanagement oder zum Konfigurationsmanagement, und geben zusätzlich ein Verfahren zur Anwendung der Domänentheorie vor, beispielsweise zum Management von Änderungsanträgen.

Als *verfahrensnahe Methoden* bezeichnen wir Methoden, die besonderen Wert auf die Beschreibung eines Verfahrens zur Lösung einer spezifischen Problemstellung legen. Die Domänentheorie mit Notation spielt dagegen eine untergeordnete Rolle. Das Verfahren ist allgemein gehalten und dient bewusst als Grundlage weiterer Methoden. Als verfahrensnahe Methoden können neben allgemeinen Algorithmen insbesondere Testmethoden (z.B. Äquivalenzklassen Test, Black-box Test, White-box Test), Refaktoringmethoden, Review-, Interview- oder Moderationsmethoden eingeordnet werden.

Anwendungsnahe Methoden definieren sowohl eine Domänentheorie wie auch ein Verfahren, das auf die Begriffswelt der Domänentheorie abgestimmt ist. Methoden dieser Klasse fokussieren die Anwendung der Methode in der Praxis. Typische Beispiele sind viele der detailliert beschriebenen UML Entwicklungsmethode wie KobrA oder Catalysis. Jede dieser Methoden nutzt ein Subset der UML als Domänentheorie und beschreibt ein für die Methode spezifisches Verfahren zum Systemdesign, mit unterschiedlichen Prinzipien und Heuristiken. Als weiteres Beispiel kann CoCoMo II genannt werden. Die Methode definiert einerseits eine Domänentheorie zur Schätzung, andererseits wird das Verfahren zur Berechnung des Aufwands angegeben. So vereinigt die Methode Domänentheorie und Verfahren in geeigneter Weise.

Die Grenze zwischen den Klassen ist nicht immer eindeutig zu erkennen. So ist es in Fällen wie beispielsweise der Netzplantechnik Ermessenssache, ob das Verfahren bereits so vollständig ist, das die Methode als anwendungsnah eingeordnet werden kann oder ob es sich noch um eine theorienahe Methode handelt. Als Richtlinie für eine Zuordnung von Methoden kann hier die Vollständigkeit der Beschreibung und der Grad der detaillierten Ausarbeitung dienen.

Abschließend wird noch die Frage der Anpassung für Methoden dieser drei Methodenklassen diskutiert. Methoden werden entwickelt um in der Praxis in einem konkreten Umfeld, in der Regel einer Organisation, zur Lösung eines Problems eingesetzt zu werden. Wie bei Vorgehensmodellen ist somit auch für Methoden eine Form von Anpassung erforderlich, um sie sinnvoll nutzen zu können. Theorienahe und verfahrensnahe Methoden sind speziell für diese Form der Anwendung gedacht. Theorienahe Methoden können einfach organisationsspezifische Verfahren ergänzt werden, verfahrensnahe Methoden um eine für die Organisation geeignete Domänentheorie. Schwieriger ist eine Anpassung jedoch bei anwendungsnahen Methoden. Durch ihre detaillierte Beschreibung von Domänentheorie und Verfahren werden Methoden dieser Klasse ähnlich zu methodenspezifischen Vorgehensmodellen schnell unflexibel hinsichtlich einer Anpassung. Dieser Aspekt sollte unbedingt berücksichtigt werden, wenn Methoden in Organisationen für spezifische Zwecke gewählt werden.

4.8 Vorgehensmodelle und Methoden

Ein kritischer Aspekt für den in dieser Arbeit betrachteten Kontext der Methodenintegration ist die Abgrenzung von Methoden zu Vorgehensmodellen. Vorgehensmodelle erfüllen im Wesentlichen alle Eigenschaften einer Methode der Systementwicklung (vgl. Abschnitt 3.1): Ein Vorgehensmodell liefert eine Lösung für das Problem der strukturierten und geordneten Projektdurchführung. Es definiert über Aktivitäts- und Produktmodell eine Prozessbeschreibung zur Durchführung von Entwicklungsprojekten in gegebener Zeit, unter Berücksichtigung des verfügbaren Budgets und bei geforderter Qualität und Leistung. Dazu agiert ein Vorgehensmodell auf der Anwendungsdomäne der Projektdurchführung und benötigt eine entsprechende Domänentheorie. Prinzipien, wie beispielsweise das Prinzip der Agilität beeinflussen maßgeblich die Prozessbeschreibung. Vorgehensmodelle werden wie Methoden durch Paradigmen beeinflusst, beispielsweise das Paradigma der

Objektorientierung und stützen sich in ihrer Vorgehensweise auf heuristische Erfahrungswerte aus der Praxis.

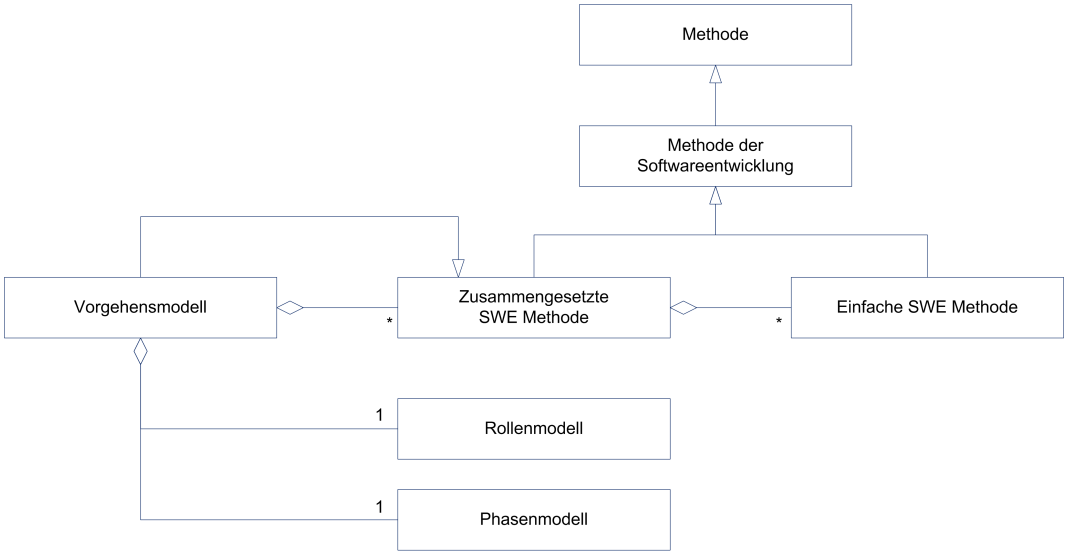

Abbildung 4.3: Vorgehensmodelle und Methoden

Ein Vorgehensmodell bietet jedoch mehr als eine Methode. Abbildung 4.3 zeigt anhand einer einfachen Ontologie den Zusammenhang von Vorgehensmodell und Methoden. Konkret verbindet ein Vorgehensmodell eine Menge von Methoden der Softwareentwicklung mit einem Phasenmodell und einem Rollenmodell. Arbeitsschritte und Artefakte der Methoden entsprechen den Produkten und Aktivitäten im Vorgehensmodell. Das Phasenmodell erlaubt eine grobgranulare zeitliche Organisation der Aktivitäten und Produkte im Vorgehensmodell, das Rollenmodell legt fest, welche Fähigkeiten eine Person mitbringen muss, um eine bestimmte Aktivität im Projekt durchzuführen. Unter Berücksichtigung der hier erarbeiteten Ergebnisse definieren wir nun ein Vorgehensmodell wie folgt:

> **Definition:** Ein Vorgehensmodell ist eine Methode zur Organisation einer Menge von Methoden der Softwareentwicklung mit dem Ziel der Durchführung eines Entwicklungsprojekts.

In Abschnitt 3.3 wurden Vorgehensmodelle nach methodenspezifisch und methodenneutral unterschieden. Im Hinblick auf die hier vorgestellte Definition eines Vorgehensmodells können wir diese Begriffe nun konkretisieren: Wir bezeichnen ein Vorgehensmodell als methodenspezifisch wenn das Verfahren aller seiner Methoden vollständig und durchgängig beschrieben ist. Wir bezeichnen dagegen ein Vorgehensmodell als methodenneutral, wenn das Verfahren seiner Methoden nicht vollständig angegeben ist und Raum für individuelle Ausgestaltungen bleibt.

4.9 Zusammenfassung

In diesem Kapitel wurde die Domäne der Methoden in der Systementwicklung systematisch untersucht. Ausgehend von einem informell definierten Methodenbegriff wurden die zentralen Konzepte, die im Zusammenhang mit Methoden eine Rolle spielen, vorgestellt. Kern einer Methode ist ein Verfahren, das den Prozess zur Problemlösung beschreibt. Als Sprache zur Beschreibung des Verfahrens stützt sich eine Methode auf eine Domänentheorie. Die Darstellung der Ergebnisse erfolgt über eine Notation. Die Notation benötigt, um für eine Methode geeignet zu sein, eine Abbildung in die Domänentheorie der Methode. Welches Verfahren und welche Domänentheorie eine Methode nutzt, wird im Rahmen der Entwicklung einer Methode festgelegt und ist wesentlich durch die zu diesem Zeitpunkt vorherrschenden Paradigmen und Theorien bestimmt. Das in diesem Kapitel gewählte Klassifikationsschema für Methoden stützt sich ebenfalls auf die Konzepte Domänentheorie und Verfahren. Danach können theorienahe, verfahrensnahe und anwendungsnahe Methoden unterschieden werden. Theorienahe Methoden definieren vorrangig eine Domänentheorie, das Verfahren ist dagegen nicht oder nur rudimentär angegeben. Verfahrensnahe Methoden fokussieren dagegen stärker die Angabe des Verfahrens und verwenden in der Regel eine eher unspezifische, allgemeine Domänentheorie. Anwendungsnahe Methoden geben dagegen sowohl Verfahren wie Domänentheorie in ähnlicher Tiefe an. Abschließend wurde anhand der hier erarbeiteten Ergebnisse der Begriff eines Vorgehensmodells neu untersucht und es wurde der Bezug zwischen Vorgehensmodellen und Methoden aufgezeigt.

Kapitel 5

Formalisierung der Methodenintegration

Vorgehensmodelle sind Modelle der Projektdurchführung; Methoden ergänzen Vorgehensmodelle um detaillierte Prozessbeschreibungen zur systematischen Erarbeitung einzelner Ergebnisse in Projekten. In diesem Kapitel wird nun im Detail die Schnittstelle zwischen Vorgehensmodellen und Methoden untersucht. Die Konzepte Vorgehensmodell und Methode werden klar voneinander abgegrenzt und es wird gezeigt, wie sichergestellt werden kann, dass die von einer Methode gelieferten Ergebnisse tatsächlich den vom Vorgehensmodell geforderten Ergebnissen entsprechen. Insbesondere wird die zentrale Rolle der Domänentheorie für eine erfolgreiche Methodenintegration aufgezeigt.

Zur Konkretisierung der Beziehungen zwischen Vorgehensmodell und Methoden werden in diesem Kapitel sowohl Methoden, wie auch die Schnittstelle zwischen Methoden und Vorgehensmodell formalisiert. Die Formalisierung dient einerseits dazu, die strukturelle Einbettung von Methoden in Vorgehensmodellen formal zu erfassen und andererseits, die inhaltlichen Abhängigkeiten und Konsistenzbedingungen an der Schnittstelle auf Basis der Domänentheorie zu formulieren. Des Weiteren dient die Formalisierung als Grundlage für die in Kapitel 6 entwickelten Metamodelle der Methodenbibliothek und der Integration. In dieser Arbeit wird darauf verzichtet, das Vorgehensmodell selbst vollständig zu formalisieren, da dies für die hier betrachtete Problemstellung als nicht relevant angesehen wird.

Das Kapitel ist wie wie folgt aufgebaut: Abschnitt 5.1 stellt eine formales Metamodell für Methoden mit abstrakter Syntax und Semantik vor. Die abstrakte Syntax des Metamodells orientiert sich in ihren Konzepten an der Methodenontologie in Abschnitt 4.6, setzt jedoch auf einer vereinfachenden patternartigen Struktur für Methoden auf, die eine flexiblere Methodenmodellierung erlaubt. Eine informelle Abgrenzung von Vorgehensmodell und Methoden wird in Abschnitt 5.2 erläutert. Die weiteren Abschnitte beschäftigen sich mit der Formalisierung der Schnittstelle zwischen Vorgehensmodell und Methoden. Abschnitt 5.3 beschreibt, wie Vorgehensmodelle ihre Anforderungen an potentiell integrierbare Methoden formulieren. Die Erfüllung der Anforderungen durch Metho-

den wird in Abschnitt 5.4 gezeigt und es werden die Probleme, die bei einer Integration auftreten können, diskutiert. Abschließend wird in Abschnitt 5.5 das formale Methoden-Metamodell um eine formale Beschreibung der Methodenintegration erweitert. Abschnitt 5.6 fasst die Ergebnisse des Kapitels zusammen.

5.1 Ein formales Methoden-Metamodell

In diesem Abschnitt wird ein formales Metamodell zur Spezifikation der abstrakten Syntax und Semantik von Methoden definiert. Zur Beschreibung des Metamodells werden Prädikatenlogik erster Stufe sowie einfache mengentheoretische Konstrukte verwendet. Dieser Abschnitt motiviert anhand einer Beispielmethode die für die Formalisierung gewählte modulare Struktur. Diese dient als Vorbild für die im Anschluss definierte abstrakte Syntax des formalen Methoden-Metamodells.

Die hier gewählte Form der Methodenmodellierung geht von einer hochgradig parallelisierten Methodendurchführung aus. Eine Reihenfolge der durchzuführenden Arbeitsschritte ist ausschließlich durch kausale Abhängigkeiten zwischen Artefakten gegeben. Existieren keine Abhängigkeiten, wird implizite Nebenläufigkeit der Durchführung angenommen. Aus diesem Grund wurde als semantische Domäne des Metamodells der Formalismus der Petrinetze gewählt. Petrinetze sind eine spezielle Form der Automaten, die auf die Modellierung nebenläufiger Prozesse ausgerichtet sind. Der Abschnitt schließt entsprechend mit der Spezifikation der Methodensemantik durch eine Abbildung der abstrakten Syntax auf den Formalismus der Petri-Netze.

Während die hier spezifizierte abstrakte Syntax des formalen Methoden-Metamodells als Vorlage für die Entwicklung des semi-formalen und damit ausdrucksschwächeren, jedoch anwendungsfreundlicheren, Methoden-Metamodells in Kapitel 6 dient, kann die formale Semantik als erster Schritt hin zur Entwicklung einer Plattform zur werkzeugunterstützten Nachverfolgung der Methodendurchführung gesehen werden, vergleichbar zu einer methodenunabhängigen, parametrisierbaren Workflow-Engine. Die Entwicklung einer entsprechenden Plattform ist jedoch nicht im Fokus dieser Arbeit.

5.1.1 Ein einführendes Beispiel

In diesem Abschnitt wird die für die abstrakte Syntax gewählte modulare Struktur motiviert und anhand der Schätzmethode CoCoMo II informell eingeführt. Ziel von CoCoMo II ist die Berechnung des Aufwands für ein Entwicklungsprojekt anhand der Formel:

$$PM = A * K * (Size)^S$$

Erster Schritt der Methode ist die Bestimmung der verschiedenen Variablen und Konstanten: der Schätzgröße *Size*, des Kostenfaktors K, des Skalarfaktors S sowie der Konstante A. Zur Bestimmung der Variablen gibt die Methode eine Reihe von Hilfestellungen in Form von Vorgaben. Vorgaben

sind unter anderem eine Liste von fünf vordefinierten Skalenfaktoren, eine Liste von 17 typischen Kostentreibern sowie die jeweiligen Bewertungstabellen. Die Größe des Schätzproblems wird anhand der Größe Source Lines of Code (SLoC) angegeben. Zur Bestimmung der Werte benötigt die Methode Informationen zum Schätzgegenstand. Wurden alle Variablen und die Konstante ermittelt, kann der Entwicklungsaufwand anhand der gegebenen Formel berechnet werden. Ergebnis der Berechnung ist der geschätzte Aufwand in Personenmonaten. Abbildung 5.1 zeigt einen ersten Entwurf für den strukturellen Aufbau von CoCoMo II. Notation ist wieder die Modellierungssprache aus Kapitel 2

Abbildung 5.1: Aufbau der Methode CoCoMo II

Mit Bezug auf die Methodenontologie in Abschnitt 4.6 identifizieren wir als zentrale Konzepte einer Methode Arbeitsschritte, Artefakte und Notationen. Arbeitsschritte beschreiben einzelne Prozessschritte, Artefakte repräsentieren Eingaben und Ergebnisse von Arbeitsschritten, Notationen dienen zur Dokumentation der Artefakte. So beschreibt der Arbeitsschritt *Bestimme den Skalenfaktor S* das Vorgehen zur Ermittlung des Faktors. Als Ergebnis produziert der Arbeitsschritt ein Artefakt, das den *Skalenfaktor S* repräsentiert. Notationen dienen zur Dokumentation der Artefakte. Im Beispiel werden zwei Notationen verwendet, die natürliche Sprache zur Beschreibung des Schätzgegenstands sowie die Mathematik für die Angabe der Variablen und der Konstante. Auf die explizite Angabe der Notation kann in trivialen Fällen, wie in diesem Beispiel, verzichtet werden. Notwendig ist die Angabe einer Notation dagegen bei komplexen Artefakten oder spezifischen Notationen, wie

beispielsweise die Angabe von UML Diagrammtypen zur Darstellung von Artefakten, die Modelle repräsentieren.

Im Beispiel wurde, abweichend zur Ontologie, ein weiteres Konzept eingeführt, das Konzept der Vorgabe. Vorgaben entsprechen Ergänzungen zu Arbeitsschritten. Sie repräsentieren jegliche Information, die die Durchführung eines Arbeitsschritts unterstützten könnte. Die Information kann beispielsweise aus der Domänentheorie abgeleitet sein oder ein Prinzip bzw. eine Heuristik repräsentieren. Beispielsweise ergänzt die Vorgabe *17 Kostentreiber* den Arbeitsschritt *Bestimme Kostenfaktor K* durch eine Information zu 17 typischen Kostentreibern, die bei der Berechnung des Kostenfaktors K zu berücksichtigen sind.

Abstrahiert man im Beispiel von der konkreten Methode CoCoMo II, kann ein wieder kehrendes Muster im Aufbau einer Methode identifiziert werden. Wir bezeichnen dieses Muster im Folgenden als Methodenschritt. Im Zentrum eines Methodenschritts steht immer ein Arbeitsschritt. Diesem Arbeitsschritt sind eine Menge (ggf. auch die leere Menge) von Artefakten zur Nutzung und eine Menge von Artefakten zur Produktion zugeordnet. Ein Arbeitsschritt nutzt die von den Artefakten bereitgestellten Informationen um weitere Artefakte als Ergebnisse zu produzieren. Dazu stehen ihm Vorgaben zur Verfügung. Die von einem Arbeitsschritt produzierten Artefakte können von weiteren Arbeitsschritten wieder genutzt werden. Durch diese nutzt- und produziert-Beziehungen zwischen Arbeitsschritten und Artefakten ist der Ablauf einer Methode vollständig festgelegt. Abbildung 5.2 zeigt graphisch den Aufbau eines Methodenschritts als UML Klassendiagramm.

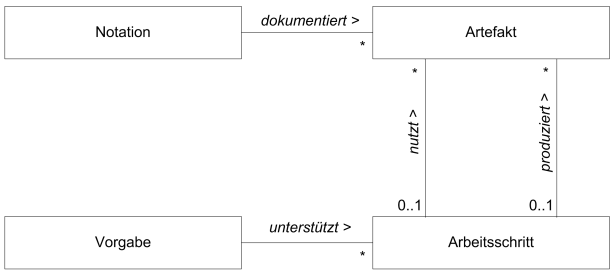

Abbildung 5.2: Aufbau eines Methodenschritts

Ein Methodenschritt kann hierarchisch zu einer Menge von Methodenschritten verfeinert werden, die jeweils wieder den gleichen Aufbau aufweisen. Eine Methode kann somit beschrieben werden als eine Menge von Methodenschritten mit einer Menge von Verfeinerungsfunktionen zwischen Methodenschritten. Strukturell kann eine Methode als eine Menge von Methodenschritt-Bäumen betrachtet werden. Im Folgenden wird das hier eingeführte informelle Modell formal spezifiziert.

5.1.2 Methodenstruktur

In diesem Abschnitt wird die abstrakte Syntax des Methoden-Metamodells vorgestellt. Auf einer sehr allgemeinen Ebene kann eine Methode als eine Menge von Transformationsfunktionen betrachtet werden, die Informationstypen als Eingaben verwenden um weitere Informationstypen als

Ausgabe zu erstellen. Welche Typen an Informationen den Transformationsfunktionen der Methode als Eingaben zur Verfügung stehen müssen und welche Typen von Informationen die Transformationsfunktionen der Methode erstellen, wird durch die der Methode zugrunde liegende Anwendungsdomäne festgelegt und durch eine entsprechende Domänentheorie konkretisiert.

Die Anwendungsdomäne legt fest, ob es sich bei einer Methode beispielsweise um eine Entwicklungsmethode, eine Planungsmethode oder eine Vergabemethode handelt. Dementsprechend kennt eine Anwendungsdomäne für die Systementwicklung den Informationstyp *Komponente*, eine Anwendungsdomäne für den Projektplanung den Informationstyp *Vorgang* und eine Anwendungsdomäne für die Vergabe von Entwicklungsprojekten den Informationstyp *Ausschreibung*. Anwendungsdomänen und Domänentheorien spielen eine zentrale Rolle für die Methodeninhalte und wurden in Abschnitt 4.5 ausführlich diskutiert. Für das Metamodell spielen sie dagegen nur eine untergeordnete Rolle.

Die Menge der Informationstypen einer Anwendungsdomäne ist eindeutig festgelegt und nicht methodenabhängig. Methoden nutzen je nach Zielrichtung und Problemstellung einen spezifischen Ausschnitt der Informationstypen ihrer Anwendungsdomäne. Unterschiedliche Methoden mit gleicher Anwendungsdomäne nutzen dabei die gleichen Informationstypen. Beispielsweise nutzen alle Entwicklungsmethoden einen Ausschnitt der Informationstypen der Anwendungsdomäne *Softwareentwicklung*, alle Planungsmethoden nutzen dagegen einen Ausschnitt der Informationstypen der Anwendungsdomäne *Projektplanung*.

Im Rahmen der Methodendurchführung werden konkrete Informationseinheiten als Methodenergebnisse erzeugt. Die Informationseinheiten sind Instanzen zu den jeweiligen Informationstypen. Eine Entwicklungsmethode benennt beispielsweise den Informationstyp *Komponente* als einen Ergebnistyp. Im Rahmen der Methodendurchführung könnten als konkrete Instanzen der Methode eine Komponente *Kundenverwaltung*, eine Komponente *Vertragsverwaltung* und eine Komponente *Persistenz* erstellt werden.

Eine Methode legt über Transformationsfunktionen und Informationstypen eine Struktur, die eine zielgerichtete Anwendung der Informationstransformationen zur Lösung spezifischer Probleme erlaubt. Als Basiselemente einer Methode wurden in der Ontologie in Abschnitt 4.6 Arbeitsschritte und Artefakte identifiziert. Artefakte repräsentieren Sichten auf die Menge der Informationstypen. Ein Arbeitsschritt entspricht einer Menge von Transformationsfunktionen mit einer Menge von Informationstypen als Eingabeparameter und einer Menge von Informationstypen als Ergebnistypen. Aufgabe eines Arbeitsschritts ist die Erarbeitung von Informationen zu den Informationstypen seiner Ausgabeartefakte unter Verwendung der Informationen zu den Informationstypen seiner Eingabeartefakte. Ein Artefakt ist eindeutig durch die ihm zugeordneten Informationstypen gekennzeichnet.

Zur Dokumentation der Informationseinheiten zu den Informationstypen können bei Bedarf geeignete Notationen vorgegeben werden. Eine Notation entspricht einer Menge von Wörtern über einer

Menge von Symbolen. Die syntaktische Struktur der Wörter wird über eine Grammatik festgelegt. Symbole einer Notation legen für atomare Informationstypen des Artefakts eine visuelle Repräsentation fest, die Grammatik der Notation definiert Regeln für die Bildung gültiger Verknüpfungen über den Informationstypen. Bei den Notationen kann es sich beispielsweise um graphische Notationen oder formale Spezifikationssprachen handeln. Ein Informationstyp kann dabei innerhalb einer Methode unterschiedlichen Artefakten zugeordnet sein. Dementsprechend können seine Instanzen mit unterschiedlichen Notationen dargestellt werden.

Vorgaben liefern Zusatzinformationen, die eine Durchführung von Arbeitsschritten unterstützen. Vorgaben können unterschiedlicher Natur sein. Denkbar sind beispielsweise Regeln und Erfahrungswerte aber auch Zusatzinformationen bezüglich der Domänentheorie. An dieser Stelle werden Vorgaben als atomare Konzepte verwendet. Sie werden nicht hinsichtlich ihrer Art unterschieden und können nicht weiter verfeinert werden.

Zur Beschreibung der abstrakten Syntax von Methoden gehen wir in zwei Schritten vor. In einem ersten Schritt definieren wir den Methodenschritt als grundlegendes Bauelement einer Methode, bevor im Weiteren die Methode selbst definiert wird.

Sei AS die Menge der Arbeitsschritte, AF die Menge der Artefakte, NT die Menge der Notationen und VG die Menge der Vorgaben.

Definition: Ein Methodenschritt ist ein n-Tupel *ms = (as, Af, Nt, Vg, Nutzt, Prod, D, U)*, mit

- Dem Arbeitsschritt $as \in AS$ aus der Menge der Arbeitsschritte.
- Der Menge $Af \subseteq AF$ der vom Arbeitsschritt genutzten und produzierten Artefakte.
- Der Menge $Nt \subseteq NT$ der Notationen die den Artefakten zugeordnet sind.
- Der Menge $Vg \subseteq VG$ der Vorgaben, die den Arbeitsschritten zugeordnet sind.
- Der totalen Funktion $Nutzt : AS \rightarrow \mathcal{P}(Af)$, die einem Arbeitsschritt eine Menge von Artefakten zur Nutzung zuordnet. Einem Arbeitsschritt kann, muss jedoch nicht, eine Menge von Informationstypen zur Nutzung zugeordnet werden. Nicht immer sind Informationen notwendig, um einen Arbeitsschritt durchzuführen.
- Der totalen Funktion $Prod : AS \rightarrow \mathcal{P}(Af) \setminus \emptyset$, die einem Arbeitsschritt eine Menge von Artefakten zur Produktion zuordnet. Ein Arbeitsschritt muss immer mindestens ein Ergebnis in Form eines Artefakts produzieren. So wird sichergestellt, dass ein Arbeitsschritt einen echten Beitrag für das Methodenergebnis leistet.
- Der totalen Funktion $D : Af \rightarrow \mathcal{P}(Nt)$, die einem Artefakt eine Menge von Notationen zuordnet. Einem Artefakt kann, muss jedoch nicht, eine Notation zugeordnet werden. Die Zuordnung einer Notation ist beispielsweise nicht notwendig,

wenn natürliche Sprache zur Dokumentation der Informationseinheiten verwendet werden kann.

- Der totalen Funktion $U : As \rightarrow \mathcal{P}(Vg)$, die einem Arbeitsschritt eine Menge von Vorgaben zuordnet.

Wir bezeichnen als Af_{in}^{ms} die Menge der Eingabeartefakte zum Arbeitsschritt *as* im Methodenschritt *ms*, als Af_{out}^{ms} die Menge der Ausgabeartefakte zum Arbeitsschritt *as* im Methodenschritt *ms* mit

$$Af_{in}^{ms} \cup Af_{out}^{ms} = Af.$$

Für einen wohlgeformten Methodenschritt wird gefordert, dass sich die Menge der Eingabeartefakte eines Arbeitsschritts mit der Menge seiner Ausgabeartefakte nicht überschneiden darf.

$$Af_{in}^{ms} \cap Af_{out}^{ms} = \emptyset$$

Mit dieser Bedingung können Redundanzen und unnötige Zwischenschritte innerhalb einer Methode weitgehend vermieden werden. Diese Forderung gilt nicht für die von den Artefakten referenzierten Informationstypen. Ein Informationstyp der von einem Eingabeartefakt eines Arbeitsschritts referenziert wird, darf durchaus auch von einem Ausgabeartefakt des gleichen Arbeitsschritts referenziert werden. Die Bedingung fordert lediglich, dass eine spezifische Sicht auf die Informationstypen nicht gleichzeitig Teil der Eingabe und Ausgabe zum gleichen Arbeitsschritt sein darf. Ein Arbeitsschritt muss substantielle Änderungen an allen seinen Ausgabeartefakten durchführen. Mit dieser Einschränkung werden in diesem Modell bewusst Fälle wie inhaltliche Änderung an einem bereits vollständigen Artefakt durch einen Arbeitsschritt als gültige Methodenschritte ausgeschlossen.

Ausgehend von der Definition für einen Methodenschritt können wir nun eine Methode definieren. Sei MS die Menge aller denkbaren Methodenschritte.

Definition: Eine Methode ist ein Tupel $m = (Ms,\ Af_{ext},\ V)$, mit:

- Der Menge $Ms \subseteq MS$ von Methodenschritten der Methode.
- Der Menge $Af_{ext} \subseteq \bigcup_{ms \in Ms} Af_{in}^{ms}$ von externen Artefakten. Als externe Artefakte bezeichnen wir Artefakte, die Informationstypen einer Methode referenzieren, deren Instanzen nicht innerhalb der Methode produziert werden. Sie müssen der Methode vielmehr von außen zur Verfügung gestellt werden. Konkret existiert kein Methodenschritt in der Methode, dessen Arbeitsschritt das Artefakt produziert.

  $$af \in Af_{ext} \Leftrightarrow \nexists ms \in Ms : af \in Prod(as_{ms})$$

  Die externen Artefakt einer Methode sind Teilmenge der Eingabeartefakte aller ihrer Methodenschritte.

- Der totalen Funktion $V : Ms \rightarrow \mathcal{P}(Ms)$, die einem Methodenschritt eine Menge von Methodenschritten als untergeordnet zuordnet. Ein Methodenschritt mit $V(ms) = \emptyset$ ist nicht mehr weiter verfeinert.

Die hier vorgestellte Spezifikation betrachtet eine Methode als eine Menge von gleichförmig aufgebauten Methodenschritten. Die Verbindung der Methodenschritte zu einer zusammenhängenden Methode erfolgt über die Eingabe- und Ausgabeartefakte ihrer Arbeitsschritten. Ein Methodenschritt einer Methode kann durchgeführt werden, sobald alle Informationseinheiten zu den Informationstypen seiner Eingabeartefakte vollständig vorliegen.

Methoden können hierarchisch aufgebaut sein. Ein Methodenschritt kann durch eine Menge weiterer Methodenschritte verfeinert werden. Eine Methode kann in dieser Hinsicht als eine Menge von Methodenschrittbäumen betrachtet werden. In der Definition wird die Hierarchie über die Funktion V modelliert. Für eine gültige Verfeinerung müssen eine Reihe von Konsistenzbedingungen gelten.

KB 1: Die Menge der von den Ausgabeartefakten eines übergeordneten Methodenschritts *ms* referenzierten Informationstypen muss vollständig in der Menge der von den Ausgabeartefakten der untergeordneten Methodenschritte *msv* referenzierten Informationstypen enthalten sein.

Für die Spezifikation der Konsistenzbedingung benötigen wir die von den Artefakten einer Methode referenzierten Informationstypen. Sei AF die Menge aller Artefakte. Die totale Funktion

$$aftypen : AF \rightarrow \mathcal{P}(IT)$$

liefert als Ergebnis die Menge der Informationstypen innerhalb der Anwendungsdomäne, die von einem Artefakt referenziert werden.

Sei

$$msv = V(ms_s)$$

die Menge der Methodenschritte einer Verfeinerung zum Methodenschritt ms_s und sei

$$Af_{out}^{msv} = \bigcup_{ms \in msv} Af_{out}^{ms}$$

die Menge aller Ausgabeartefakte der Verfeinerung zum Methodenschritt ms_s. Sei

$$It_{out}^{msv} = \bigcup_{af \in Af_{out}^{msv}} aftypen(af)$$

die Menge der Informationstypen zu den Ausgabeartefakte der Verfeinerung von ms. Für die Mengen der Informationstypen der Ausgabeartefakte muss bei einer Verfeinerung gelten:

$$It_{out}^{ms_s} \subseteq It_{out}^{msv}$$

Mit dieser Bedingung ist sichergestellt, dass bei der Verfeinerung eines Methodenschritts zu einer Menge von Methodenschritten genau die vom übergeordneten Methodenschritt geforderten Ergebnisse erstellt werden.

KB 2: Die Menge der von den Eingabeartefakten aller untergeordneten Methodenschritte referenzierten Informationstypen muss vollständig in der Menge der von den Eingabeartefakten des übergeordneten Methodenschritte referenzierten Informationstypen enthalten sein.

Sei

$$msv = V(ms_s)$$

die Menge der Methodenschritte einer Verfeinerung zum Methodenschritt ms_s und sei

$$Af_{in}^{msv} = \bigcup_{ms \in msv} Af_{in}^{ms}$$

die Menge aller Eingabeartefakte der Verfeinerung zum Methodenschritt ms_s. Sei

$$It_{in}^{msv} = \bigcup_{af \in Af_{in}^{msv}} aftypen(af)$$

die Menge der Informationstypen zu den Eingabeartefakten der Verfeinerung von ms. Für die Mengen der Informationstypen der Eingabeartefakte muss bei einer Verfeinerung gelten:

$$It_{in}^{msv} \subseteq It_{in}^{ms_s} \cup It_{out}^{msv}$$

Mit dieser Bedingung ist sichergestellt, dass bei der Verfeinerung eines Methodenschritts zu einer Menge von Methodenschritten alle von den untergeordneten Methodenschritte geforderten Informationstypen zur Verfügung stehen

KB 3: Die Menge der Vorgaben, die für den übergeordneten Methodenschritt gilt, vererbt sich auf alle Arbeitsschritte der Methodenschritte in der Verfeinerung.

Sei

$$msv = V(ms_s)$$

die Menge der Methodenschritte einer Verfeinerung zum Methodenschritt ms_s. Für jeden Methodenschritt der Verfeinerung gelten zusätzlich zu den unmittelbar zugeordneten Vorgaben immer auch die Vorgaben des übergeordneten Methodenschritts.

$$ms \in msv \Rightarrow Vg_{ms} = Vg_{ms} \cup Vg_{ms_s}$$

KB 4: Für hierarchische Methodenschritte muss gelten, dass innerhalb der Hierarchien keine Zyklen enthalten sind. Konkret darf ein Methodenschritt sich nicht selbst verfeinern. Zur Spezifikation der Bedingung definieren wir die totale Funktion

$$V^* : MS \rightarrow \mathcal{P}(MS)$$

die als Ergebnis alle Methodenschritte liefert, die in der Hülle der Verfeinerung liegen. Die totale Funktion ist wie folgt definiert:

$$ms \in V^*(ms_s) \Leftrightarrow ms \in V(ms_s) \vee \exists ms_x : ms_x \in V(ms_s) \wedge ms \in V^*(ms_x)$$

Für einen hierarchischen Methodenschritt ms_s muss nun gelten, dass er nicht Element seiner eignen Hierarchie sein darf.

$$ms_s \notin V^*(ms_s)$$

Die Verfeinerung einer Methode durch ihre Methodenschritte kann in identischer Weise zur hierarchischen Strukturierung von Methoden verwendet werden. Statt einer Menge von Methodenschritten der gleichen Methode werden einem Methodenschritt die Methodenschritte einer weiteren eigenständigen Methode zugeordnet. Die Bedingungen zur Wohlgeformtheit müssen in gleicher Weise gelten. Diese Form der Modellierung löst die strikten Grenzen von Methoden auf und reduziert Methoden auf die Menge ihrer Einzelschritte.

5.1.3 Methodendurchführung

In diesem Abschnitt wird die Ablaufsemantik einer Methode untersucht und formal beschrieben. Den Ablauf einer Methode bezeichnen wir als Methodendurchführung. Wie gezeigt wird, kann die Durchführung einer Methode unmittelbar über den Formalismus der Petrinetze beschrieben werden. Zur Einführung wird im Folgenden der Formalismus der Petrinetze im Überblick vorgestellt. Anschließend wird die Durchführung einer Methode auf der Basis von Methodenschrittdurchführungen formal spezifiziert und es wird eine Abbildung der Methodendurchführung auf Konzepte eines Petrinetzes vorgestellt.

Petri-Netze

Ein Petrinetz ist ein bipartiter und gerichteter Graph. Er besteht aus Stellen bzw. Plätzen und Übergängen bzw. Transitionen. Stellen und Transitionen sind durch gerichtete Kanten verbunden. Es gibt keine direkten Verbindungen zwischen zwei Stellen oder zwei Transitionen. Petrinetze erweitern Automaten um die Fähigkeit, parallele Prozesse zu modellieren.

> **Petrinetz:** Ein Petrinetz ist ein Tripel $P = (S, T, F)$ mit einer Menge von Stellen S, einer endlichen Menge von Transitionen T und einer Relation F mit $F \subseteq S \times T \cup T \times S$. P bildet einen bipartiten Graphen mit Knoten $S \cup T$ und gerichteten Kanten F.

Die Stellen S repräsentieren Zustände bzw. Bedingungen des modellierten Systems und werden typischerweise als Kreise dargestellt, die Transitionen repräsentieren Zustandsübergänge bzw. Aktivitäten und werden als Rechtecke dargestellt.

> **Markierung:** Der Zustand eines Petrinetzes wird durch eine Markierung angegeben. Eine Markierung ist eine Funktion $M_p : S \rightarrow \mathbb{N}$, die jeder Stelle eine Anzahl an Marken zuordnet.

Marken werden im Graphen als Punkte in den Stellen dargestellt. Die Markierung in einem Petrinetz definiert den Zustand zu einem bestimmten Zeitpunkt. Zustandsänderungen treten ein, wenn Transitionen schalten und so die Markierung ändern.

> **Vor- und Nachbereich:** Zu einer Transition t in einem Petrinetz P sind zwei Mengen von Stellen definiert:
> Vorbereich(t): $= s|(s,t) \in F$
> Nachbereich(t): $= s|(t,s) \in F$

Mit Hilfe der Vor- und Nachbereiche einer Transition kann die Dynamik von Petrinetzen formalisiert werden.

> **Schaltregeln:** Eine Transition t eines Petrinetzes P kann schalten, wenn für alle Stellen $s \in Vorbereich$ gilt $M(s) \geq 1$. Wenn zu einem Zeitpunkt mehrere Transitionen schalten können, wird eine davon nichtdeterministisch ausgewählt.
> Wenn bei einer Markierung M eine Transition schaltet, gilt für die Nachfolgemarkierung M':
> $M'(v) = M(v) - 1$ für alle $v \in Vorbereich(t) - Nachbereich(t)$
> $M'(n) = M(n) + 1$ für alle $n \in Nachbereich(t) - Vorbereich(t)$
> $M'(s) = M(s)$ für alle übrigen Stellen

Transitionen können miteinander konkurrieren. Dieser Fall tritt ein wenn sich die Vorbereiche der Transitionen überschneiden.

> **Konflikt:** Zwei Transitionen t_1 und t_2 stehen im Konflikt, wenn
> $Vorbereich(t_1) \cap Vorbereich(t_2) \neq \emptyset$

Ein Konflikt hat zur Folge, dass zwei Transitionen mit gleichem Vorbereich niemals gleichzeitig schalten können.

Diese kurze Einführung in Petrinetze dient als Grundlage für die im Folgenden spezifizierte Semantik der Methodendurchführung. Für eine detaillierte Beschreibung verweisen wir auf die entsprechende Literatur (z.B. [73]).

Methodendurchführung als semantische Domäne

Wir definieren die Semantik einer Methode über die Menge ihrer zulässigen Methodendurchführungen. Eine Methodendurchführung entspricht einer Menge von Methodenschrittdurchführungen. Jede Methodenschrittdurchführung entspricht dem Prozess der Durchführung eines Methodenschritts der Methode. Als Basiskonzepte einer Methodenschrittdurchführung definieren wir die Menge der Artefaktinstanzen AI und die Menge der Prozessschritte PS. Wir definieren eine Methodenschrittdurchführung wie folgt:

> **Definition:** Eine Methodenschrittdurchführung ist ein n-Tupel *msd = (ps, Ai, N, P)* mit
>
> - Einem Prozessschritt $ps \in PS$ als konkreten Prozessschritt der Methodenschrittdurchführung.
> - Einer Menge $Ai \subseteq AI$ von Artefaktinstanzen der Methodenschrittdurchführung.
> - Der totalen Funktion $N : PS \rightarrow \mathcal{P}(Ai)$, die dem Prozessschritt ps eine Menge von Artefaktinstanzen zur Nutzung zuordnet.
> - Der totalen Funktion $P : PS \rightarrow \mathcal{P}(Ai) \setminus \emptyset$ die dem Prozessschritt ps eine Menge von Artefaktinstanzen zur Produktion zuordnet.

Eine Methodenschrittdurchführung kann unmittelbar auf einen Ausschnitt in einem Petrinetz abgebildet werden: der Prozessschritt einer Methodenschrittdurchführung entspricht einer Transition im Petrinetz, die Artefaktinstanzen einer Methodenschrittdurchführung entsprechen den Stellen. Die Menge aller Artefaktinstanzen, die dem Prozessschritt *ps* über die Funktion N zugeordnet sind, entspricht dem Vorbereich des Prozessschritts *ps*. Die Menge aller Artefaktinstanzen, die dem Prozessschritt über die Funktion P zugeordnet sind, entspricht dem Nachbereich von *ps*. Für eine gültige Methodenschrittdurchführung muss gelten, dass sich Vorbereich und Nachbereich eines Prozessschritts nicht überschneiden dürfen. Konkret bedeutet das, die Mengen der Artefaktinstanzen im Vor- und im Nachbereich eines Prozessschritts dürfen sich nicht überschneiden.

$$N(ps) \cap P(ps) = \emptyset$$

Diese Bedingung ist erforderlich, um Deadlocks innerhalb der Methodendurchführung zu vermeiden. Sie stellt sicher, dass ein Prozessschritt nicht auf die Fertigstellung von Artefaktinstanzen wartet, die von ihm selbst zu erstellen sind.

Einem Artefakte ist eine Menge von Informationstypen zugeordnet, einer Artefaktinstanz ist dementsprechend eine Menge von Informationseinheiten als Instanzen von Informationstypen zugeordnet. Eine Informationseinheit ist ein konkretes Stück Information bzw. ein konkretes Datum, das im Rahmen der Methodendurchführung von einem Prozessschritt verwendet oder erzeugt wird. Ein Informationstyp kann innerhalb einer Artefaktinstanz mehrfach instanziiert werden.

Die einer Artefaktinstanz zugeordneten Informationseinheiten entsprechen in Petrinetz-Terminologie den Marken zu einer Stelle. Wurden alle erforderlichen Informationseinheiten zu den Artefaktinstanzen im Vorbereich eines Prozessschritts erstellt, kann der Prozessschritt durchgeführt werden, d.h. die Transition im Petrinetz kann schalten. Als Ergebnis werden alle geforderten Informationseinheiten zu allen Artefaktinstanzen erstellt und entsprechen den Marken im Nachbereich der Transition.

Eine Artefaktinstanz ist Instanz eines Artefakts einer Methode. Die totale Funktion

$$aiaf : AI \rightarrow AF$$

ordnet einer Artefaktinstanz ihr Artefakt als Typ zu. Die Informationseinheiten der Artefaktinstanz müssen Instanzen der Informationstypen des Artefakts sein.

Einen Prozessschritt definieren wir als Instanz eines Arbeitsschritts einer Methode. Die totale Funktion

$$psas : PS \rightarrow AS$$

ordnet einem Prozessschritt einen Arbeitsschritt als Typ zu.

Für die Abbildung der Funktionen auf ihre Typen definieren wir die totale Funktion

$$nnutzt : N \rightarrow Nutzt$$

die einer Funktion aus der Menge N zwischen einem Prozessschritt und einer Artefaktinstanz eine Funktion aus der Menge *Nutzt* zwischen einem Arbeitsschritt und einem Artefakt als Typ zuordnet, sowie die Funktion

$$pprod : P \rightarrow Prod$$

die einer Funktion aus der Menge P zwischen einem Prozessschritt und einer Artefaktinstanz eine Funktion aus der Menge *Prod* zwischen einem Arbeitsschritt und einem Artefakt als Typ zuordnet.

Mit diesen Hilfsfunktionen können wir nun die Beziehung zwischen Methodenschritten und ihren Methodenschrittdurchführungen spezifizieren. Eine Methodenschrittdurchführung repräsentiert innerhalb der Methodendurchführung genau einen Methodenschritt der Methode. Die Funktion *rep* beschreibt nun den Bezug zwischen Methodenschritt und Methodenschrittdurchführung.

$$rep : MSD \rightarrow MS$$

mit folgender Definition:

Sei

$$msd = (ps,\ Ai_{msd},\ N_{msd},\ P_{msd})$$

eine Methodenschrittdurchführung und

$$ms = (as,\ Af_{ms},\ Nt_{ms},\ Vg_{ms},\ Nutzt_{ms},\ Prod_{ms},\ D_{ms},\ U_{ms})$$

ein Methodenschritt. Dann gilt:

$$\begin{aligned}
&rep(msd) = ms \Leftrightarrow psas(ps) = as \quad \wedge \\
&\forall ai \in Ai_{msd} : aiaf(ai) \in Af_{ms} \quad \wedge \\
&\forall af \in Af_{ms} : \exists ai \in Ai_{msd} : aiaf(ai) = af \quad \wedge \\
&\forall n \in N_{msd} : nnutzt(n) \in Nutzt_{ms} \quad \wedge \\
&\forall nutzt \in Nutzt_{ms} : \exists n \in N_{msd} : nnutzt(n) = nutzt \quad \wedge \\
&\forall p \in P_{msd} : pprod(n) \in Prod_{ms} \quad \wedge \\
&\forall prod \in Prod_{ms} : \exists p \in P_{msd} : nnutzt(p) = prod
\end{aligned}$$

Methodenschrittdurchführungen repräsentieren dabei nur einen Teil eines Methodenschritts. Notationen und Vorgaben fallen auf Prozessebene weg. Diese Konzepte unterstützen zwar die Methodendurchführung, werden hier jedoch nicht als Teil des Prozesses selbst eingeordnet. Konkret existieren keine Instanzen dieser Konzepte innerhalb der Methodendurchführung.

Mit Hilfe der Funktion *rep* kann nun die Beziehung zwischen einer Methode und ihren Methodendurchführungen definiert werden. Sei MSD die Menge der Methodenschrittdurchführungen. Wir definieren eine Methodendurchführung $md \in \mathcal{P}(MSD)$ als eine Menge von Methodenschrittdurchführungen. Sei MD die Menge der Methodendurchführungen und M die Menge der Methoden. Die totale Funktion

$$repm : MD \rightarrow M$$

ordnet einer Methode eine Methodendurchführung zu. Die Methodendurchführung muss gültig für die Methode sein und sie muss die Methode vollständig umsetzen. Zur Sicherstellung dieser Eigenschaften definieren wir zwei Hilfsprädikate.

Das Prädikat

$$gueltig : MD \times M \rightarrow Boolean$$

kennzeichnet, dass alle Methodenschrittdurchführungen einer Methodendurchführung einen Methodenschritt der Methode repräsentieren. Das Prädikat ist wie folgt definiert:

$$gueltig(md, m) \Leftrightarrow \forall msd \in md : \exists ms \in Ms : rep(msd) = ms$$

Das Prädikat

$$vollstaendig : MD \times M \rightarrow Boolean$$

kennzeichnet, dass die Methodenschrittdurchführungen einer Methodendurchführung die Methodenschritte der Methode vollständig repräsentieren.

Für die Vollständigkeit einer Methodendurchführung muss zusätzlich der Aspekt der Hierarchisierung von Methodenschritten berücksichtigt werden. Methodenschritte können Hierarchien bilden. Da der Prozess die reale Durchführung der Methode beschreibt, ist eine Hierarchisierung der Methodendurchführung nicht mehr erforderlich. Wir definieren daher die Methodendurchführung als flachen Prozess. Damit eine Methodendurchführung eine Methode vollständig repräsentiert, muss gelten, dass ihre Methodenschrittdurchführungen ausschließlich und vollständig die Blätter der Methodenschrittbäume der Methode repräsentieren, d.h. ausschließlich Methodenschritte, die nicht weiter hierarchisiert sind.

Sei Ms die Menge der Methodenschritte einer Methode m und Msd die Menge der Methodenschrittdurchführungen einer Methodendurchführung md. Wir definieren das Prädikat wie folgt:

$$vollstaendig(md, m) \Leftrightarrow \forall ms \in Ms : V(ms) = \emptyset : \exists msd \in Msd : rep(msd) = ms$$

Für die Funktion *repm* gilt somit die Definition:

$$repm(md) = m \Leftrightarrow gueltig(md, m) \wedge vollstaendig(md, m)$$

5.2 Methoden im Vorgehensmodell

In den folgenden Abschnitten wird nun die Einbettung von Methoden in ein Vorgehensmodell untersucht und formal beschrieben. Zur informellen Beschreibung der Integration greifen wir auf das allgemeine Konzept eines Prozessmodells zurück. Ein Prozessmodell definiert eine Sprache zur Beschreibung von Prozessen einer spezifischen Anwendungsdomäne. Das Prozessmodell legt über Konzepte und Beziehungen die Menge der im Bezug auf das Prozessmodell gültigen Prozesse innerhalb der Anwendungsdomäne fest. Instanzen des Prozessmodells sind konkrete Prozesse, die innerhalb der Anwendungsdomäne durchgeführt werden. Als Anwendungsdomäne bezeichnen wir einen abgrenzbaren Anwendungsbereich innerhalb der realen Welt. Sowohl Vorgehensmodelle wie Methoden beschreiben Prozesse innerhalb ihrer Anwendungsdomäne und können somit als Prozessmodelle eingeordnet werden. Die Integration von Methoden in ein Vorgehensmodell entspricht demnach einer Integration von Prozessmodellen in ein Prozessmodell.

Das Vorgehensmodell als übergeordnetes Prozessmodell beschreibt den Prozess zur organisatorischen Durchführung eines Projekts, Methoden als eigenständige Prozessmodelle beschreiben die Erstellung einzelner Ergebnisse im Projekt. Ein um integrierte Methoden erweitertes Vorgehensmodell ist wieder ein Prozessmodell, das eine Hierarchie von untergeordneten Prozessmodellen umfasst. Kritisch für die Integration ist die Erhaltung der Redundanzfreiheit und der Konsistenz des resultierenden hierarchischen Prozessmodells. Das bedeutet konkret:

- Die integrierten Methoden müssen untereinander konsistent sein, d.h. die Methoden müssen so aufeinander abgestimmt sein, dass ein durchgängiger, widerspruchsfreier Prozess der Projektdurchführung möglich ist.
- Die integrierten Methoden müssen konsistent zu den Produkten und Aktivitäten im Vorgehensmodell sein.
- Die integrierten Methoden dürfen keine Redundanzen untereinander aufweisen, d.h. ein vom Vorgehensmodell erwartetes Ergebnis darf nicht parallel von verschiedene Methoden geliefert werden.

Zur Prüfung der Konsistenz und Redundanzfreiheit müssen die Anwendungsdomänen von Vorgehensmodellen und Methoden betrachtet werden. Anwendungsdomäne eines Vorgehensmodells ist die Durchführung von Entwicklungsprojekten. Die Anwendungsdomäne einer Methode in einem Vorgehensmodell entspricht einem Teil der Anwendungsdomäne des Vorgehensmodells. Eine Methode kann nicht sinnvoll in ein Vorgehensmodell integriert werden, wenn ihre Anwendungsdomäne nicht auch Teil der Anwendungsdomäne des Vorgehensmodells ist. Abbildung 5.3 illustriert den Zusammenhang zwischen Vorgehensmodellen, Methoden und ihren Anwendungsdomänen.

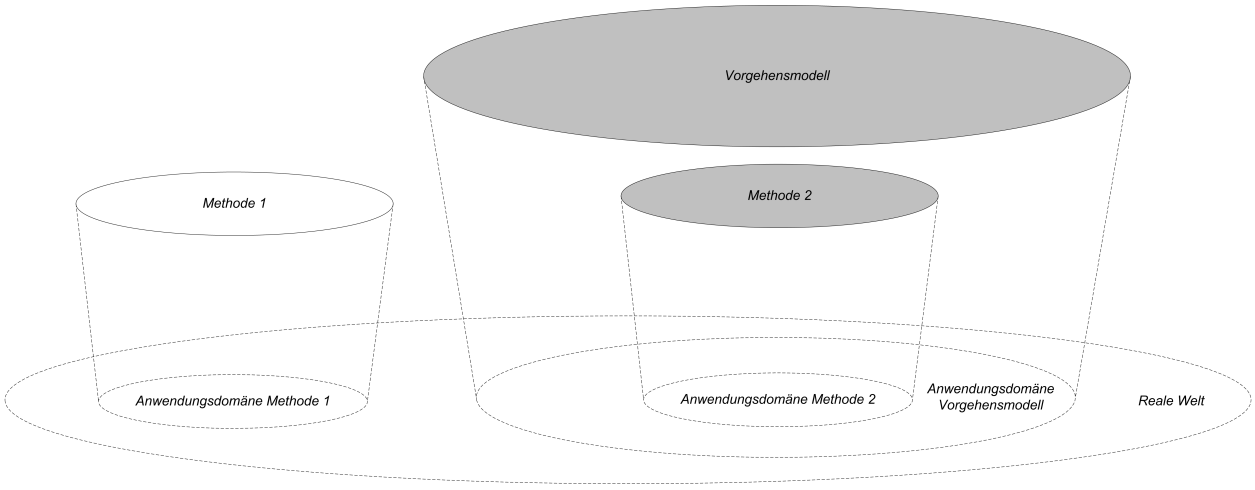

Abbildung 5.3: Rolle der Anwendungsdomänen bei der Methodenintegration

Damit eine Methode sinnvoll in ein Vorgehensmodell integriert werden kann, muss die Anwendungsdomäne der Methode einem Ausschnitt der Anwendungsdomäne des Vorgehensmodells entsprechen. Methode 2 in der Graphik kann sinnvoll integriert werden, da ihre Anwendungsdomäne

sich mit der Anwendungsdomäne des Vorgehensmodells überdeckt. Eine Integration von Methode 1 ist dagegen nicht sinnvoll. So macht es beispielsweise keinen Sinn, eine Vergabemethode in ein Vorgehensmodell zu integrieren, das die Durchführung von Vergabeprojekten nicht unterstützt. Die Anwendungsdomäne der Projektvergabe, die der Methode zu Grunde liegt, wäre in diesem Fall nicht Teil der Anwendungsdomäne des Vorgehensmodells. Sinnvoll ist dagegen die Integration einer objektorientierten Entwicklunsmethode in ein Vorgehensmodell, das die Entwicklung von Informationssystemen unterstützt. Hier entspricht die Anwendungsdomäne der Methode einem Teil der Anwendungsdomäne des Vorgehensmodells. Bei einer teilweisen Überdeckung der Anwendungsdomänen kann nur der Teil einer Methode integriert werden, dessen Anwendungsdomäne sich mit der Anwendungsdomäne des Vorgehensmodells überdeckt.

5.3 Definition der Anforderungen

Andockstellen für die Integration von Methoden in ein Vorgehensmodell sind Aktivitäten und Produkte. Eine Methode liefert mit ihrem Arbeitsschrittmodell eine detaillierte Prozessbeschreibung für die Durchführung einzelner Aktivitäten im Vorgehensmodell, das Artefaktmodell der Methode liefert die von den Produkten geforderten Ergebnisse.

Zur Beschreibung der Schnittstelle zwischen Vorgehensmodell und Methoden stützen wir uns im Weiteren auf das Konzept der Domänentheorie. In Abschnitt 4.5 wurde bereits der Begriff einer Domänentheorie für Methoden eingeführt. Eine Domänentheorie liefert für eine Methode eine Sprache zur Formulierung von Ausdrücken über der Anwendungsdomäne. Insbesondere liefert eine Domänentheorie die innerhalb der Anwendungsdomäne relevanten Informationstypen. In ähnlicher Weise definiert eine Domänentheorie für Vorgehensmodelle eine Sprache zur Formulierung von Ausdrücken im Bezug auf die Projektdurchführung. Die Domänentheorie des Vorgehensmodells legt entsprechend zur Domänentheorie einer Methode die Menge der für ein Vorgehensmodell relevanten Informationstypen fest. Produkte im Vorgehensmodell entsprechen Sichten auf die Menge der Informationstypen. Ein Informationstyp kann von verschiedenen Produkten im Vorgehensmodell referenziert werden. Typischer Informationstyp eines Vorgehensmodells für Entwicklungsprojekte ist beispielsweise die *funktionale Anforderung*. In einem Vorgehensmodell, das die Vergabe von Projekten unterstützt, wird der Informationstyp *funktionale Anforderung* sowohl von einem Produkt *Lastenheft* wie auch von einem Produkt *Pflichtenheft* referenziert: im Rahmen der Projektdurchführung werden die konkreten funktionalen Anforderungen für ein Softwaresystem bei der Erstellung eines Lastenhefts identifiziert und beschrieben, bei der Erstellung des Pflichtenhefts werden sie überarbeitet und verfeinert.

Ein Vorgehensmodell referenziert über seine Produkte Mengen von Informationstypen, zu denen im Rahmen der Projektdurchführung konkrete Informationseinheiten zu erstellen sind. Die Informationstypen definieren die Anforderungen des Vorgehensmodells an potentielle Methoden. Eine Methode muss, damit sie sinnvoll integriert werden kann, mit den von ihren Artefakten referen-

zierten Informationstypen die von den Produkten geforderten Informationstypen mindestens abdecken. Beispielsweise erwartet eine Aktivität *Schätzung durchführen* mit dem Produkt *Schätzung* von einer Methode mindestens Artefakte, die Informationstypen zu *Problemumfang* und *geschätztem Aufwand* liefern. Als geeignete Methode würde CoCoMo II die Informationstypen *Größe des Schätzgegenstands* und *Aufwand in PM* liefern. Die Übereinstimmung der Informationstypen ist in diesem Beispiel intuitiv erkennbar.

Vorgehensmodelle unterscheiden sich hinsichtlich der Detaillierungstiefe, in der sie ihre Anforderungen formulieren. Je detaillierter ein Vorgehensmodell Vorgaben zu den von ihm erwarteten Informationstypen macht, desto stärker wird das Spektrum der verfügbaren Methoden eingeschränkt. Vorgehensmodelle, die in ihren Anforderungen eher allgemein bleiben, sind wesentlich flexibler hinsichtlich der Methodenauswahl und daher in vielen Kontexten einsetzbar. Ein Vorgehensmodell, das dagegen sehr konkrete Anforderungen an Methoden stellt, erlaubt nur wenig Freiheiten bei der Methodenwahl. Es garantiert jedoch eine gewisse Mindestqualität der angewendeten Methoden.

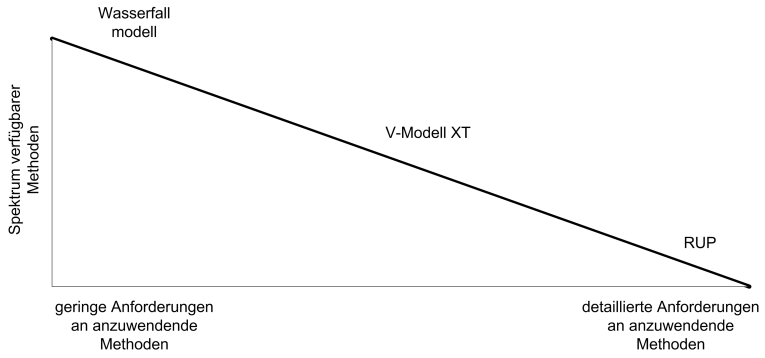

Abbildung 5.4: Einfluss der Anforderungskonretisierung auf die Methodenauswahl

Abbildung 5.4 ordnet beispielhaft einige Phasen- und Vorgehensmodelle hinsichtlich ihrer Detaillierungstiefe bei der Definition der Anforderungen an Methoden ein. Phasenmodelle wie das Wasserfallmodell [99] oder das Spiralmodell [18] bleiben in ihren Anforderungen so allgemein, dass sie in vielen Kontexten mit unterschiedlichen Methoden eingesetzt werden können. Diese Modelle benennen üblicherweise die erwarteten Ergebnisse, verzichten jedoch auf eine konkrete Formulierung der angeforderten Inhalte. Ein Wasserfallmodell spricht beispielsweise sehr allgemein von einem Anforderungsdokument als Ergebnis der Analysephase ohne konkrete Angabe zu weiteren Informationstypen. Das V-Modell XT ist dagegen konkreter in seinen Anforderungen an Methoden. Insbesondere definiert es über die Themenstruktur der Produkte detailliertere Anforderungen an Methoden. Als ein Spezialfall können Vorgehensmodelle wie der Rational Unified Process [77] eingeordnet werden. Diese Vorgehensmodelle sind in ihren Anforderungen so detailliert, dass eine Auswahl von Methoden nicht mehr sinnvoll möglich ist. Die zu verwendenden Methoden werden vom Vorgehensmodell direkt mit vorgegeben.

5.4 Erfüllung der Anforderungen

Ein Vorgehensmodell definiert seine Anforderungen an Methoden über Informationstypen. Eine Methode sichert über die Informationstypen seiner Artefakte zu, dass im Rahmen der Methodendurchführung die entsprechenden Informationseinheiten erstellt werden und so die im Projekt erwarteten Produktinstanzen vorliegen.

Zur Beschreibung der Beziehung zwischen den von einem Produkt geforderten Informationstypen und den von einer Methode bereitgestellten Informationstypen nutzen wir das aus der Objektorientierung bekannte Konzept der Spezialisierung (vgl. [82]). Als Spezialisierung wird die Beziehung zwischen einer Entität als Supertyp und einer Menge von Entitäten als Subtypen bezeichnet, wobei für jeden Subtypen gilt, dass er mindestens die Eigenschaften des Supertyps erfüllt. In unserem konkreten Fall wird ein Informationstyp B durch einen Informationstypen A spezialisiert, wenn A mindestens alle Eigenschaften von B erfüllt und diese gegebenenfalls ergänzt. Anhand der Spezialisierungsbeziehung können wir nun die Voraussetzungen für die sinnvolle Integration einer Methoden in ein Vorgehensmodell konkret beschreiben. Damit eine Methode zur Erarbeitung eines Produkts im Vorgehensmodell herangezogen werden kann, müssen die Informationstypen ihrer Artefakte Spezialisierungen der vom Produkt geforderten Informationstypen sein.

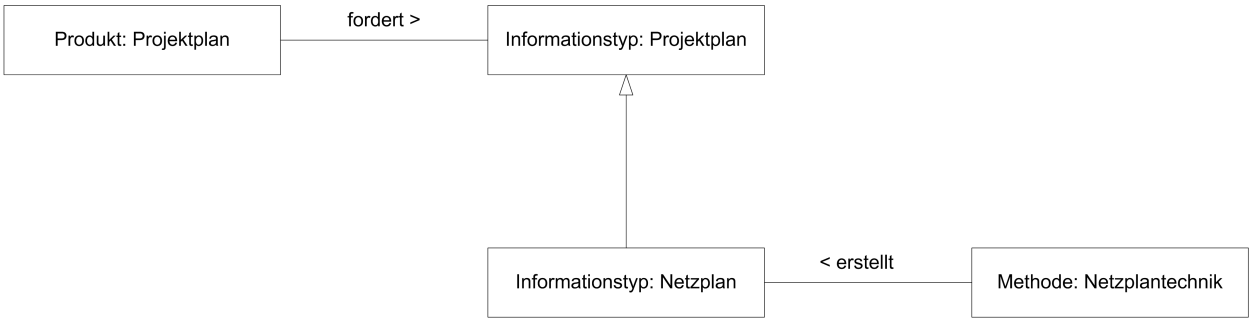

Abbildung 5.5: Spezialisierung von Informationstypen

Abbildung 5.5 stellt diesen Zusammenhang am Beispiel der Netzplantechnik dar. Ein Produkt *Projektplan* im einem Vorgehensmodell fordert einen allgemeinen Informationstyp *Projektplan*. Die Netzplantechnik als Methode liefert ein Artefakt *Netzplan* mit dem Informationstyp *Netzplan*, bestehend aus weiteren Informationstypen: dem *Vorgang* und der *Anordnungsbeziehung*. Ein Netzplan ist eine mögliche Spezialisierung eines Projektplans. Es kann jedoch auch andere Planungsmethoden geben, die ebenfalls geeignete Spezialisierungen eines Projektplans liefern. Entscheidend ist, welche Informationstypen das Produkt im Vorgehensmodell fordert. Wird im Vorgehensmodell bereits explizit ein Netzplan gefordert, können nur Planungsmethoden integriert werden, die Varianten der Netzplantechnik sind.

Kritisch für die Zuordnung von Methoden ist die Sicherstellung der Konsistenz des resultierenden integrierten Vorgehensmodells. Insbesondere können durch inhaltliche Überlappungen von Metho-

den bei der Zuordnung Redundanzen in den Produktinstanzen auftreten. Werden die Artefakte unterschiedlicher Methoden ein und demselben Produkt zugeordnet, können durch mehrfache Spezialisierungen von Informationstypen Redundanzen auftreten. Beispielsweise unterstützen viele Entwurfsmethoden in Teilen auch den Analyseprozess und erarbeiten entsprechende Artefakte. Bei einer Kombination von Analysemethode und Entwurfsmethode kann so die Situation auftreten, dass Informationstypen für die Analyse von zwei unterschiedlichen Methoden geliefert werden. Für eine gültige Integration muss somit sichergestellt sein, dass ein solcher Fall nicht auftreten kann. Konkret darf ein Informationstyp in einem Vorgehensmodell nur einmal durch den Informationstyp einer Methode spezialisiert werden.

5.5 Formalisierung der Integration

In diesem Abschnitt wird nun abschließend die Integration von Methoden in ein Vorgehensmodell formal spezifiziert. Insbesondere werden Konsistenzbedingungen bezüglich der Informationstypen an der Schnittstelle definiert, die bei einer Integration erfüllt sein müssen. Die Formalisierung dient unter anderem als Vorlage für das semi-formale Metamodell der Integration in Kapitel 6. Zur Integration wird für das Vorgehensmodell angenommen, dass es auf einem produktorientierten Metamodell basiert (vgl. Abschnitt 3.6). Durch Aktivitäten und Produkte wird somit keine feste Ablaufstruktur vorgegeben, die im Widerspruch zur Ablaufstruktur der zugeordneten Methoden stehen könnte. In diesem Abschnitt wird keine Formalisierung des Vorgehensmodells selbst durchgeführt. Es wird lediglich die Schnittstelle des Vorgehensmodells zu den Methoden, bestehend aus Produkten und Aktivitäten, formal beschrieben.

Produkt- und Aktivitätsschnitt im Vorgehensmodell orientieren sich an organisatorischen Aspekten, wie beispielsweise der Rollenzuordnung oder der Meilensteinabfolge. Ein Produkt Lastenheft umfasst beispielsweise alle Projektergebnisse, die zu einem spezifischen Meilenstein in einem Projekt hinsichtlich des zu entwickelnden Systems zu erstellen sind. Diese können beispielsweise die Anforderungen an das System, ein Vorschlag für eine erste Systemarchitektur, die wichtigsten Abnahmekritierien sowie Lieferbedingungen umfassen. Jedes dieser Ergebnisse kann durch unterschiedliche Methoden oder Teile von Methoden erstellt werden. Bei einer Integration von Methoden in ein Vorgehensmodell stehen daher immer die Anforderungen der Produkte im Vorgehensmodell im Vordergrund, nicht die Methoden als vollständige und unteilbare Einheiten. Auch Zwischenergebnisse von Methoden können Anforderungen von Vorgehensmodellen erfüllen.

Auf eine unmittelbare Zuordnung von Methoden zu Aktivitäten und Produkten wird daher verzichtet. Vielmehr spezifizieren wir die Integration von Methoden in ein Vorgehensmodell über eine Zuordnung von Methodenschritten zu Aktivitäten und Produkten. Die Methodenschritte können dabei von unterschiedlichen Methoden stammen. Diese Form der Modellierung löst die strikten Methodengrenzen auf, zu Gunsten einer flexiblen Kombination von Methodenschritten. Konkret spezifizieren wir die Integration von Methoden in ein Vorgehensmodell durch eine Zuordnung von

Methodenschritten zu einer Menge von Aktivitäten im Vorgehensmodell mit den von ihnen zu erstellenden Produkten. Die Produkte im Vorgehensmodell definieren über ihre Informationstypen Anforderungen an die Artefakte der zugeordneten Methodenschritte, die Artefakte erfüllen diese Anforderungen über die von ihnen referenzierten Informationstypen, die gültige Spezialisierungen der von den Produkten geforderten Informationstypen sind. Die Arbeitsschritte der Methodenschritte dienen zur Ausgestaltung der Aktivitäten im Vorgehensmodell. Alle Methodenschritte, deren Artefakte Ergebnisse für ein Produkt liefern, werden der Aktivität zum Produkt zugeordnet. Das Ablaufmodell einer Aktivität beschreibt somit keinen durchgängigen Prozess, sondern eine Menge durchzuführender Prozessschritte.

Zur Prüfung, ob Methodenschritte die Anforderungen des Vorgehensmodells erfüllen, benötigen wir einige Hilfsfunktionen. Die Funktion

$$spez : IT \times IT \rightarrow Boolean$$

gibt an, ob ein Informationstyp eine gültige Spezialisierung eines anderen Informationstypen darstellt. Sie ist wie folgt definiert:

$spez(it_1, it_2) \Leftrightarrow it_1$ ist eine gültige Spezialisierung von it_2.

Auf eine Formalisierung der Eigenschaft *ist eine gültige Spezialisierung von* wird, da im Weiteren nicht mehr benötigt, an dieser Stelle verzichtet. Statt dessen wird auf die informelle Beschreibung der Spezialisierung in Abschnitt 5.4 verwiesen.

Des Weiteren wird ähnlich zu den Artefakten einer Methode eine Funktion benötigt, die alle Informationstypen zu einem Produkt liefert. Sei Pr die Menge der Produkte in einem Vorgehensmodell. Die Funktion

$$ptypen : Pr \rightarrow \mathcal{P}(IT)$$

liefert als Ergebnis die Menge der von einem Produkt referenzierten Informationstypen. Wir können nun die Zuordnung von Methodenschritten zu Aktivitäten und Produkten im Vorgehensmodell spezifizieren.

Definition: Eine Methodenintegration ist ein n-Tupel *mi = (Ak, Pr, Ms, Erst, Erw)* mit

- Einer Menge *Ak* von Aktivitäten im Vorgehensmodell.
- Einer Menge *Pr* von Produkten im Vorgehensmodell.
- Einer Menge *Ms* von Methodenschritten.

- Der totalen Funktion $Erst : Ak \rightarrow \mathcal{P}(Pr)$, die einer Aktivität eine Menge von Produkten zur Erstellung zuordnet.
- Der totalen Funktion $Erw : Ak \rightarrow \mathcal{P}(Ms)$, die einer Aktivität eine Menge von Methodenschritten zuordnet. Für hierarchische Methodenschritte gilt, dass alle untergeordneten Methodenschritte ebenfalls der Aktivität zugeordnet sind. Für alle $ms \in Erw(ak)$ gilt:

  $$\forall ms_i \in V^*(ms) \Rightarrow ms_i \in Erw(ak).$$

Damit ein Methodenschritt einer Aktivität und ihren Produkten zugeordnet werden kann, müssen die Ausgabeartefakte des Methodenschritts Teile der Anforderungen der Produkte zur Aktivität erfüllen.

Sei

$$ak \in Ak$$

eine Aktivität und

$$ms \in Ms$$

ein Methodenschritt, der der Aktivität ak zugeordnet ist:

$$ms \in Erw(ak).$$

Sei as der Arbeitsschritt im Methodenschritt ms und

$$Af_{out}^{ms} = Prod(as)$$

die Menge der Ausgabeartefakte zum Arbeitsschritt as. Die Menge

$$It_{out}^{ms} = \bigcup_{af \in Af_{out}^{ms}} aftypen(af)$$

enthält dann alle von den Ausgabeartefakten zum Arbeitsschritt as referenzierten Informationstypen.

Sei

$$P_{ak} = Erst(ak)$$

die Menge der Produkte der Aktivität und

$$It_{ak} = \bigcup_{p \in P_{ak}} ptypen(p)$$

die Menge der Informationstypen aller Produkte zu einer Aktivität *ak*. Für die Zuordnung von Methodenschritten zu einer Aktivität muss gelten, dass die Menge der Ausgabeartefakte über ihre Informationstypen die Anforderungen der Produkte zur Aktivität vollständig erfüllt. Konkret muss gelten, dass die Informationstypen der Ausgabeartefakte der zugeordneten Methodenschritte vollständig die Informationstypen der Produkte zur Aktivität spezialisieren. Sei

$$It_{out} = \bigcup_{ms \in Erw(ak)} It^{ms}_{out}$$

die Menge aller Informationstypen, die von den Ausgabeartefakten aller einer Aktivität *ak* zugeordneten Methodenschritte referenziert werden. Für eine vollständige Integration der Methodenschritte zur Aktivität gilt:

$$\forall it_p \in It_{ak} : \exists it \in It_{out} : spez(it, it_p).$$

Bei der Integration wird dagegen nicht weiter berücksichtigt, dass Methodenschritte ggf. mehr Informationstypen abdecken können, als von den Produkten gefordert.

Das hier vorgestellte Modell einer Methodenintegration geht von einer vollständigen Abdeckung methodischer Aspekte eines Vorgehensmodells durch die integrierten Methoden und ihre Methodenschritte aus. Dies ist in der Praxis nicht unbedingt realistisch, für das Verständnis der Integration jedoch hilfreich. Das Metamodell in Kapitel 6 wird daher eine vollständige Integration unterstützen, jedoch auch eine partielle Abdeckung der Produkte im Vorgehensmodell durch Artefakte von Methoden zulassen.

Die Semantik eines Vorgehensmodells mit vollständig integrierter Methodik kann nun einfach über Methodenschrittdurchführungen spezifiziert werden. Bei einer vollständigen Integration von Methoden in ein Vorgehensmodell werden alle Aktivitäten im Vorgehensmodell durch die Arbeitsschritte der zugeordneten Methodenschritte abgedeckt, die Produkte im Vorgehensmodell werden vollständig durch die Artefakte der Methodenschritte repräsentiert. Die Durchführung eines Projekts auf Basis eines Vorgehensmodells mit integrierten Methoden kann daher vollständig durch die Menge aller Methodenschrittdurchführungen der zugeordneten Methodenschritte dargestellt werden. Sei *Msd* die Menge aller Methodenschrittdurchführungen und *Proj* die Menge aller Projektdurchführungen. Eine Projektdurchführung $proj \in Proj$ entspricht einer Menge von Methodenschrittdurchführungen:

$$proj \in \mathcal{P}(Msd).$$

Ähnlich zu Methodendurchführungen ist eine Projektdurchführung gültig bzw. vollständig bezüglich eines Vorgehensmodells. Sei *Vm* ein Vorgehensmodell und *Ms* die Menge aller dem Vorgehensmodell zugeordneten Methodenschritte. Wir bezeichnen eine Projektdurchführung $proj \in Proj$ als *gültig* für ein Vorgehensmodell *Vm*, wenn zu allen Methodenschrittdurchführungen ein entsprechender Methodenschritt existiert:

$$\forall msd \in proj : \exists ms \in Ms \wedge rep(msd) = ms.$$

Eine Projektdurchführung kann als *vollständig* bezeichnet werden, wenn zu jedem dem Vorgehensmodell zugeordneten nicht hierarchischen Methodenschritt mindestens eine Methodenschrittdurchführung existiert:

$$\forall ms \in Ms \wedge V(ms) = \emptyset : \exists msd \in proj : rep(msd) = ms.$$

5.6 Zusammenfassung

In diesem Kapitel wurden Methoden formal spezifiziert, sowie die Integration von Methoden in Vorgehensmodelle informell eingeführt und danach ebenfalls formal spezifiziert. Ausgehend von der Spezifikation eines formalen Methoden-Metamodells wurde gezeigt, wie Vorgehensmodelle über die Informationstypen ihrer Domänentheorie ihre Anforderungen an Methoden definieren und wie die Informationstypen der Methoden diese Anforderungen erfüllen können. Des Weiteren wurden typische Probleme wie Inkonsistenzen und Redundanzen diskutiert, die bei einer Methodenintegration auftreten können und gezeigt, wie diese zum Teil vermieden werden können. Eine Formalisierung der Methodenintegration mit Beschreibung einer auf Methodendurchführungen basierenden Semantik für Vorgehensmodelle schließen das Kapitel ab. Die Formalisierung von Methoden und Methodenintegration dienen einerseits zum Verständnis der Problemstellung, andererseits bilden sie die Grundlage für die Entwicklung der Erweiterung zu Integrations-Metamodellen im folgenden Kapitel.

Kapitel 6

Metamodell der Methodenintegration

In diesem Kapitel wird ein Metamodell eingeführt, das die Entwicklung und Verwaltung von Methoden sowie ihre Integration in Vorgehensmodelle unterstützt. Das Metamodell besteht aus zwei Teilen, einem Metamodell für Methodenbibliotheken sowie einem Metamodell der Methodenzuordnung zum Vorgehensmodell. Das Metamodell der Methodenbibliotheken dient als Framework zur Entwicklung und Verwaltung von Methoden der Softwareentwicklung, das Metamodell der Methodenzuordnung modelliert die Schnittstelle zwischen Vorgehensmodell und Methodenbibliothek und unterstützt die dynamische Zuordnung und projektspezifische Auswahl der Methoden im Vorgehensmodell.

Das Kapitel ist wie folgt aufgebaut: in Abschnitt 6.1 werden Ziele und Grenzen des hier vorgestellten Metamodells diskutiert. Im Weiteren wird das Metamodell der Methodenbibliotheken Schritt für Schritt aufgebaut. Ausgangspunkt ist das Metamodell einer Methode in Abschnitt 6.2. Das Methoden-Metamodell wird in Abschnitt 6.3 um ein Metamodell methodenübergreifender Beziehungen erweitert. Abschließend wird in Abschnitt 6.4 die Einbettung des Methoden-Metamodells in ein übergeordnetes Metamodell der Methodenbibliotheken vorgestellt. Die Konsistenzbedingungen zum Metamodell werden in Abschnitt 6.5 formalisiert. Das Metamodell der Methodenbibliotheken wird in Abschnitt 6.6 anhand verschiedener Beispielmethoden evaluiert. Abschnitt 6.7 stellt das allgemeine Konzept zur Modellierung der Schnittstelle zwischen Vorgehensmodell und Methodenbibliothek vor. Die konkrete Ausgestaltung der Schnittstelle ist dabei von dem jeweils gewählten Vorgehens-Metamodell abhängig. Abschließend wird in Abschnitt 6.8 der hier entwickelte Ansatz zu ähnlichen Ansätzen in der Literatur verglichen und abgegrenzt. Abschnitt 6.9 fasst die Ergebnisse des Kapitels zusammen.

6.1 Ziele und Grenzen

Das hier entwickelte Metamodell stützt sich auf das in Kapitel 5 entwickelte formale Metamodell, hat jedoch eine stärker anwendungsbezogene Ausrichtung. Im Fokus des Metamodells steht die werkzeugunterstützte Entwicklung von Methoden und ihre Verwaltung. So werden keine Kon-

sistenzbedingungen definiert, die sich auf Methodeninhalte, konkret auf die Domänentheorie der Methode und ihre Informationstypen beziehen. Die Überprüfung der inhaltlichen Konsistenz einer Methode sowie der Methodenintegration wird vielmehr als Aufgabe des Methodenautors gesehen. Der Vorteil des hier entwickelten Metamodell, insbesondere des Metamodells der Methodenbibliotheken, liegt vorrangig in der Vereinheitlichung der strukturellen Darstellung von Methoden, unabhängig von deren Inhalten. Konkret bietet das Metamodell folgende Vorteile:

- Bereitstellung eines einheitlichen Formats für die Methodenbeschreibung.
- Möglichkeit zur werkzeugunterstützten Manipulation der Methoden.

Ein einheitliches Format der Methodenbeschreibung bringt sowohl für Methodenautoren wie für Methodenanwender Vorteile. Für Autoren stellt das Methoden-Metamodell einen einheitlichen Entwicklungsrahmen dar. Der Rahmen definiert klare Richtlinien für die Methodenentwicklung und fördert durch eine einheitliche Methodenstruktur die Konzentration auf wesentliche Methodeninhalte. Insbesondere hilft eine einheitliche Methodenstruktur sowohl das Auftreten von Redundanzen als auch das Auftreten von methodischen Lücken in den Methodenbeschreibungen zu vermeiden.

Für die Anwender einer Methode unterstützt ein für alle Methoden einheitliches Format dagegen das Verständnis und erleichtert die Einarbeitung in neue Methoden. Der Aufbau einer Methode ist klar und bekannt. Die Möglichkeit zur werkzeugunterstützten Weiterverarbeitung von Methoden ist außerdem eine wesentliche Voraussetzung für die in dieser Arbeit angestrebte dynamische Integration von Methoden in Vorgehensmodelle.

6.2 Metamodell einer Methode

In diesem Abschnitt wird der Kern der Methodenbibliothek vorgestellt, das Metamodell einer Methode. Das hier entwickelte Metamodell orientiert sich an der in Abschnitt 4.6 eingeführten Methodenontologie und an den Basiskonzepten des formalen Metamodells in Abschnitt 4.3. Im Zentrum des Metamodells stehen *Arbeitsschritte*, *Artefakte*, *Vorgaben* und *Notationen*. Arbeitsschritte *nutzen* Artefakte, die ihnen Informationen zur Verfügung stellen. Sie benötigen diese Informationen um weitere Artefakte zu *produzieren*. Arbeitsschritte können eine hierarchische Struktur bilden, d.h. ein Arbeitsschritt wird durch eine Menge weiterer Arbeitsschritte verfeinert. Die *untergeordneten* Arbeitsschritte beschreiben jeweils einen spezifischen Ausschnitt des *übergeordneten* Arbeitsschritts im Detail.

Ein Artefakt repräsentiert innerhalb der Methode eine Menge von Informationen, die im Rahmen der Methodendurchführung *genutzt* oder *produziert* werden. Einem Artefakt ist das Attribut *extern* zugeordnet, welches kennzeichnet, ob ein Artefakt innerhalb der Methode produziert wird, oder ob das Artefakt mit den von ihm repräsentierten Informationen der Methode von außen zur Verfügung

gestellt werden muss. Artefakten kann eine *Notation* zur Dokumentation zugeordnet werden. Notationen sind typischerweise unabhängig von spezifischen Methoden und können mehrfach genutzt werden. Sie sind daher im Metamodell nicht als Teil einer Methode modelliert. Notationen können eine Hierarchie bilden. Typisches Beispiel einer hierarchischen Notation ist die UML. Die UML als referenzierbare Notation entspricht einer Menge von Diagrammtypen, wie beispielsweise Klassendiagramm, Sequenzdiagramm oder Automatendiagramm, die jeweils als eigenständige Notation einem Artefakt zugeordnet werden können.

Arbeitsschritte werden um *Vorgaben* ergänzt. Im Metamodell sind die Typen der zulässigen Vorgaben festgelegt. Möglich ist die Angabe von *Regeln*, *Erfahrungen* und *Theorien*. Erfahrungen kommen aus der Praxis und geben dem Anwender einer Methode Hinweise zur geeigneten Durchführung der Methode oder eines einzelnen Arbeitsschritts. Beispiele für *Erfahrungen* sind Checklisten, Heuristiken, Prinzipien, Best Practices etc. Das Konzept *Theorie* kann zur Ergänzung einer Methode um relevante Inhalte der zu Grunde liegenden Domänentheorie genutzt werden. *Regeln* sind aus Erfahrungen oder Theorien abgeleitete und aufbereitete Handlungsanweisungen für den Methodenanwender. Hinsichtlich der Zuordnung von Vorgaben werden hier im Gegensatz zum formalen Metamodell keine Einschränkungen gemacht. Eine Vorgabe kann einem oder mehreren Arbeitsschritten zugeordnet werden. Insbesondere bei hierarchischen Arbeitsschritten gelten die den übergeordneten Arbeitsschritten zugeordneten Vorgaben immer auch für die untergeordneten Arbeitsschritte. Vorgaben, die keinem Arbeitsschritt explizit zugeordnet sind, gelten für die gesamte Methode. Abbildung 6.1 stellt das Methoden-Metamodell dar. Notation ist dabei die Modellierungssprache aus Kapitel 2.3.

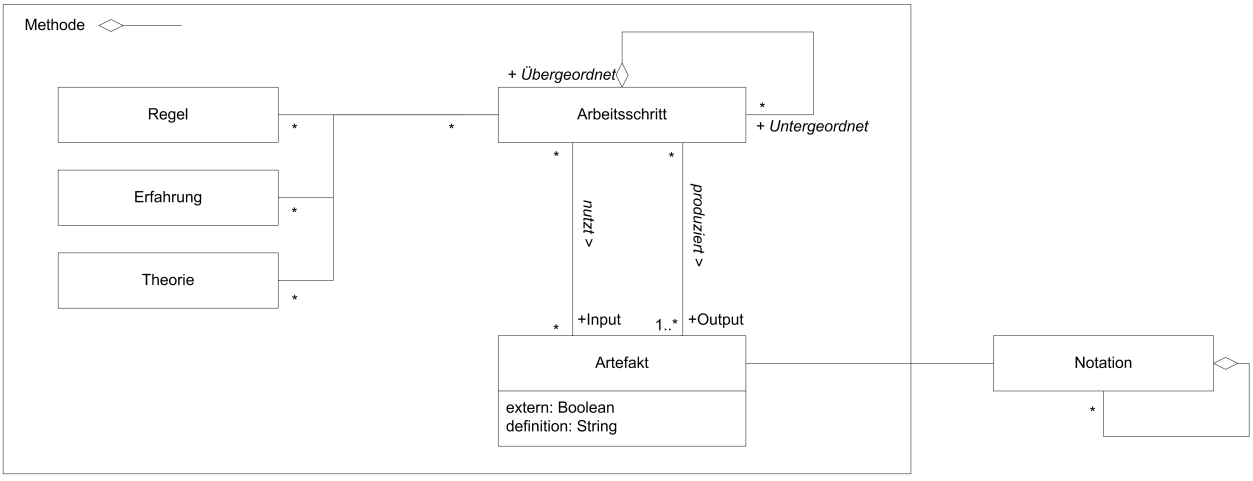

Abbildung 6.1: Metamodell einer Methode

6.3 Metamodell methodenübergreifender Beziehungen

Das Metamodell der methodenübergreifenden Beziehungen erweitert das Metamodell einer Methode um die Definition von Methodenhierarchien und Methodenspezialisierungen. Die Entwicklung von Methodenhierarchien erfolgt über die Zuordnung einer Methode zu einem Arbeitsschritt einer anderen Methode. Die von dem Arbeitsschritt genutzten und produzierten Artefakte definieren die Rahmenbedingungen für die hierarchisch untergeordnete Methode. Die genutzten Artefakte des Arbeitsschritt geben den Rahmen der verfügbaren Informationen vor, über die eine integrierte Methode verfügen kann; die von dem Arbeitsschritt produzierten Artefakte repräsentieren alle Informationen, die von der zugeordneten Methode mindestens zu erstellen sind. Das Metamodell unterstützt die hierarchische Unterordnung einer Methode durch eine 0..1:* Beziehung zwischen einem Arbeitsschritt und einer Methode. Eine Konsistenzbedingung stellt sicher, dass keine Zyklen innerhalb einer Methodenhierarchie entstehen können.

Als eine weitere methodenübergreifende Beziehung unterstützt das Metamodell die Methodenspezialisierung. Anhand einer *spezialisiert* Beziehung zwischen Methoden kann eine Methode als Erweiterung bzw. Spezialisierung einer anderen Methode der Methodenbibliothek gekennzeichnet werden. Die spezialisierte Methode definiert ausschließlich Erweiterungen zu ihrer Generalisierung. Sie ist nicht als eigenständige Methode anwendbar. Erlaubt sind im Rahmen der Spezialisierung ausschließlich Erweiterungen der ursprünglichen Methode. Konkret dürfen Arbeitsschritte zu einer Menge von Arbeitsschritten verfeinert werden. Artefaktmengen von Arbeitsschritten dürfen erweitert werden. Des Weiteren können Vorgaben und Notationen ergänzt werden. Änderungen an Elementen der bestehenden Methode sind dagegen nicht erlaubt. Wichtig ist hierbei, dass eine Spezialisierung selbst keine eigenständige Methode darstellt. Insbesondere gelten die Konsistenzbedingungen nur im Bezug auf eine Methode bzw. eine Methode mit Spezialisierung. Abbildung 6.2 stellt das Metamodell der methodenübergreifenden Beziehungen dar.

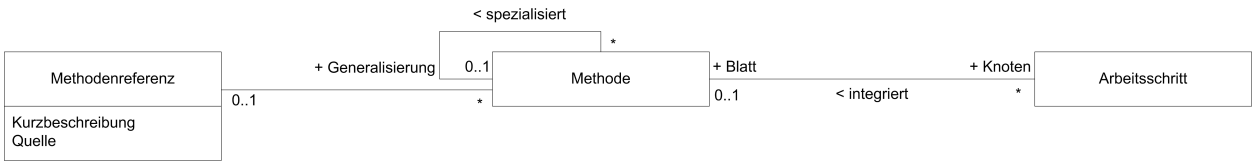

Abbildung 6.2: Metamodell methodenübergreifender Beziehungen

6.4 Metamodell der Bibliothek

Die Verwaltung der Methoden erfolgt in einer Methodenbibliothek. Innerhalb der Bibliothek werden die Methoden nach ihren *Anwendungsbereichen* gruppiert. Um Redundanzen bei den Methodenbeschreibungen zu vermeiden, sind in der Bibliothek alle Konzepte, die von mehreren Methoden

unabhängig voneinander genutzt werden können, ebenfalls auf Ebene der Anwendungsbereiche angesiedelt. Dies sind die Konzepte *Notation*, *Werkzeugreferenz* und *Methodenreferenz*.

Das Konzept einer Notation erlaubt die Beschreibung von Notationen oder die Referenzierung bereits dokumentierter und referenzierbarer Notationen. Notationen können hierarchisch aus weiteren Notationen zusammengesetzt sein. Methodenreferenzen erlauben die Modellierung von Referenzen auf Methoden, die nicht in der Bibliothek selbst vorliegen. Werkzeugreferenzen unterstützen schließlich die Angabe von Werkzeugtypen, die zur Durchführung einer Methode herangezogen werden können. Das Metamodell des Bibliotheksrahmen ist in Abbildung 6.3 dargestellt.

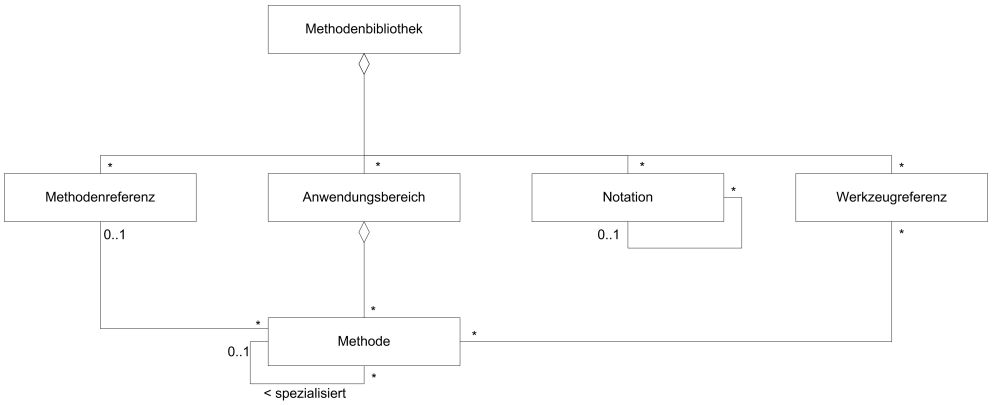

Abbildung 6.3: Metamodell der Methodenbibliothek

6.5 Konsistenzbedingungen

Neben den strukurell durch Entitäten, Beziehungen und Multiplizitäten definierten Vorgaben des Metamodells müssen für die Instanzen des Metamodells zusätzliche Konsistenzbedingungen gelten.

KB: Durchgängigkeit einer Methode

Das hier vorgestellte Metamodell kennt kein explizites Ablaufmodell für Methoden. Vielmehr ist das Ablaufmodell implizit über nutzt- und produziert- Beziehungen zwischen Arbeitsschritten und Artefakten vorgegeben. Dennoch muss die Durchgängigkeit der Methode sichergestellt sein. Diese Konsistenzbedingung fordert, dass jedes Artefakt der Methode, das nicht als extern gekennzeichnet ist, mindestens einmal innerhalb der Methode von einem Arbeitsschritt produziert wird.

```
context Methode inv:
    self.Artefakt→forall(af | af.extern <> 'true' implies
    self.Arbeitsschritt→exists(as | as.Output→includes(af))
```

KB: Konsistenz von Arbeitsschritthierarchien

Arbeitsschritte bilden eine echte Baumhierarchie. Innerhalb der Arbeitsschritthierarchie muss sichergestellt sein, dass keine Selbstverweise und keine Zyklen auftreten. Dementsprechend fordert diese Konsistenzbedingung, dass ein Arbeitsschritt weder von sich selbst noch von einem ihm in der Hierarchie übergeordneten Arbeitsschritt wieder als übergeordnet referenziert werden darf.

```
def: vater-as(as1:Arbeitsschritt,as2:Arbeitsschritt): boolean =
    if (as1.Übergeordnet = as2) then true
    else false endif

def: vater-as-rec(as1:Arbeitsschritt,as2:Arbeitsschritt): boolean =
    vater-as(as1,as2) or
    if (as1.Übergeordnet→notEmpty()) then
        vater-as-rec(as1.Übergeordnet,as2)
    endif

context Arbeitsschritt inv:
     Arbeitsschritt→forAll(not vater-as-rec(self,self))
```

KB: Konsistenz von Notationshierarchien

Notationen bilden ebenfalls eine Baumhierarchie. Ähnlich zu Arbeitsschritten muss auch hier gelten, dass innerhalb der Notationshierarchie keine Zyklen und keine Selbstverweise auftreten dürfen.

```
def: vater-n(n1:Notation,n2:Notation): boolean =
    if (n1.Übergeordnet = n2) then true
    else false endif

def: vater-n-rec(n1:Notation,n2:Notation): boolean =
    vater-n(n1,n2) or
    if (n1.Übergeordnet→notEmpty()) then
        vater-n-rec(n1.Übergeordnet,n2)
    endif

context Notation inv:
    Notation→forAll(not vater-n-rec(self,self))
```

KB: Konsistenz der Methodenhierarchien

Nicht nur Arbeitsschritte und Notationen, auch Methoden selbst können Hierarchien bilden. Innerhalb der Methodenhierarchie gilt ebenfalls, dass Zyklen und Selbstverweise nicht erlaubt sind. Die Konsistenzbedingung fordert, dass innerhalb einer Methodenhierarchie ein Arbeitsschritt einer

Methode nicht seine eigene Methode oder eine in der Integrationshierarchie übergeordnete Methode integrieren darf.

```
def: integriert (m1:Methode,m2:Methode): boolean =
    if (m1.Arbeitsschritt→exists(as | as.Blatt→includes(m2) then true
    else false endif

def: integriert-rec(m1:Methode,m2:Methode): boolean =
    integriert(m1,m2) or
    if(m1.Arbeitsschritt→exists(as | as.Methode→notEmpty())) then
        integriert-rec(as.Methode,m2)
    endif

context Methode inv:
    Methode→forAll(not integriert-rec(self,self))
```

KB: Konsistenz der Methodenspezialisierung

Methoden können andere Methoden spezialisieren (vgl. Abschnitt 6.3). Wie bei der Hierarchisierung muss jedoch sichergestellt sein, dass eine Methode direkt oder indirekt nicht sich selbst spezialisiert. Diese Konsistenzbedingung stellt sicher, dass eine Methode keine direkte oder indirekte *spezialisiert-*Beziehung zu sich selbst definiert.

```
def: spezialisiert (m1:Methode, m2:Methode)
    if (m1.Generalisierung = m2) then true
    else false endif

def: spezialisiert-rec (m1:Methode, m2:Methode)
    spezialisiert(m1,m2) or
    if (m1.Generalisierung.notEmpty()) then
        spezialisiert-rec(m1.Generalisierung,m2)
    endif

context Methode inv:
    Methode→forAll(not spezialisiert-rec(self,self))
```

6.6 Evaluierung am Beispiel

Das Metamodell der Methodenbibliothek erlaubt die Entwicklung von Methoden und ihre Verwaltung in einem übergeordneten Framework. In diesen Abschnitt wird das Metamodell anhand einiger Beispielmethoden evaluiert. Zur Evaluierung werden Instanzmodelle für Methoden, für me-

thodenübergreifende Beziehungen und für die Methodenbibliothek auf Basis des Metamodells entwickelt. Auf eine Belegung der Attribute wurde dabei aus Gründen der besseren Lesbarkeit verzichtet.

6.6.1 Instanzmodell einer Methode

In diesem Abschnitt wird ein Instanzmodell zum Methoden-Metamodell erstellt. Als Beispielmethode wird die Netzplantechnik (für eine detaillierte Beschreibung der Methode siehe Anhang A) verwendet. Ziel der Netzplantechnik ist die Erstellung eines Projektplans für ein Entwicklungsvorhaben. Abbildung 6.4 zeigt das Instanzmodell zur Netzplantechnik.

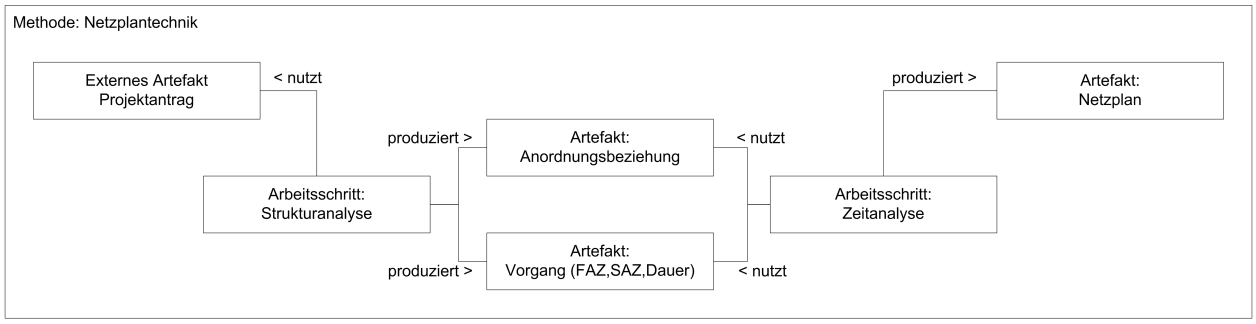

Abbildung 6.4: Instanzmodell der Netzplantechnik

Die Netzplantechnik geht grob in zwei Schritten vor. In einem ersten Schritt erfolgt die Strukturanalyse. Ausgehend von einem Projektantrag wird der Projektgegenstand ermittelt und geplant. Ziel der Strukturanalyse ist die Identifikation von Vorgängen und Anordnungsbeziehungen. Zu den Vorgängen werden die Werte zu FAZ, SAZ und Dauer ermittelt. Im Anschluss an die Strukturanalyse werden im Rahmen der Zeitanalyse die noch fehlenden Werte zu SEZ, FEZ und Gesamtpuffer berechnet. Grundlage der Berechnung sind die in der Strukturanalyse ermittelten Werte. Ergebnis ist ein Netzplan auf der Basis von Vorgängen und Anordnungsbeziehungen. Zu jedem Vorgang sind die entsprechenden Werte gesetzt.

6.6.2 Instanzmodell methodenübergreifender Beziehungen

Das Metamodell erlaubt zwei Formen von methodenübergreifenden Beziehungen, die Hierarchisierung - modelliert über eine *integriert*-Beziehung - und die Spezialisierung - modelliert über eine *spezialisiert*-Beziehung.

Hierarchisierung

Die Hierarchisierung von Methoden wird anhand der Methoden CoCoMo II und Function Point Analyse dargestellt. CoCoMo II wird in Projekten zur Ermittlung des Aufwands in Personenmonaten eingesetzt. Grundlage der Berechnung ist der jeweilige Schätzgegenstand. Anhand des Schätzgegenstands werden die Artefakte Kostenfaktor, Skalenfaktor und Größe ermittelt. Zur Berechnung

der Größe des Schätzgegenstands verweist die Methode CoCoMo II auf die Methode Function Point Analyse. Im Rahmen der Function Point Analyse wird die Größe eines Schätzgegenstand in der Einheit *unadjusted Function Points* angegeben. Das Ergebnis wird im Rahmen von CoCoMo II in die Größe *Source Lines of Code* umgerechnet und weiterverwendet.

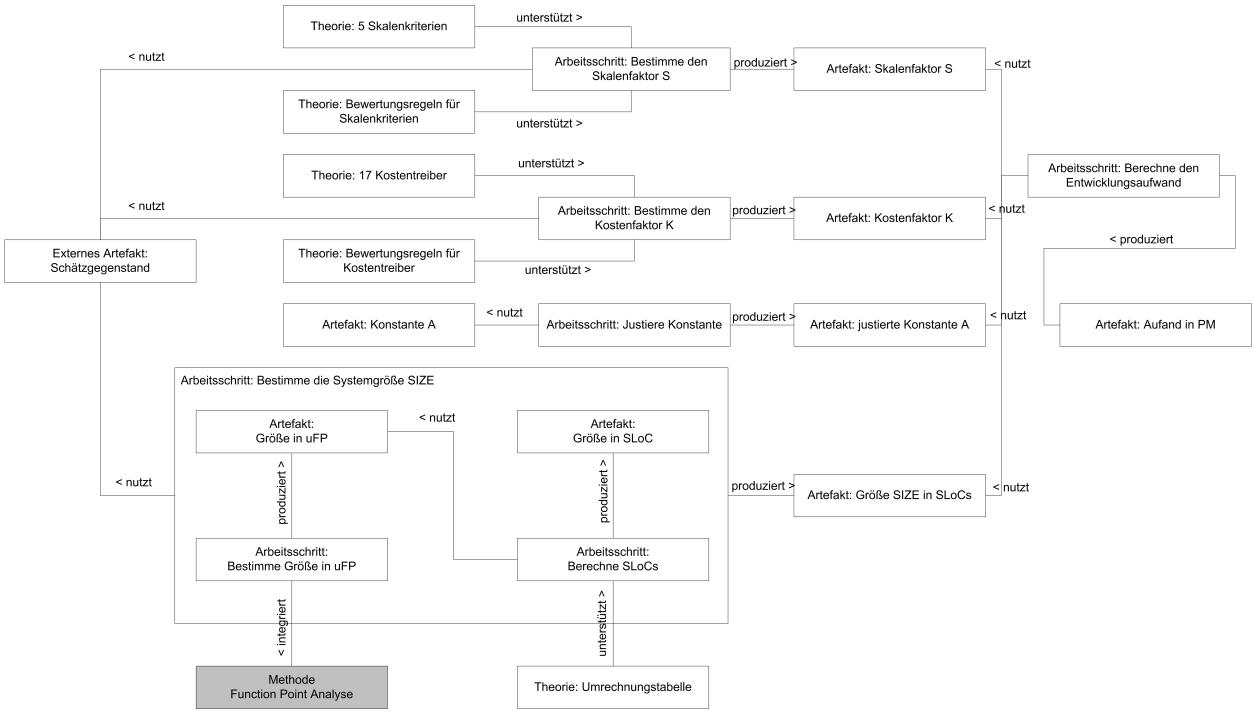

Abbildung 6.5: Instanzmodell der Methode CoCoMo II

Abbildung 6.5 zeigt das Instanzmodell der Methode CoCoMo II mit Integration der Methode Function Point Analyse. Auf eine explizite Modellierung der Function Point Analyse wird an dieser Stelle verzichtet. Es ist jedoch erforderlich, dass diese Methode ebenfalls in der Methodenbibliothek angelegt wurde.

Spezialisierung

Bei der Spezialisierung von Methoden handelt es sich um eine Form der Vererbung. Eine Methode, die eine andere Methode spezialisiert, erbt alle Konzepte, kann diese jedoch erweitern oder ausgestalten. Als Beispiel verwenden wir in diesem Fall die Critical Path Method als Spezialisierung der Netzplantechnik. Die Critical Path Method definiert eine Reihe von Erweiterungen zur Netzplantechnik. Diese betreffen die Ermittlung des kritischen Pfads, die Regeln zur Erarbeitung eines Netzplans und die zu verwendende Notation für den Netzplan.

Im Modell wird der Arbeitsschritt *Zeitanalyse* um eine Hierarchiestufe verfeinert. Als neue Arbeitsschritte mit den entsprechenden Artefakten werden die Vorwärtsrechnung und Rückwärtsrechnung eingeführt, sowie die Arbeitsschritte zur Ermittlung des kritischen Pfads und der Gesamtprojekt-

dauer. Des Weiteren wird die Methode um eine Menge von Regeln erweitert. Diese sind entsprechend [22]:

- *Regel 1:* Die Abhängigkeiten zwischen zwei Vorgängen wird durch einen Pfeil dargestellt. Die Pfeilrichtung gibt die Reihenfolge der Vorgänge an.
- *Regel 2:* Ein Vorgang kann einen oder mehrere Vorgänger haben. Ein Vorgang kann einen oder mehrere Nachfolger haben.
- *Regel 3:* Ein Netzplan darf keine Schleifen enthalten.
- *Regel 4:* Vom Projektanfang bis zum Projektende muss mindestens ein ununterbrochener Ablauf vorhanden sein.
- *Regel 5:* Der Vorgang am Projektanfang beginnt mit einem FAZ von Null (FEZ = FAZ + Dauer).
- *Regel 6:* Ein Vorgang kann erst beginnen, nachdem sein Vorgänger abgeschlossen ist.
- *Regel 7:* Besitzt ein Vorgang mehrere Vorgänger, so entspricht der FAZ des Vorgangs dem spätestens FEZ aller Vorgänger
- *Regel 8:* Der FEZ des Vorgangs am Projektende ist gleichzeitig der SEZ des Projekts (SAZ = SEZ - Dauer).
- *Regel 9:* Der SAZ eines Vorgangs ist gleichzeitig der SEZ aller Vorgänger des Vorgangs.
- *Regel 10:* Besitzen mehrere Vorgänge einen gemeinsamen Vorgänger, so entspricht der SEZ des gemeinsamen Vorgängers dem frühesten SAZ aller Nachfolger.

Diese Regeln sind als Ergänzungen der CPM zur Netzplantechnik zu sehen. Sie sind nicht explizit einem der Arbeitsschritte zugeordnet, gelten somit für die gesamte Methode. Als weitere Ergänzung zur Netzplantechnik ist dem Artefakt Netzplan in der Critical Path Method als konkrete Notation der Vorgangspfeilnetzplan zugeordnet.

Abbildung 6.6 stellt das Instanzmodell der Critical Path Method als Spezialisierung der Netzplantechnik dar. Elemente, die von der Methode Netzplantechnik bereits vorgegeben werden, sind grau hinterlegt. Der Bezug zwischen den ursprünglichen Elementen (z.B. Arbeitsschritt: Zeitanalyse in der Netzplantechnik) und ihren erweiterten Kopien (z.B. Arbeitsschritt: Zeitanalyse in der Critical Path Method) kann auf unterschiedliche Weise umgesetzt werden. Denkbar sind beispielsweise Kopien der Elemente, die der spezialisierten Methode zugeordnet werden. Der Bezug zu den Originalen erfolgt über den Namen. Ebenfalls denkbar ist die unmittelbare Zuordnung der Erweiterungen zu den Elementen der übergeordneten (spezialisierten) Methode.

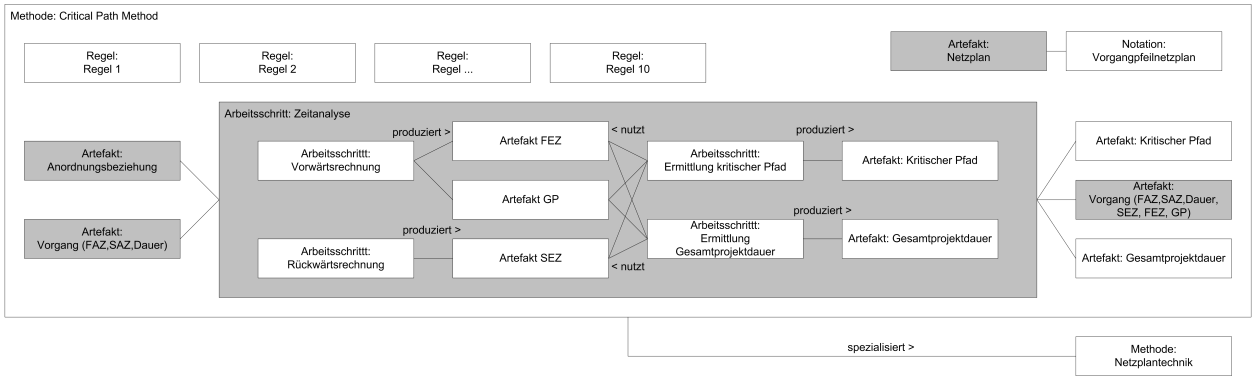

Abbildung 6.6: Critical Path Method als Spezialisierung der Netzplantechnik

6.6.3 Instanzmodell der Methodenbibliothek

Abschließend wird nun ein Instanzmodell der Methodenbibliothek vorgestellt, mit den Methoden Critical Path Method, Netzplantechnik, CoCoMo II und Function Point Analyse. Zur Verbesserung der Darstellung werden im Instanzmodell Knotenelemente eingeführt, die als Container für Modellelemente dienen, jedoch keine eigenständige Semantik aufweisen, wie beispielsweise Anwendungsbereiche, Notationen oder Werkzeugreferenzen. Abbbildung 6.7 stellt das Instanzmodell einer Methodenbibliothek dar.

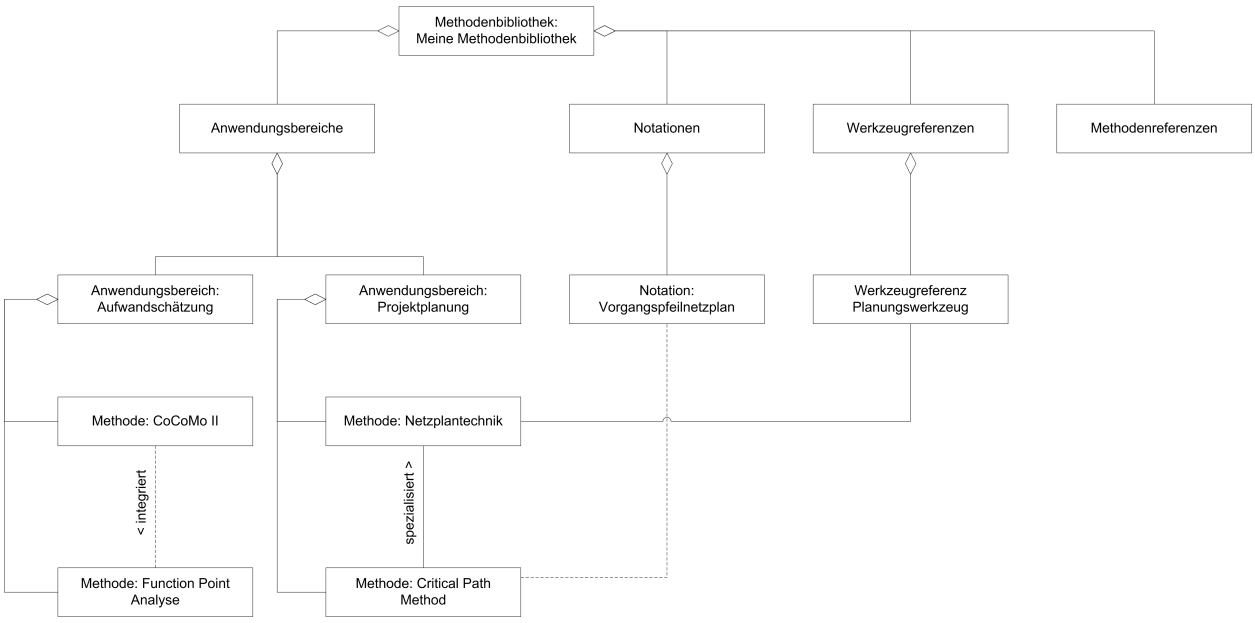

Abbildung 6.7: Instanzmodell der Methodenbibliothek

Die Methoden in der Bibliothek sind zwei Anwendungsbereichen zugeordnet, dem Anwendungsbereich Projektplanung und dem Anwendungsbereich Aufwandsschätzung. Das Instanzmodell stellt neben der Struktur der Methodenbibliothek auch die Beziehungen zwischen den Elementen dar. So nutzt die Critical Path Method als Notation für den Netzplan den Vorgangspfeilnetzplan. Die Netz-

plantechnik verweist auf die Werkzeugreferenz Planungswerkzeug. Diese Referenz wird im Rahmen der Spezialisierung an die Critical Path Method vererbt. Gestrichelte Linien deuten darauf hin, dass hier die Beziehung von untergeordneten Elementen, die in der Graphik nicht dargestellt sind, ausgeht. Beispielsweise wird die Beziehung *integriert* zwischen CoCoMo II und der Function Point Analyse zwischen einem Arbeitsschritt und einer Methode modelliert.

In diesem Abschnitt wurde ausgehend vom Metamodell einer Methode über das Metamodell methodenübergreifender Beziehungen abschließend das Metamodell der Methodenbibliothek entwickelt. Der folgende Abschnitt beschäftigt sich nun mit der Schnittstelle zwischen Vorgehens-Metamodell und Methoden-Metamodell.

6.7 Teilprozessmodell und Methodenmenge

Ziel des Metamodells ist die Unterstützung einer konsistenten und redundanzfreien Integration der Methoden in ein gegebenes Vorgehensmodell. Die Eigenschaften Konsistenz und Redundanzfreiheit beziehen sich dabei weitestgehend auf Methodeninhalte und nicht auf Methodenstrukturen. Sie sind daher nicht über Konsistenzbedingungen prüfbar. Eine vollständige Konsistenz und Redundanzfreiheit kann daher mit den begrenzten Mitteln eines Metamodells, wie es hier eingeführt wurde, nicht gewährleistet werden. Vielmehr ist es Aufgabe des Anwenders Konsistenz und Redundanzfreiheit bei der Methodenzuordnung im Integrationsmodell sicherzustellen. Um ihn dabei zu unterstützen, werden im Metamodell zwei neue Konzepte eingeführt, die gemeinsam die Schnittstelle zwischen Vorgehensmodell und Methodenbibliothek modellieren: die *Methodenmenge* und das *Teilprozessmodell*.

Eine *Methodenmenge* referenziert eine Menge von sich geeignet ergänzenden Methoden innerhalb der Methodenbibliothek. Dabei muss durch den Anwender sichergestellt werden, dass zwischen den Methoden innerhalb einer Methodenmenge keine Redundanzen hinsichtlich ihrer Artefakte existieren. Beispielsweise dürfen einer Methodenmenge nicht zwei Planungsmethoden mit zwei unterschiedlichen Artefakten *Projektplan* zugeordnet sein.

Ein *Teilprozessmodell* repräsentiert einer Methodenmenge gegenüber eine Menge von inhaltlich zusammengehörigen Produkten im Vorgehensmodell mit ihren Aktivitäten. Teilprozessmodelle dürfen sich hinsichtlich der von ihnen referenzierten Produkte nicht überschneiden, d.h. ein Produkt ist immer eindeutig einem Teilprozessmodell zugeordnet. Diese Eindeutigkeit gilt für Produkte, jedoch nicht für ihre Aktivitäten. So kann eine Aktivität mehrere Produkte erstellen, die unterschiedlichen Teilprozessmodellen zugeordnet sind. Die Aktivität wäre dementsprechend allen Teilprozessmodellen aller ihrer Produkte zugeordnet. Ausschlaggebend ist hier das jeweils gewählte Vorgehens-Metamodell mit seinen Beziehungen und Multiplizitäten.

Zur Integration der Methoden in ein Vorgehensmodell wird eine Methodenmenge jeweils einem Teilprozessmodell zugeordnet. Einem Teilprozessmodell können dabei mehrere Methodenmengen

alternativ zugeordnet werden. Die Artefakte aller Methoden einer Methodenmenge werden den Produkten des entsprechenden Teilprozessmodells zugeordnet. Innerhalb einer Methodenmenge kann so sichergestellt werden, dass bei der Zuordnung der Artefakte zu den Produkten keine inhaltlichen Überschneidungen entstehen können. Durch die Zuordnung mehrerer Methodenmengen zu einem Teilprozessmodell besteht außerdem die Möglichkeit, bei alternativ zugeordneten Methodenmengen eine davon projektspezifisch auszuwählen. Abbildung 6.8 stellt das Metamodell der Methodenzuordnung vor. Danach werden ausschließlich die Produkte dem Teilprozessmodell zugeordnet. Die entsprechenden Aktivitäten zum Teilprozessmodell können anhand der Assoziation zwischen Produkten und Aktivitäten berechnet werden.

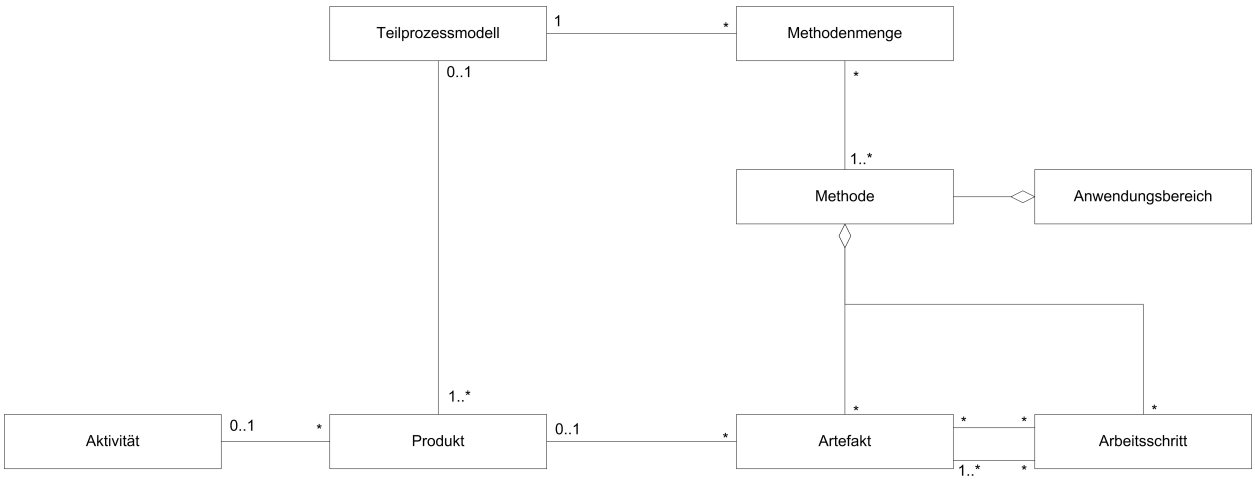

Abbildung 6.8: Allgemeines Metamodell zur Methodenzuordnung

Die Methodenintegration erfolgt über eine Zuordnung der Artefakte der Methoden einer Methodenmenge zu den Produkten eines Teilprozessmodells. Artefakte der Methoden einer Methodenmenge dürfen dabei ausschließlich Produkte des zugeordneten Teilprozessmodells referenzieren. Für Integrationsmodelle müssen zusätzlich folgende Bedingung zur Sicherstellung der konsistenten Methodenzuordnung gelten:

```
context Methodenmenge inv:
    self.Methode→forAll(m | m.Artefakt→forAll( af | self.Teilprozessmodell.Produkt→exists(p |
   p = af.Produkt)
```

Diese Konsistenzbedingung stellt sicher, dass einem Produkt ausschließlich Artefakte von Methoden zugeordnet werden, die zu einer dem Teilprozessmodell zugeordneten Methodenmenge gehören.

Teilprozessmodell und Methodenmenge bilden gemeinsam die Schnittstelle zwischen Vorgehensmodell und Methoden. Diese Konzepte sind dabei unabhängig von der Wahl eines konkreten Vorgehensmodells. Für die Modellierung der Schnittstelle sind jedoch Spezifika des jeweils gewählten Vorgehens-Metamodells zu berücksichtigen. Beispielsweise kennen manche Vorgehensmodelle eine

hierarchische Struktur ihrer Aktivitäten und Produkte, andere unterscheiden sich hinsichtlich der Multiplizitäten zwischen Produkten und Aktivitäten. Die Festlegung eines allgemein gültigen Metamodells der Methodenzuordnung ist daher nicht möglich. Der in diesem Abschnitt vorgestellte Ansatz beschreibt das allgemeine Konzept der Schnittstelle. Zur Anwendung ist es erforderlich, dieses Konzept auf die jeweilige Situation, d.h. das Metamodell des gewählten Vorgehensmodells, anzupassen. Eine entsprechende beispielhafte Anpassung mit Instanzmodellen wird in Kapitel 8 vorgestellt.

6.8 Vergleich zu alternativen Ansätzen

Das hier entwickelte Metamodell unterstützt die dynamische Integration und Auswahl von Methoden im Rahmen des Tailorings. Die Dynamik wird vor allem durch eine Auslagerung der Methodenzuordnung in Teilprozessmodelle und Methodenmengen erreicht. Aufgabe dieser Konzepte ist es einerseits die Integration selbst zu modellieren, andererseits über Konsistenzbedingungen die Korrektheit der Integration sicherzustellen. Im Folgenden werden weitere Ansätze zur Methodenintegration vorgestellt und der Ansatz des Integrations-Metamodells im Vergleich dazu bewertet.

6.8.1 Der OPEN Process

Das Akronym OPEN steht für Objekt-oriented Process Environment and Notation. OPEN [53] ist ein frei verfügbarer Prozessbaukasten zur Entwicklung organisationsspezifischer Vorgehensmodelle. Kern von OPEN ist das OPEN Process Framework (OPF) [44]. Elemente des Frameworks sind ein Metamodell zur Instanziierung organisationspezifischer Vorgehensmodelle, fertige Prozesskomponenten zur Zusammenstellung individueller Vorgehensmodelle nach dem Baukastenprinzip und Guidelines zur Entwicklung der Vorgehensmodelle. Idee ist die maßgeschneiderte Anpassung von Vorgehensmodellen durch Auswahl von Prozesskomponenten.

Abbildung 6.9 zeigt einen Ausschnitt aus dem OPF Metamodell. OPEN kennt auf Metamodellebene nicht das Konzept einer Methode wie es hier verstanden wird (Der Begriff *Method Component* ist hier als Prozesskomponente zu verstehen). Das Metamodell unterstützt jedoch zwei Methodenähnliche Konzepte: Sprachen und Techniken. Unter dem Konzept Sprache (Language) werden alle Formen von Notationen eingeordnet, wie sie zur Erstellung von Work Products Verwendung finden, beispielsweise formale Spezifikationssprachen, natürliche Sprache oder graphische Notationen. Eine Technik wird in OPEN als eine konkrete Umsetzung einer Work Unit definiert. Nach den von den Entwicklern des OPF genannten Beispielen (Protoyping, Incremental Development, Generalization) kann der Begriff der Technik am ehesten mit dem Konzept eines Prinzips gleichgesetzt werden. Im Metamodell können Techniken allen Formen von Work Units zugeordnet werden, Sprachen dienen zur Dokumentation von Work Products. Wie bei vielen aktivitätsorientierten Ansätzen kann somit

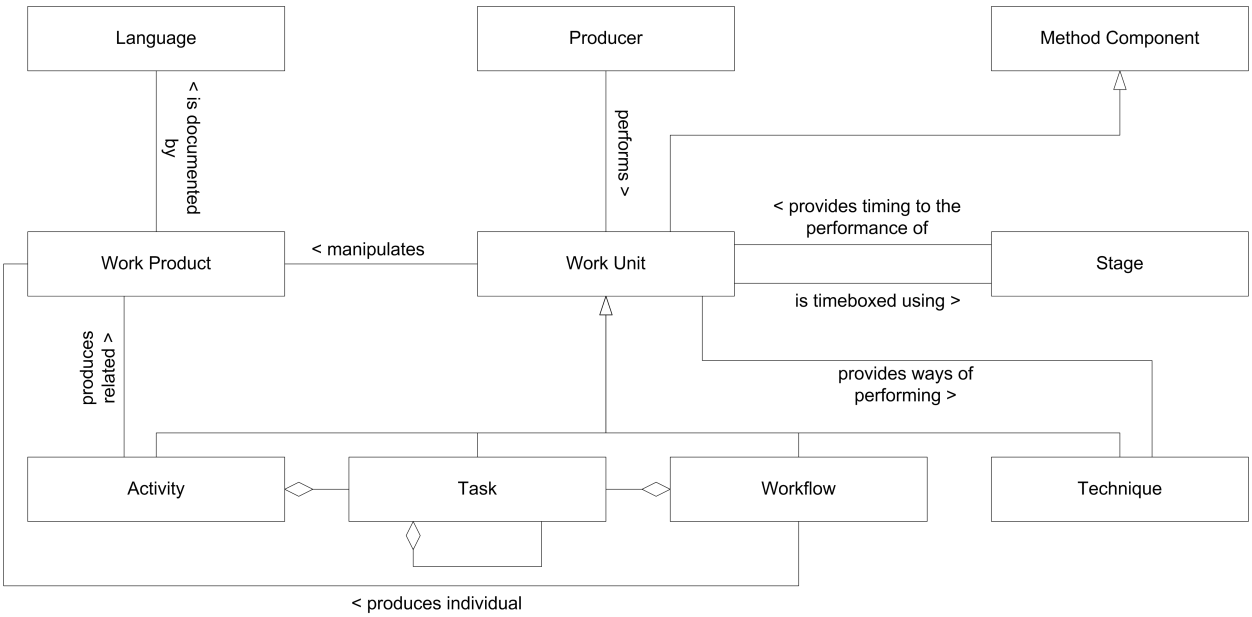

Abbildung 6.9: Open Process Metamodell (Quelle [44], eigene Darstellung)

auch bei OPEN die eigentliche Methodik im Vorgehensmodell als Teil von Work Units und Work Products gesehen werden, ergänzt um Sprachen und Techniken (Prinzipien).

Das OPF Metamodell kombiniert die Austauschbarkeit von einzelnen Prozesskomponenten mit einer Austauschbarkeit von Sprache und Prinzipien in den Vorgehensmodellen. Im Vergleich zum Integrations-Metamodell werden Methoden nicht als eigenständige Einheiten, sondern als Teil des Prozesses betrachtet. Kernproblem dieses Ansatzes ist die geringe Flexibilität bei der Anpassung der Vorgehensmodelle im Detail. Das Metamodell unterstützt zwar zur Anpassung die unterschiedliche Kombination von Prozesskomponenten, die Prozesskomponenten selbst sind jedoch mit ihrer Methodik fest vordefiniert. Das Integrations-Metamodell unterstützt dagegen eine echte Entkopplung von Vorgehensmodell und Methoden und erlaubt eine verbesserte Flexibilität bei der Anpassung. Insbesondere wird durch die Verwendung von Methodenmengen und Teilprozessmodellen die Zuordnung von Alternativmethoden mit werkzeugunterstützter Auswahl der geeigneten Methoden im Anpassungsprozess unterstützt.

6.8.2 Process Pattern

Der Process Pattern Ansatz von [50] verfolgt die Idee eines 'lebendigen' Software Entwicklungsprozesses. Prozessingenieure haben die Möglichkeiten, beliebig flexible Prozessmodelle zu entwickeln, die dynamisch auf sich ändernde Rahmenbedingungen in Projekten angepasst werden können. Das Metamodell selbst ist schlank gehalten. Zentrale Konzepte sind *Work Artefact Descriptions* und *Process Artefact Descriptions*. Diese bilden eine Einheit nach dem typischen aktivitätsorientierten Ansatz: eine Menge von *Work Artefacts* muss vorliegen, damit ein *Process Artefact* durchgeführt werden kann, das *Process Artefact* produziert neue *Work Artefacts*. Abbildung 6.10 zeigt die prin-

zipielle Idee des Process Pattern Metamodells. Bisher handelt es sich bei dem Process Pattern Ansatz um einen rein akademischen Vorschlag für den Entwurf von Vorgehensmodellen. Eine konkrete Implementierung existiert jedoch nicht.

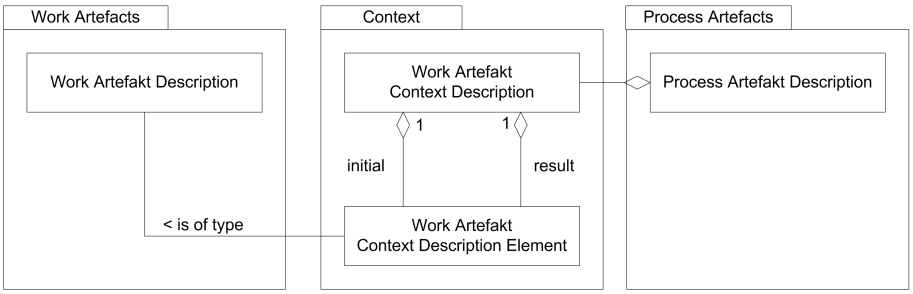

Abbildung 6.10: Process Pattern Metamodell (Quelle [50], eigene Darstellung)

Eine Besonderheit im Metamodell ist das Paket *Context*. Es bildet die Schnittstelle zwischen *Work Artefacts* und *Process Artefacts*. Das Schnittstellenelement *Work Artefact Context Description* fasst zu einem *Process Artefact* die Menge aller Eingabe und Ausgabe *Work Artefacts* zusammen. Dies erlaubt die flexible Austauschbarkeit der tatsächlichen Repräsentation des *Process Artefacts*. Es muss lediglich gewährleistet sein, dass die im Context durch Eingabe- und Ausgabeartefakte definierten Bedingungen erfüllt werden.

Entsprechend [50] kann ein Element vom Typ Process Artefact Description in zwei Ausprägungen vorliegen. So erlaubt der Process Pattern Ansatz einerseits das Vorliegen von 'leeren' Aktivitäten als Platzhalter für als bekannt und nicht mehr notwendigerweise zu dokumentierende Aktivitäten und andererseits den Einsatz konkreter Process Pattern. Process Pattern liefern methodische Beschreibungen zur Durchführung des Process Artefacts. Der Aufbau eines Process Pattern folgt dabei einem über Attribute vorgegebenen Muster, ähnlich zu den von Buschmann oder Gamma entwickelten Pattern Ansätzen (vgl. [47],[25]). Ein Beispiel für eine konkrete Umsetzung des Process Pattern Ansatzes in Form eines Prozesses zur komponentenbasierten Entwicklung findet sich in [15].

Der Process Pattern Ansatz kommt dem hier entwickelten Integrations-Metamodell von allen genannten Ansätzen am nächsten. Insbesondere die Idee der Entkopplung über eine explizite Schnittstelle findet auch im Integrations-Metamodell Verwendung. Durch die formalisierte Darstellung der Methoden geht der Ansatz hier jedoch einen Schritt weiter. Insbesondere erlaubt es die Zuordnung von Methoden nicht nur zu einzelnen Aktivitäten sondern auch zu aktivitätsübergreifenden Teilprozessen.

6.8.3 SPEM 2.0

In der Industrie konzentrieren sich aktuelle Aktivitäten im Bereich Vorgehensmodellierung vorrangig auf die Weiterentwicklung der SPEM Spezifikation. Ende 2004 veröffentlichte die OMG einen

Request for Proposal zur Einreichung von Vorschlägen für das Software Process Engineering Metamodell (SPEM) Version 2.0. Aktuell liegen unterschiedliche Vorschläge aus der Industrie vor: ein Vorschlag der Firmen Adaptive, ESI, Fujitsu, IBM und Softteam [89] sowie ein alternativer Vorschlag der Firmen Borland, Microsoft, Osellus und Sun Microsystems [90]. Der von IBM ausgearbeitete Vorschlag basiert im Wesentlichen auf den Konzepten der Unified Method Architecture (UMA) [131], einer Vorgehens-Metamodell Spezifikation für Vorgehensmodelle, die den Ideen und Prinzipien des Unified Process [69] folgt. Eine Referenzimplementierung der UMA steht im Eclipse Process Frameworks [119] zur Verfügung. Alle drei der genannten Ansätze verfolgen eine ähnliche Strategie hinsichtlich Methodenintegration und Anpassung. Im Folgenden wird diese stellvertretend am Beispiel der UMA vorgestellt. Es wird an dieser Stelle jedoch darauf hingewiesen, dass weder die Vorschläge zur SPEM 2.0 Spezifikation noch die UMA zum aktuellen Zeitpunkt als fertig gestellt betrachtet werden können.

Die UMA ist ein Metamodell zur Entwicklung modular aufgebauter Vorgehensmodelle. Kern des modularen Konzepts sind *PlugIns*. Ein PlugIn entspricht einem Container für Inhalts- und Prozessbeschreibungen von Vorgehensmodellen. Inhaltsbeschreibungen umfassen Rollen, Tasks und Work Products, Prozessbeschreibungen bestehen aus Iterationen, Meilensteinen und Aktivitäten. Mit Hilfe der Prozessbeschreibungen werden die Inhalte in eine geordnete Reihenfolge gebracht. Die Entwicklung und Anpassung von Vorgehensmodellen erfolgt über das PlugIn Konzept. Ausgehend von einem in PlugIns organisierten Basis-Vorgehensmodell (im Eclipse Process Framework steht hierzu der OpenUp Prozess zur Verfügung) werden Änderungen, Anpassungen und Erweiterungen über neue PlugIns im Vorgehensmodell eingeführt. Eine Konfiguration fasst jeweils eine Auswahl an PlugIns zu einer neuen Vorgehensmodell-Variante zusammen.

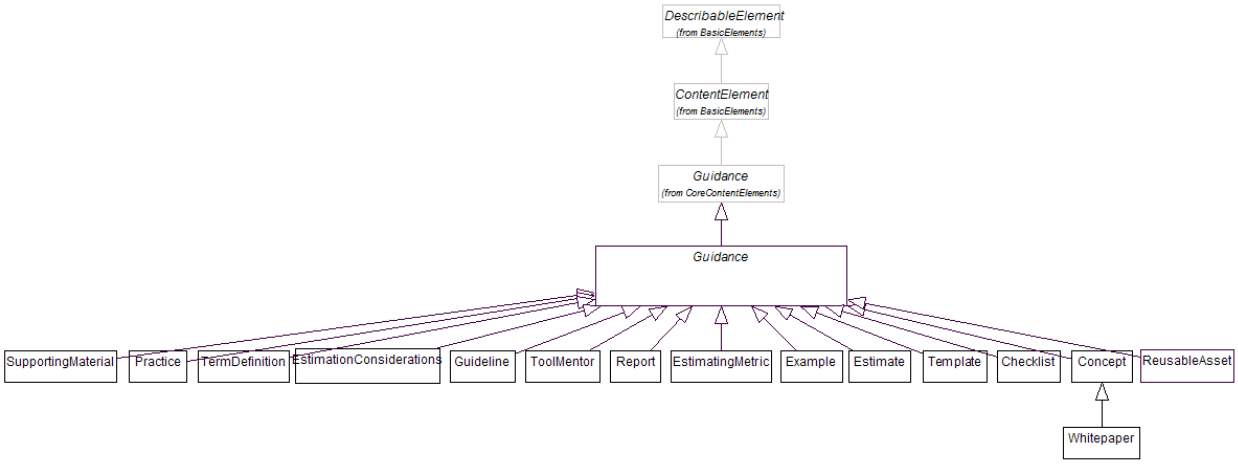

Abbildung 6.11: Guidance Taxonomie der UMA (Quelle [131])

Als aktivititätsorientierter Ansatz unterscheidet die UMA nicht explizit zwischen Methodik und Vorgehensmodell. Tasks und Work Products repräsentieren sowohl Vorgehensmodell Anteile wie auch methodisches Vorgehen. Zur Integration zusätzlicher Hilfskonzepte definiert die UMA das

abstrakte Konzept *Guidance*. Jedes von *Guidance* abgeleitete Element ist genau einem *GuidanceTyp* zugeordnet. Abbildung 6.11 zeigt anhand einer Taxonomie alle in der UMA vordefinierten Guidance Typen. Elemente vom Typ *Guidance* können beliebig allen Rollen, Work Products und Tasks zugeordnet werden. Die entsprechenden Strukturen im Metamodell sind (vereinfacht) in Abbildung 6.12 dargestellt.

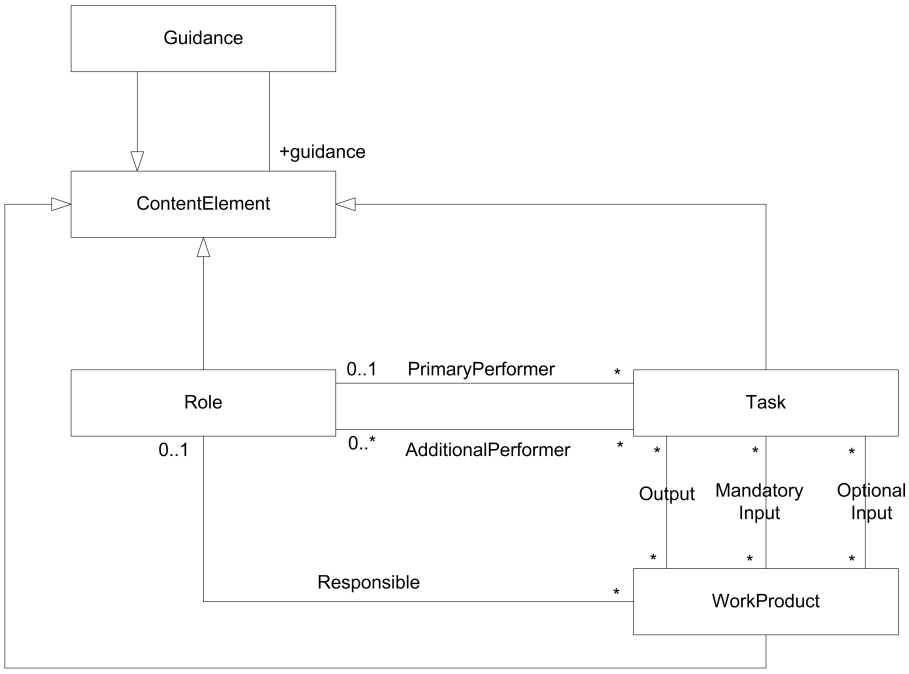

Abbildung 6.12: Guidance in der Unified Method Architecture (Quelle [131], eigene Darstellung)

Das Metamodell des SPEM erlaubt ähnlich zum OPF Metamodell über den aktivitätsorientierten Ansatz prinzipiell zwei Formen der Methodenintegration in das Vorgehensmodell: die Zuordnung von Hilfskonzepte zu Work Products und Tasks entsprechend dem Guidance Konzept, sowie die Zusammenstellung von Vorgehensmodellen über das Plug In Konzept. Durch den PlugIn Ansatz besteht die Möglichkeit, je nach Bedarf den Projekten unterschiedliche Vorgehensmodell-Konfigurationen mit entsprechender Methodik zur Verfügung zu stellen.

Der erste Ansatz führt zu einer losen Kopplung von Vorgehensmodell und Methodik. Dem Anwender stehen mit den Guidance Elementen eine Menge von methodischen Einzelelementen zur Verfügung, ein Leitfaden zur geordneten und übergreifenden Anwendung der Einzelelemente fehlt jedoch. Methodik und Prozessbeschreibung stehen nebeneinander ohne direkten Bezug. Der zweite Ansatz führt dagegen zu einer engen Kopplung von Vorgehensmodell und Methoden. Ein PlugIn umfasst immer Anteil des Vorgehensmodells und Anteile der Methodik. Ein Austausch der Methodik ist nur durch einen Austausch von Teilen des Vorgehensmodells möglich. Die Flexibilität bei der Methodenzuordnung, wie es das Integrations-Metamodell durch Entkopplung von Vorgehensmodell und Methoden erreicht, wird mit diesem Ansatz nicht erreicht.

6.9 Zusammenfassung

In diesem Kapitel wurde ein anwendungsnahes Metamodell zur Entwicklung und Verwaltung von Methoden sowie zur Integration der Methoden in ein Vorgehensmodell vorgestellt. Das Metamodell stützt sich auf Ergebnisse der Formalisierung von Methoden und Methodenintegration in Kapitel 5. Das Metamodell wurde in mehreren Schritten entwickelt. Ausgehend von einem Metamodell für einfache Methoden wurde das Metamodell erweitert zum Metamodell einer umfassenden Methodenbibliothek und ergänzt um ein allgemeines Metamodell der Schnittstelle zwischen Vorgehensmodell und Methodenbibliothek. Zur Modellierung der Schnittstelle wurden die Konzepte Teilprozessmodell und Methodenmenge eingeführt. Gemeinsam erlauben sie eine redundanzfreie und dynamisch auswählbare Zuordnung von Methoden zum Vorgehensmodell. Abschließend wurde der hier entwickelte Metamodell mit ähnlichen Ansätzen zur Methodenintegration verglichen und bewertet.

Kapitel 7

Lebenszyklusmodell für Integrationsmodelle

Nachdem in den vorangegangenen Kapiteln ein Konzept zur Erweiterung von Vorgehens-Metamodellen zu Integrations-Metamodellen vorgestellt wurde, beschäftigt sich dieses Kapitel mit den Lebenszyklen seiner Instanzen, den Integrationsmodellen. Ein Integrationsmodell entspricht einem Vorgehensmodell mit der Möglichkeit zur werkzeugunterstützten, projektspezifischen Methodenauswahl. Als Vorgehensmodell durchläuft ein Integrationsmodell, ähnlich zu einem Softwaresystem, einen iterativen Lebenszyklus, ausgehend von der Analyse bis hin zur Realisierung und Anwendung. In diesem Kapitel wird ein Lebenszyklusmodell für Integrationsmodelle vorgestellt, das auf einem allgemeinen Lebenszyklusmodell für Vorgehensmodelle beruht, zusätzlich jedoch Aspekte berücksichtigt, die für Integrationsmodelle relevant sind, wie Methodenentwicklung, Methodenanpassung, Methodenzuordnung und Methodenauswahl. Ziel des Kapitels ist es zu verdeutlichen, in welcher Form die Entscheidung für den Einsatz eines Integrationsmodells die Abläufe in IT-Organisationen beeinflusst.

Das Kapitel ist wie folgt aufgebaut: Abschnitt 7.1 führt ein Lebenszyklusmodell für Vorgehensmodellstandards ein, das unter anderem auch die Veröffentlichung und die Weiterentwicklung des Standards berücksichtigt. Ein Lebenszyklusmodell für organisationsspezifische Vorgehensmodelle wird in Abschnitt 7.2 vorgestellt. Abschnitt 7.3 zeigt die Synchronisationspunkte zwischen den Lebenszyklen auf. Abschnitt 7.4 diskutiert abschließend notwendige Ergänzungen des hier vorgestellten Lebenszyklusmodells im Hinblick auf die Anwendung von Integrationsmodelle. Abschnitt 7.5 fasst die Ergebnisse des Kapitels zusammen.

7.1 Lebenszyklusmodell für Vorgehensmodellstandards

Bekannte Lebenszyklusmodelle, wie beispielsweise das IDEAL Modell [42], unterscheiden in der Regel nicht zwischen dem Lebenszyklus eines Vorgehensmodellstandards und dem Lebenszyklus eines organisationsspezifischen Vorgehensmodells. Dementsprechend können diese Modelle nicht geeig-

net auf Besonderheiten und Unterschiede in den Lebenszyklen eingehen, wie beispielsweise unterschiedliche Arten der Anpassung, zeitliche und inhaltliche Abhängigkeiten zwischen dem Standard, seinen organisationsspezifischen Anpassungen sowie unterschiedliche Strategien zur zyklischen Verbesserung. In diesem Abschnitt wird ein Lebenszyklusmodell für Vorgehensmodelle vorgestellt, das explizit die Besonderheiten der Entwicklung und Pflege eines Standards berücksichtigt.

Ein Vorgehensmodellstandard ist ein generisches Vorgehensmodell, das von einer unabhängigen Organisation entwickelt und interessierten Anwendern zur Verfügung gestellt wird. Aufgabe der Organisation ist die Entwicklung, Veröffentlichung und Pflege des Vorgehensmodellstandards sowie die Initiierung und Weiterführung des Standardisierungsprozesses. Sie wendet den Standard jedoch in der Regel nicht an. Abbildung 7.1 stellt das Lebenzyklusmodell eines Vorgehensmodellstandards dar.

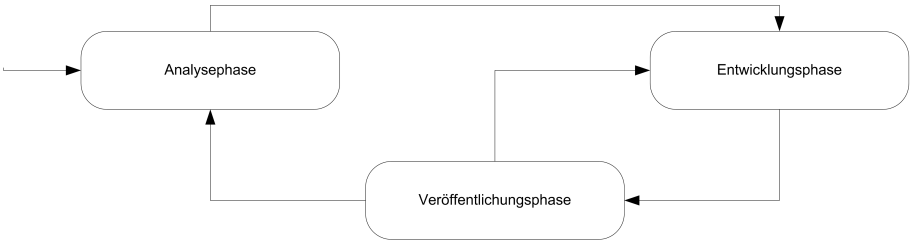

Abbildung 7.1: Lebenszyklus eines Vorgehensmodellstandards

Start im Lebenszyklus ist die *Analysephase*. Im Rahmen der Analyse werden die prinzipiellen Anforderungen an das zu entwickelnde Vorgehensmodell ermittelt. Bei der Analyse müssen beispielsweise folgende Fragestellungen berücksichtigt werden:

- Welche Projekttypen sollen durch das Vorgehensmodell abgedeckt werden (beispielsweise Softwareentwicklung, Hardwareentwicklung, SAP Einführung, Systembetreuung)?
- Welche Teilprozesse innerhalb der Projekttypen sollen durch das Vorgehensmodell abgedeckt werden (beispielsweise Projektmanagement, Systementwicklung, Auftraggebermanagement)?
- Soll das Vorgehensmodell explizit um Methoden erweitert werden oder methodenneutral bleiben?
- Welcher Philosophie soll das Metamodell des Vorgehensmodells folgen (beispielsweise aktivitätsorientiert, produktorientiert, agil)?
- Welchen Grad der Formalisierung soll das Vorgehensmodell unterstützen (informell, semiformal, formal) und welche Werkzeugunterstützung ist geplant?

Neben den konkreten Anforderungen bezüglich der Inhalte und des Metamodells werden mit der Anforderungsanalyse auch die allgemeinen Rahmenbedingungen für das neue Vorgehensmodell festgelegt wie Finanzrahmen, Rechte und Lizenzierungsmodell. Parallel zur Anforderungsanalyse wer-

den ein Werkzeugkonzept für die spätere Anwendung definiert und eine Entwicklungsumgebung aufgebaut.

In der *Entwicklungsphase* wird anhand der ermittelten Anforderungen das neue Vorgehensmodell konzipiert und erstellt. Das Metamodell wird entsprechend der gewählten Philosophie umgesetzt und der strukturelle Rahmen des neuen Vorgehensmodells entwickelt. Die Inhalte des Vorgehensmodells werden ausgearbeitet. Ziel ist es, die in der Anforderungsphase ermittelten Anforderungen in geeigneter Form umzusetzen.

Abschließend wird in der *Veröffentlichungsphase* des Vorgehensmodells geplant und umgesetzt. Das entwickelte Vorgehensmodell wird einem Reviewprozess unterzogen und eine geeignete Plattform zur Veröffentlichung aufgebaut und bereitgestellt. Ein Änderungsmanagement wird aufgesetzt und so der Grundstein für einen kontinuierlichen Pflege und Weiterentwicklungsprozess des Vorgehensmodellstandards aufgesetzt. Mit Ende der Veröffentlichungsphase steht den Anwendern ein Release des Vorgehensmodellstandards mit Werkzeugen, Dokumentation und Unterstützungskonzepten zur Verfügung. Eine Kopie des Vorgehensmodells kann nun von den entsprechenden Anwenderorganisationen genutzt werden. Parallel wird der Vorgehensmodellstandard in einem neuen Zyklus weiterentwickelt. Änderungswünsche der Anwenderorganisationen werden dabei entsprechend berücksichtigt.

7.2 Lebenszyklusmodell für organisationsspezifische Vorgehensmodelle

Anwenderorganisationen von Vorgehensmodellstandards sind beliebige Unternehmen, Behörden und Organisationen, die ihre Projekte nach den Vorgaben eines Vorgehensmodellstandards durchführen möchten. Eine Anwenderorganisation, die den Vorgehensmodellstandard einführen möchte, erhält eine Kopie des aktuellen Standards. Die Kopie durchläuft innerhalb der Organisation einen weiteren Anpassungs- und Anwendungszyklus (vgl. Abbildung 7.2).

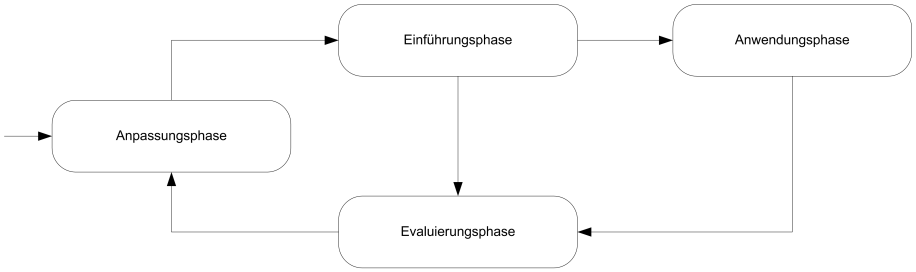

Abbildung 7.2: Lebenszyklus eines Organisationsspezifischen Vorgehensmodells

Erster Schritt einer Einführung ist die *Anpassung* des Vorgehensmodellstandards an die Rahmenbedingungen und Anforderungen der Anwenderorganisation. Mit der Anpassung wird das Vorgehensmodell auf die Bedürfnisse der Organisation hinsichtlich der Projektdurchführung zuge-

schnitten. Das resultierende organisationsspezifische Vorgehensmodell wird nun im Rahmen einer *Einführungsphase* explizit in der Organisation eingeführt. Im Rahmen der Einführungsphase wird eine Einführungsstrategie gewählt, die unter anderem festlegt, welche Projekte innerhalb der Organisation ab welchem Zeitpunkt nach dem neuen Vorgehensmodell vorzugehen haben. Des Weiteren werden in der Einführungsphase Schulungen und Informationsveranstaltungen vorbereitet und durchgeführt, sowie eine Wissensbasis und gegebenenfalls eine Messdatenbank eingerichtet. Das angepasste Vorgehensmodell wird anschließend unter Berücksichtigung der Einführungsstrategie in der Organisation eingeführt und steht den Projekten von nun an zur Anwendung zur Verfügung.

Das organisationsspezifisch angepasste Vorgehensmodell wird in der Anwendungsphase auf die Bedürfnisse der einzelnen Projekte angepasst und dient den Projekten als verbindlicher Leitfaden der Projektdurchführung. Parallel zur Anwendung wird das Vorgehensmodell und seine Anwendung im Rahmen der Evaluierungsphase bewertet. Erfolge, Probleme und Unklarheiten bei der Anwendung werden analysiert, Änderungswünsche der Anwender werden identifiziert und Anwenderfeedback eingeholt. Anhand der Ergebnisse wird gegebenenfalls ein Überarbeitungszyklus für das Vorgehensmodell initiiert und durchgeführt.

Das Zusammenspiel von Anpassungs-, Einführungs- und Analysephase wird im Bereich der Softwareentwicklung auch unter dem Schlagwort Software Process Improvement (SPI) subsumiert. Ziel des SPI ist die Verbesserung der Systemqualität durch eine gezielte Verbesserung des Entwicklungsprozesses (vgl. [98]). Hierzu wird systematisch der aktuelle Status eines Entwicklungsprozesses analysiert, entsprechende Verbesserungen werden konzipiert, eingeführt und schließlich die Wirkungsweise der Verbesserungen geprüft. Geeignete Modelle zur Unterstützung der Softwareprozessverbesserung wurden in Kapitel 3.7 vorgestellt. Neben Einführungsmodellen wie dem IDEAL Modell [42], können dies Qualitätsmanagementmodelle wie ISO 9000, Reifegradmodelle wie CMMI oder Assessmentmodelle wie SCAMPI sein. Jeder dieser Modelltypen unterstützt einen oder mehrere spezifische Aspekte im Pflege- und Weiterentwicklungsprozess.

Ein weiterer zentraler Baustein des SPI sind Metriken und Methoden zur Analyse des Ist-Zustands, zur Messung der Veränderungen/Verbesserungen sowie zur Auswertung der Daten. So ist beispielsweise die Goal Question Metric Methode ([10]) eine häufig eingesetzt Methode zur Prozessmessung und -verbesserung. QGM unterstützt den Aufbau eines Metrikprogramms in einer Organisation zur Messung der Prozessverbesserung. Die Methode geht hierzu in drei Stufen vor: Ermittlung der Ziele (Goals): Welches Ziel wurde mit einer bestimmten Maßnahme zur Prozessverbesserung verfolgt? Konkretisierung durch Fragen (Question): Anhand welcher Konzepte, Abläufe oder Daten kann die Erreichung der Ziele ermittelt werden? Ableitung von Metriken (Metrics): Mit Hilfe welcher konkreten Metriken kann gemessen werden, ob das Ziel erreicht wurden?

7.3 Synchronisation der Lebenszyklen

Die Lebenszyklen von Vorgehensmodellstandard und organisationsspezifischem Vorgehensmodell wurden bisher als weitgehend unabhängig voneinander dargestellt. Dies ist jedoch häufig nicht der Fall, insbesondere wenn das organisationsspezifische Vorgehensmodell auf einem Vorgehensmodellstandard beruht. Möchte beispielsweise eine Organisation von Weiterentwicklungen des Standards profitieren, braucht es einen Synchronisationspunkt zwischen den Lebenszyklen, an dem Änderungen des Vorgehensmodellstandards im organisationsspezifischen Vorgehensmodell übernommen werden können. Dies gilt insbesondere für Organisationen, die beispielsweise aus Zertifizierungsgründen dem Vorgehensmodellstandard folgen. Mit Ende der Veröffentlichungsphase im Zyklus eines Vorgehensmodellstandards steht ein neues Release zur Verfügung. Mit Erscheinen des Releases muss die Organisation eine Anpassung ihres Vorgehensmodells vornehmen und die Änderungen übernehmen. Abbildung 7.3 stellt den Zusammenhang auf der Zeitschiene dar.

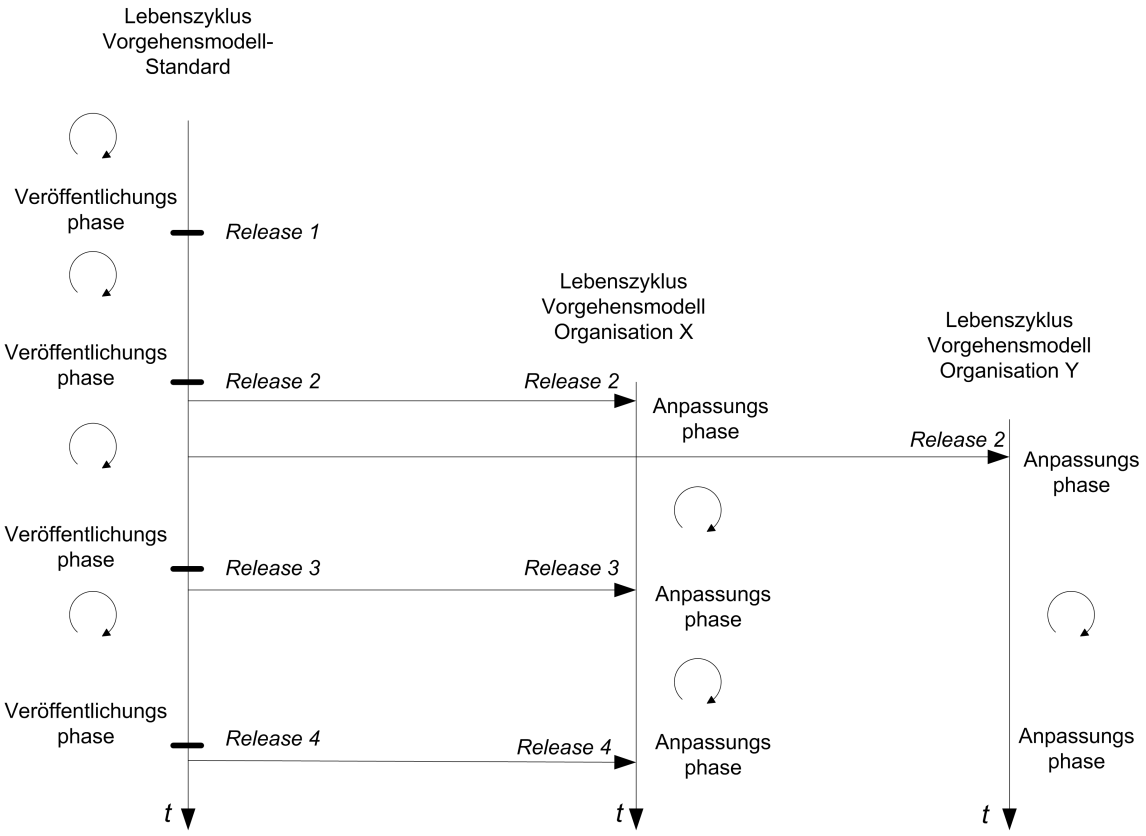

Abbildung 7.3: Synchronisation der Lebenszyklen

Im Beispiel hat Organisation X den Lebenzyklus ihres Vorgehensmodells auf den Lebenszyklus des Standards ausgerichtet. Erscheint ein neues Release, wird mit der Anpassungsphase ein neuer Zyklus für das organisationsspezifische Vorgehensmodell gestartet. Änderungen am Standard sowie Änderungen, die sich aus der Anwendung des Vorgehensmodells in der Organisation ergeben haben, werden in diesem Zyklus eingearbeitet.

Eine Synchronisation der Zyklen ist nicht für alle Organisationen von Belang. Im Beispiel übernimmt Organisation Y das aktuelle Release und entwickelt es weiter. Es besteht jedoch kein Interesse, Neuerungen des Standards in regelmäßigen Abständen zu übernehmen. Die Lebenszyklen von Standard und organisationsspezifischem Vorgehensmodell sind im Weiteren unabhängig voneinander.

7.4 Erweiterungen für Integrationsmodelle

Das hier vorgestellte allgemeine Lebenszyklusmodell für Vorgehensmodelle unterscheidet sich in seinem prinzipiellen Ablauf nicht von dem Lebenszyklusmodell für Integrationsmodelle. Unterschiede treten jedoch im Detail auf. Insbesondere der Anpassungsprozess im Lebenszyklus einer organisationsspezifischen Vorgehensmodellvariante unterscheidet sich vom Anpassungsprozess einer organisationsspezifischen Integrationsmodellvariante.

In diesem Abschnitt wird nun untersucht, welche Auswirkungen die Verwendung eines Integrationsmodells im Vergleich zu einem einfachen Vorgehensmodell auf den Anpassungsprozess im Lebenszyklus hat. In einem ersten Schritt wird der Anpassungsprozess eines einfachen Vorgehensmodells vorgestellt. Anschließend wird ein erweiterter Anpassungsprozess für Integrationsmodelle eingeführt, der zusätzlich den Prozess der Methodenentwicklung, Methodenanpassung, Methodenzuordnung und Methodenauswahl unterstützt.

7.4.1 Der Anpassungsprozess im Lebenszyklus

Der Anpassungsprozess eines organisationsspezifischen Vorgehensmodells zieht sich durch seinen gesamten Lebenszyklus. In der Anpassungsphase findet die organisationsspezifische Anpassung statt. Der Vorgehensmodellstandard wird inhaltlich an die Belange der Organisation angepasst. Diese Form der Anpassung findet genau einmal innerhalb eines Zyklus statt. Verantwortlich für die organisationsspezifische Anpassung ist der Prozessingenieur. In der Anwendungsphase wird das organisationsspezifisch angepasste Vorgehensmodell auf die Belange einzelner Projekte angepasst. Die projektspezifische Anpassung wird von den jeweiligen Projektleitern durchgeführt und erfolgt unabhängig für jedes Projekt von Neuem. Abbildung 7.4 stellt den einfachen Anpassungsprozess dar.

Organisationsspezifische und projektspezifische Anpassung verfolgen unterschiedliche Zielrichtungen und stellen dementsprechend unterschiedliche Anforderungen an die anzuwendende Anpassungsmethodik. Methoden zur organisationsspezifischen Anpassung haben das Ziel, ein Vorgehensmodell in seinen Inhalten so zu verändern, dass es die Prozesse in der Organisation so gut wie möglich abdeckt. Im Rahmen der organisationsspezifischen Anpassung wird das Vorgehensmodell durch Löschen, Ergänzen und Verändern/Verfeinern seiner Inhalte an den Kontext einer spezifischen Organisation angepasst. Ziel der projektspezifischen Anpassung ist es dagegen, über formale Verfah-

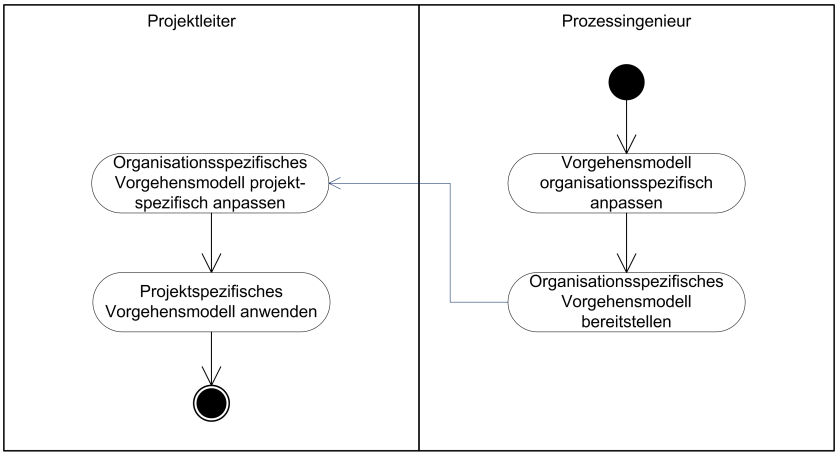

Abbildung 7.4: Einfacher Anpassungsprozess eines Vorgehensmodells

ren ein in sich konsistentes und geschlossenes Teilmodell im Vorgehensmodell zu identifizieren und zu extrahieren, das die Anforderungen eines konkreten Projekts an ein Vorgehensmodell abdeckt. Während sich die organisationsspezifische Anpassung stärker mit den Inhalten eines Vorgehensmodells beschäftigen, stehen bei der projektspezifischen Anpassung strukturelle Umwandlungen zur Bildung von Teilmodellen im Vordergrund. Das Vorgehen zur projektspezischen Anpassung ist dabei stark vom jeweiligen Vorgehensmodell und seinem Metamodell abhängig. So bietet jedes Vorgehensmodell sein individuelles Verfahren zur projektspezifischen Anpassung, häufig mit Werkzeugunterstützung, das sich nicht oder nur schwer auf andere Vorgehensmodelle übertragen lässt.

7.4.2 Erweiterter Anpassungsprozess für Integrationsmodelle

Der Anpassungsprozess für Integrationsmodelle entspricht einer Erweiterung des Anpassungsprozesses für einfache Vorgehensmodelle. Insbesondere sind Aspekte der Methodenentwicklung, Methodenintegration und projektspezifischen Methodenauswahl zu berücksichtigen. Der erweiterte Anpassungsprozess ist in Abbildung 7.5 dargestellt.

Parallel zur Anpassung des Vorgehensmodells wird im erweiterten Anpassungsprozess eine Methodenbibliothek angelegt oder eine existierende Methodenbibliothek an die in der Organisation verwendete Methodik angepasst. Vorgehensmodell und Methodenbibliothek sind zu diesem Zeitpunkt vollständig unabhängig. Erst in einem nächsten Schritt werden die Methoden über Methodenmengen den entsprechenden Teilprozessmodellen im Vorgehensmodell zugeordnet. Ergebnis der Zuordnung ist ein organisationsspezifisches Integrationsmodell, das nun zur Anwendung innerhalb der Organisation zur Verfügung gestellt werden kann. Das organisationsspezifisch angepasste Integrationsmodell wird von den Projektleitern zur Ableitung projektspezifischer Integrationsmodelle verwendet. Die projektspezifische Anpassung findet entsprechend der vom Vorgehensmodell zur Verfügung gestellten Methodik statt. Ergebnis ist ein projektspezifisches Vorgehensmodell. An-

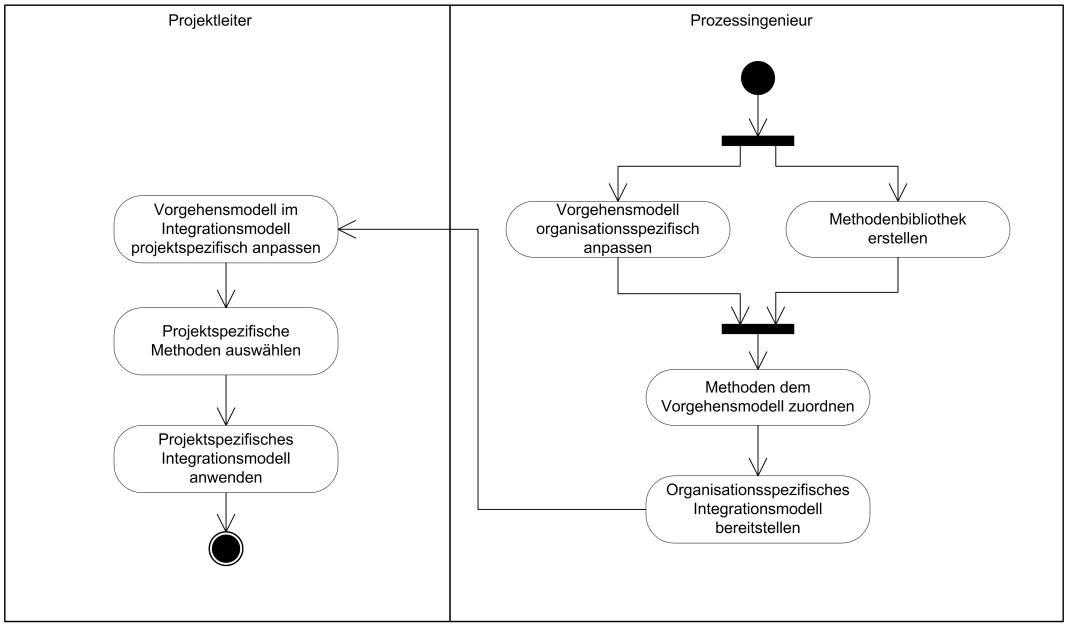

Abbildung 7.5: Erweiterter Anpassungsprozess für Integrationsmodell

schließend werden aus den im Rahmen der organisationsspezifischen Anpassung bereitgestellten Methodenmengen die für das jeweilige Projekt geeigneten Methoden ausgewählt. Ergebnis ist ein ideal auf ein Projekt angepasstes und um entsprechende Methoden erweitertes Integrationsmodell.

7.5 Zusammenfassung

In diesem Kapitel wurde ein Lebenszyklusmodell für Integrationsmodelle vorgestellt. Das Modell basiert auf einem allgemeinen Lebenszyklusmodell für Vorgehensmodelle, unterscheidet jedoch zwischen dem Lebenszyklus eines Vorgehensmodellstandard und dem Lebenszyklus eines organisationsspezifischen Vorgehensmodells. Diese Unterscheidung wird von üblichen Lebenszyklusmodellen nicht berücksichtigt, spielt jedoch im Detail durchaus eine Rolle. Insbesondere können Abhängigkeiten im Lebenszyklus organisationsspezifischer Anpassungen zu ihrem Standard berücksichtigt werden. Das resultierende Modell mit zwei Teilzyklen wurde abschließend auf die Anforderungen von Integrationsmodellen hin erweitert. Die Erweiterungen betreffen insbesondere den Anpassungsprozess des Vorgehensmodell, der sich durch den gesamten Lebenszyklus zieht.

Kapitel 8

Das V-Modell XT als Integrationsmodell

Das in Kapitel 6 eingeführte Metamodell der Integration kann prinzipiell mit jedem Vorgehens-Metamodell kombiniert werden, das dem produktorientierten Ansatz folgt (vgl. Abschnitt 3.6). In diesem Kapitel wird beispielhaft das Metamodell des V-Modells XT um das in dieser Arbeit entwickelte Metamodell der Methodenintegration zu einem vollständigen Integrations-Metamodell erweitert. Das resultierende Integration-Metamodell unterstützt die Ableitung von Integrationsmodellen deren Vorgehensmodelle Instanzen des V-Modell XT Metamodells sind.

Das Kapitel ist wie folgt aufgebaut: In Abschnitt 8.1 wird das V-Modell XT als Nachfolgestandard des V-Modells 97 mit seinen kennzeichnenden Eigenschaften im Überblick vorgestellt und es wird die Eignung des V-Modell Metamodell als Vorgehens-Metamodell im Integrations-Metamodell motiviert. In Abschnitt 8.2 wird das allgemeine Integrationskonzept aus Abschnitt 6.7 auf die strukturellen Vorgaben des V-Modell XT Metamodells angepasst und es werden die Konsistenzbedingungen entsprechend erweitert. Eine Besonderheit des V-Modell XT Metamodells ist das Metamodell zum Tailoring, das eine werkzeugunterstützte projektspezifische Anpassung der abgeleiteten Vorgehensmodelle erlaubt. Abschnitt 8.3 zeigt, wie das Tailoring-Metamodell um die werkzeugunterstützte Auswahl von Methodenmengen erweitert werden kann. Eine Evaluierung des Metamodells anhand von Beispielen erfolgt in Abschnitt 8.4. Abschnitt 8.5 diskutiert am Beispiel des V-Modells XT die Frage, welche inhaltlichen Eigenschaften das Vorgehensmodell in einem Integrationsmodell haben sollte, bzw. welche Eigenschaften Probleme bereiten. Eine beispielhafte Umsetzung und Anwendung von Integrationsmodellen auf Basis der V-Modell Technologien und Werkzeuge wird in Abschnitt 8.6 vorgestellt. Abschnitt 8.7 fasst das Kapitel zusammen.

8.1 Einführung und Motivation

Das V-Modell XT ist ein umfassendes Vorgehensmodell für die Durchführung von Entwicklungsprojekten, sowohl im Hardware- wie auch im Softwarebereich. Es ist aktueller Entwicklungsstandard

für IT-Vorhaben im Behördenbereich in Deutschland. Die Wurzeln des V-Modells XT liegen im 'ursprünglichen' V-Modell begründet, einer Erweiterung des Wasserfallmodells (vgl. Abschnitt 3.4). Während das Wasserfallmodell eine einfache Testphase am Ende des Entwicklungsprozesses vorsieht, unterstützt das V-Modell in seinen Phasen einen mehrstufigen Prozess zur Qualitätssicherung. Insbesondere unterscheidet das V-Modell zwischen Verifikationsphasen und Validierungsphasen in der Systementwicklung. Unter Verifikation wird die Überprüfung der Übereinstimmung zwischen einem Softwaresystem oder einem Teilsystem und seiner Spezifikation verstanden. Validierung betrachtet die Eignung eines Softwaresystems für seinen jeweiligen Anwendungszweck.

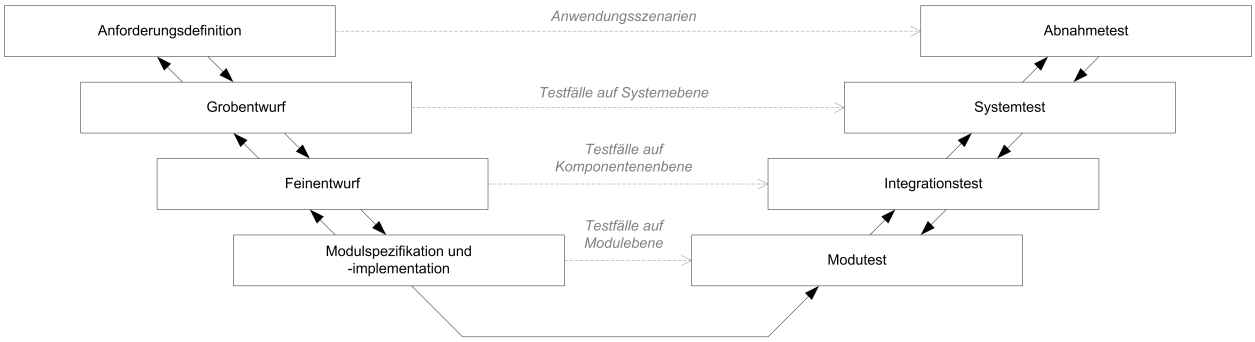

Abbildung 8.1: Das 'ursprüngliche' V-Modell' (Quelle: [8], eigene Darstellung)

Abbildung 8.1 stellt das V-Modell mit seinen kennzeichnenden Phasen und Phasenübergängen dar. Entsprechend dem Wasserfallmodell startet das V-Modell mit einer Phase zur Anforderungsdefinition, in der die Anforderungen an das zu entwickelnde System identifiziert und in strukturierter Form dokumentiert werden. Im Rahmen des Grob- und Feinentwurfs wird auf Basis der Anforderungen die Architektur des zukünftigen Systems festgelegt und verfeinert. Das Ergebnis dient als Spezifikation für die Implementierung der Module.

Die Testaktivitäten sind im V-Modell auf mehrere unabhängige Phasen verteilt, wobei jede Phase in direkter Korrelation zu einer Entwicklungsphase steht. Jede Testphase im V-Modell dient zur Verifikation/Validierung der in der entsprechenden Entwicklungsphase erstellten Spezifikationen bzw. der Anforderungen. Grundlage für die Verifikation/Validierung sind dann die aus den Spezifikationen jeweils abgeleiteten Testfälle.

Die resultierende Ablaufstruktur entspricht der für das V-Modell charakteristischen V-Form: Im Rahmen der *Anforderungsdefinition* werden Anwendungfälle erstellt, die die geforderte Funktionalität des zu entwickelnden Systems spezifizieren. Die Anwendungsfälle dienen als Grundlage für die Entwicklung der Testfälle zur Abnahme des Systems. Der *Abnahmetest* entspricht der Validierung des Systems. In ähnlicher Weide dienen die Ergebnisse der Phasen *Grobentwurf*, *Feinentwurf* und *Modulspezifikation* als Vorlage für die Entwicklung von Testfällen für *Systemtest*, *Integrationstest* und *Modultest*. In den Testphasen findet auf Basis der Testfälle jeweils die Verifikation zur Spezifikationen der entsprechenden Entwicklungsphase statt.

Die Idee zum V-Modell geht auf einen Vorschlag von Rook [99] zurück. Es bildet die Basis für das 1992 entwickelte und veröffentlichte V-Modell 92, den ersten umfassenden deutschen Entwicklungsstandard für die Durchführung von IT Vorhaben im Behördenbereich. Das V-Modell 92 setzte auf dem V-Modell als Ablaufmodell auf und erweiterte dieses um explizite Vorgaben zu Produkten und Aktivitäten.

Fünf Jahre später, 1997, wurde das V-Modell 92 durch das V-Modell 97 [39] abgelöst. Das V-Modell 97 war bereits eine überarbeitete, aktualisierte und hinsichtlich seiner Anwendbarkeit verbesserte Version des V-Modells 92. Es zeigte sich jedoch, dass auch das V-Modell 97 bei der Anwendung einige Defizite aufwies, wie beispielsweise:

- Das V-Modell 97 wurde rein dokumentenorientiert entwickelt. Eine Formalisierung der Modellstruktur über ein Metamodell und damit die Möglichkeit zur werkzeugunterstützten Bearbeitung und Manipulation war nie gegeben. Dies führte zu einer deutlich erschwerten Anwendung und zu geringer Akzeptanz bei den Anwendern. Um die Anwendbarkeit des Modells zu verbessern, investierten einige Unternehmen selbst in eine organisationsspezifische Implementierung des V-Modells. Eine entsprechende Implementierung für den Standard selbst gab es jedoch nicht.

- Das V-Modell 97 unterstützt einen zweistufigen Tailoringprozess auf der Basis von vordefinierten Projekttypen sowie der individuellen Auswahl von Produkten und Aktivitäten innerhalb des gewählten Projekttyps. Dieser Prozess war hilfreich, jedoch nicht ausreichend. Insbesondere gab es zu wenig methodische Unterstützung für den Tailoringprozess. Wurden beispielsweise zu viele oder die falschen Produkte und Aktivitäten aus dem Modell abgewählt, führte das zu einem unvollständigen und inkonsistenten getailorten Modell. Wurden dagegen für das jeweilige Projekt zu wenig Produkte und Aktivitäten abgewählt, führte das Ergebnis schnell zu einer Dokumentationsflut im Projekt. Ein Umstand, der bis heute für das V-Modell (zu Unrecht) als charakteristisch gilt.

- Das V-Modell 97 wurde 1997 fertig gestellt und seitdem nicht mehr weiterentwickelt. Änderungsanträge der Anwender wurden erfasst, jedoch nicht mehr im Modell übernommen. Technologische Neuentwicklungen und Forschungsergebnisse wurden nicht berücksichtigt. Die Ursache lag nicht zuletzt in der Repräsentation des V-Modells 97 als reines Textdokument, die eine konsistente und redundanzfreie Weiterentwicklung des Modells stark erschwerte, wenn nicht unmöglich machte.

Die genannten, aber auch weitere Gründe, wurden erkannt und führten 2002 zur Beauftragung einer vollständigen Überarbeitung des V-Modells 97. Ergebnis war das am 4.Februar 2005 veröffentlichte V-Modell XT. Mit dem V-Modell XT wurden zwar teilweise Konzepte des V-Modells 97 übernommen, es handelt sich jedoch strukturell und inhaltlich um ein vollständig neues Modell mit neuen Konzepten und neuer Philosophie.

Mit der Entscheidung zur Ablösung des V-Modells 97 wurden eine Reihe von Zielen verfolgt. Unter anderem sollten die oben genannten Defizite und Probleme bei der Anwendung des V-Modells 97 geeignet berücksichtigt und im neuen Modell, wo möglich, behoben werden. Ziel war es, so die Akzeptanz bei den Anwendern für das neue Modell entscheidend zu verbessern.

Die aus den Zielen abgeleiteten Anforderungen an das neue Modell betrafen sowohl inhaltliche wie auch strukturelle Aspekte. Das neue V-Modell sollte den Prozess zur organisations- und projektspezifische Anpassung geeignet, beispielsweise durch Werkzeuge, unterstützen und so die Anwendbarkeit des V-Modells grundlegend verbessern. Des Weiteren sollte das neue V-Modell besser auf unterschiedliche Projektgrößen skalierbar sein, um für Projekte unterschiedlicher Größe und Komplexität anwendbar zu sein. Diese und weitere Anforderungen waren richtungsweisend für die Entwicklung des V-Modells. Um den Anforderungen zu genügen, wurden insbesondere drei Designentscheidungen getroffen, die weitreichende Auswirkungen auf Inhalt und Struktur des V-Modells hatten:

- Dem V-Modell unterliegt ein formal definiertes Metamodell. Das Metamodell legt die Sprache zur Modellierung des V-Modells eindeutig fest. Das Metamodell gibt die zur Modellbildung zulässigen Konzepte, ihre Beziehungen, Multiplizitäten und Konsistenzbedingungen vor und erlaubt so die Entwicklung, Manipulation und Anpassung des Modells mit Hilfe von Werkzeugen.

- Das V-Modell ist produktorientiert (vgl. hierzu auch Abschnitt 3.6). Als produktorientiertes Vorgehensmodell stellt das V-Modell die in einem Projekt zu erstellenden Produkte in den Vordergrund. Aktivitäten als Beschreibungen einzelner Prozessschritte spielen dagegen nur eine untergeordnete Rolle. Der produktorientierte Ansatz führt in letzter Konsequenz zu einer Verringerung expliziter Abhängigkeiten im Vorgehensmodell und erlaubt dadurch eine höhere Flexibilität bei der Modellanpassung.

- Das V-Modell ist modular aufgebaut. Module im V-Modell sind so genannte Vorgehensbausteine. Ein Vorgehensbaustein umfasst alle Produkte, Aktivitäten und Rollen, die für einem spezifischen Teilprozess, beispielsweise Projektmanagement oder Systementwurf, in einem Projekt benötigt werden. Der modulare Ansatz erlaubt eine flexible Auswahl der für ein Projekt notwendigen Vorgehensbausteine und stellt so die Grundlage der werkzeugunterstützten, projektspezifischen Anpassung des V-Modells dar.

Die Umsetzung der hier genannten Eigenschaften manifestiert sich vorrangig im V-Modell Metamodell. Das V-Modell und sein Metamodell stehen jedoch selbst nicht im Fokus dieser Arbeit und werden hier nicht weiter betrachtet. Im Folgenden wird vielmehr beispielhaft am V-Modell Metamodell die Erweiterung eines Vorgehens-Metamodells zum Integrations-Metamodell gezeigt. Eine ausführliche Beschreibung des V-Modell Metamodells sowie des V-Modells selbst findet sich in Anhang B.

8.2 Einführung der Schnittstelle zu Methoden

In diesem Abschnitt wird die Erweiterung des V-Modell Metamodells zu einem vollständigen Integrations-Metamodell gezeigt. Dazu werden das V-Modell XT Metamodell und das Metamodell der Methodenbibliothek aus Kapitel 6 über eine Schnittstelle auf Basis der Konzepte *Teilprozessmodell* und *Methodenmenge* zusammengeführt (vgl. Abschnitt 6.7). Das resultierende Integrations-Metamodell erlaubt die Ableitung von Integrationsmodellen, d.h. von Vorgehensmodellen mit dynamisch auswählbaren Methoden.

Zur Erweiterung ist eine Anpassung der Schnittstelle auf die Gegebenheiten des V-Modell Metamodells erforderlich. Insbesondere auf die im Metamodell vorgegebene Struktur von Produktmodell und Aktivitätsmodell. Das V-Modell Metamodell fasst Produktmodell und Aktivitätsmodell in Vorgehensbausteinen zusammen. Abbildung 8.2 stellt das Metamodell eines Vorgehensbausteins entsprechend der V-Modell Dokumentation dar.

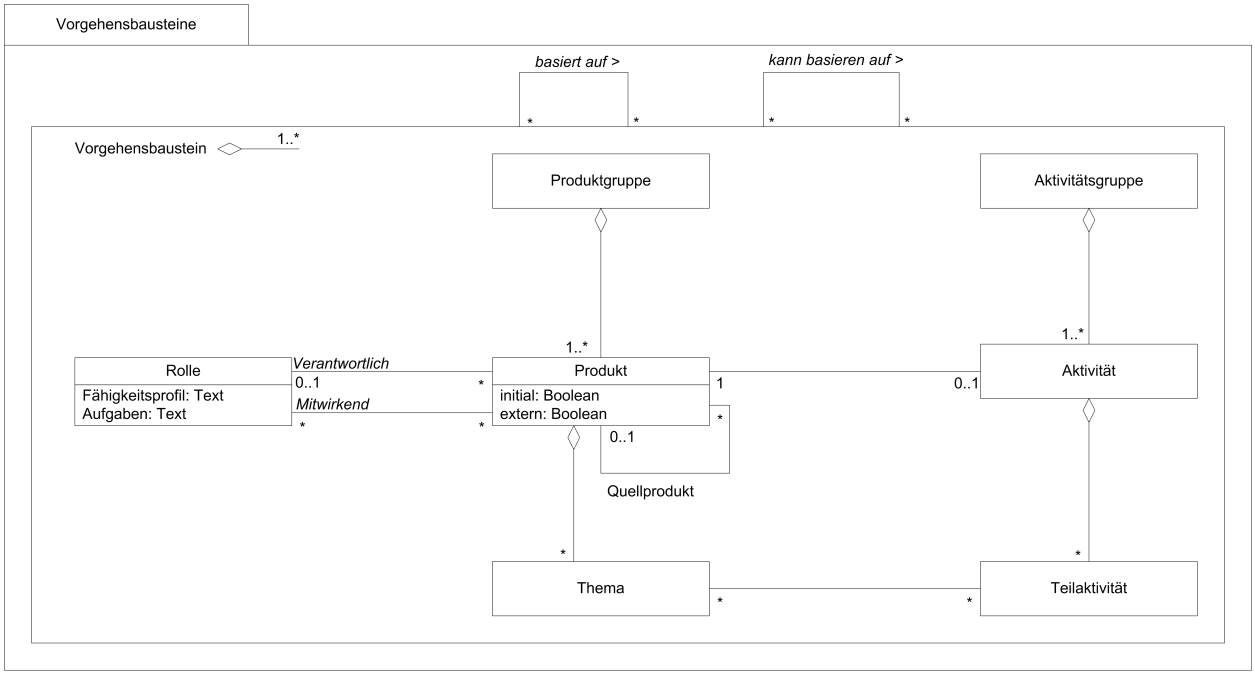

Abbildung 8.2: Metamodell der Vorgehensbausteine (Quelle [132], eigene Darstellung)

Im Zentrum steht das Metamodell des Produktmodells, repräsentiert durch die Elemente *Produktgruppe*, *Produkt* und *Thema*. Eine Produktgruppe fasst eine Menge von Produkten nach inhaltlichen Gesichtspunkten zusammen. Produktgruppen dienen ausschließlich zur Strukturierung des Produktmodells im V-Modell; sie haben keine Relevanz für die Projektdurchführung selbst. Ein Produkt deckt einen thematisch zusammengehörigen Bereich innerhalb der Produktgruppe ab. Es ist gleichzeitig ein im Vorgehensmodell definierter Ergebnistyp. Instanzen von Produkten sind konkrete Projektergebnisse. Ein Produkt kann aus einer Menge von Themen bestehen. Ein Thema deckt einen thematischen Teilbereich innerhalb eines Produkts ab.

Ein Produkt kann als initial, extern oder abhängig gekennzeichnet sein. Ein initiales Produkt wird genau einmal in einem Projekt instanziiert. Beispiele sind der Projektplan oder das Lastenheft. Diese Produkte braucht es immer und genau einmal in einem Projekt. Ein externes Produkt wird einem Projekt von außen zur Verfügung gestellt. Externe Produkte sind dadurch gekennzeichnet, dass ihnen keine verantwortliche Rolle und keine Aktivität zugeordnet wird. Ein Beispiel eines externen Produkts ist der Projektvorschlag. Ein Projektvorschlag initiiert ein V-Modell Projekt. Bei seiner Erstellung existiert das Projekt jedoch noch nicht. Abhängige Produkte sind Produkte, die als Ergebnis einer erzeugenden Abhängigkeit in einem Projekt erstellt werden.

Das Metamodell zum Aktivitätsmodell ist parallel zum Metamodell des Produktmodells aufgebaut. Elemente sind *Aktivitätsgruppen*, *Aktivitäten* und *Teilaktivitäten*. Bei der Entwicklung des V-Modells wurde auf eine Korrelierung von Produktgruppen und Aktivitätsgruppen geachtet. Diese Einschränkung wird im Metamodell nicht explizit gefordert, hat sich jedoch auf Grund der zu berücksichtigenden Abhängigkeitsstrukturen als sinnvoll erwiesen. Demnach gibt es im V-Modell zu jeder Produktgruppe genau eine Aktivitätsgruppe. Die Aktivitätsgruppe umfasst genau die Aktivitäten, die zur Erstellung der Produkte in der zugehörigen Produktgruppe erforderlich sind. Jede Aktivität beschreibt genau den Prozessschritt zur Erstellung ihres Produktes. Bei komplexen Aktivitäten kann eine verfeinerte Beschreibung über Teilaktivitäten sinnvoll sein. Eine Teilaktivität beschreibt einen Prozessschritt innerhalb einer Aktivität. Ergebnisse von Teilaktivitäten werden ausschließlich in den Themen zu dem von der Aktivität erstellten Produkt dokumentiert. Die Einhaltung dieser Einschränkung wird über eine entsprechende Konsistenzbedingung sichergestellt.

Verantwortlich für die Produkterstellung sind *Rollen*. Das Metamodell zum Rollenmodell umfasst genau ein Element, die Entität Rolle. Eine Rolle wird definiert durch einen Namen, ein Fähigkeitsprofil und eine Aufgabendefinition. Aufgabe einer Rolle ist die Erarbeitung der ihr zugeordneten Produkte. Das Metamodell unterscheidet bei der Zuordnung zwischen verantwortlichen Rollen und mitwirkenden Rollen. Für jedes Produkt im V-Modell (mit Ausnahme externer Produkte) gibt es eine für die Erstellung verantwortliche Rolle. An der Erstellung können jedoch weitere Rollen unterstützend mitwirken und dabei ihre Fähigkeiten bzw. ihr Wissen einbringen.

Beziehungen zwischen den Elementen können sich über Vorgehensbausteingrenzen hinweg ziehen. So kann ein Produkt einem anderen Vorgehensbaustein zugeordnet werden wie die Produktgruppe, zu der es gehört oder ein Thema, das ihm zugeordnet ist. Eine Rolle kann einem anderen Vorgehensbaustein zugeordnet werden, wie das Produkt für das sie verantwortlich ist. Dies führt zu einer Abhängigkeitsstruktur zwischen Vorgehensbausteinen. Modelliert werden diese Abhängigkeiten explizit über die Beziehungen *basiert auf* und *kann basieren auf* zwischen Vorgehensbausteinen. Diese Beziehungen sind wie folgt zu verstehen:

- Ein Vorgehensbaustein A *basiert auf* einem Vorgehensbaustein B, wenn Elemente aus dem Vorgehensbaustein A Beziehung zu Elementen aus dem Vorgehensbaustein B haben. Diese Information ist essentiell für das Tailoring. Wurde Vorgehensbaustein A im Tailoring aus-

gewählt, muss auch Vorgehensbaustein B mit ausgewählt werden um ein konsistentes Teilmodell als Tailoringergebnis zu gewährleisten. Eine *basiert auf* Beziehung entspricht einem mathematischen 'und' zwischen Vorgehensbausteinen.

- Die Beziehung *kann basieren auf* entspricht einer alternativen *basiert* auf Beziehung. Ein Vorgehensbaustein A kann auf Vorgehensbaustein B oder Vorgehensbaustein C basieren. Bei einer Auswahl von Vorgehensbaustein A muss demnach entweder Vorgehensbaustein B oder Vorgehensbaustein C mit ausgewählt werden um ein konsistentes Teilmodell als Tailoringergebnis zu erhalten. Eine *kann basieren auf* Beziehung entspricht einem mathematischen 'xor' zwischen Vorgehensbausteinen.

Abbildung 8.3 zeigt nun die Erweiterung des Metamodells für Vorgehensbausteine zur Unterstützung der Methodenintegration durch die Einführung der Konzepte Teilprozessmodell und Methodenmenge. Die Erweiterung unterstützt neben einer Zuordnung von Artefakten zu Produkten auch die Zuordnung von Artefakten zu einzelnen Themen. Durch Konsistenzbedingungen wird sichergestellt, dass ein Artefakt innerhalb einer Methodenmenge höchstens einmal zugeordnet wird.

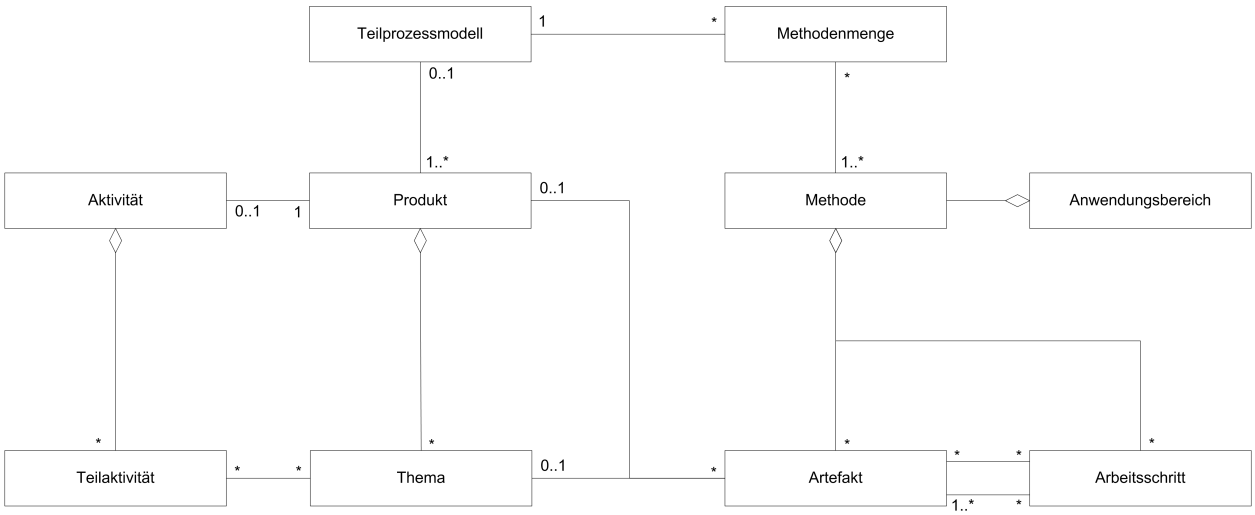

Abbildung 8.3: Metamodell der Integration angepasst auf das V-Modell Metamodell

Die Beziehungen zwischen Vorgehensmodell und Methodenbibliothek wurden bewusst minimal gehalten. Neben Teilprozessmodell und Methodenmengen existiert ausschließlich zwischen Artefakten und Produkten bzw. Themen eine Beziehung. Damit ist sichergestellt, dass Methoden und Vorgehensmodell weitgehend unabhängig voneinander bleiben. Alle weiteren Beziehungen, die für eine Integration notwendig sind, wie beispielsweise die Zuordnung von Arbeitsschritten zu Aktivitäten, können aus den vorhandenen Beziehungen berechnet werden. Als kritisch ist in diesem Fall die parallele Angabe von Teilaktivitäten im Vorgehensmodell und Arbeitsschritten der Methoden zu werten. Um hier Widersprüche zu vermeiden, sollten Aktivitäten, deren Produkte einem Teilprozessmodell zugeordnet sind, selbst keine Teilaktivitäten definieren.

Für Instanzen des erweiterten Metamodells müssen hinsichtlich der Integration die allgemeinen Bedingungen zur Sicherstellung der konsistenten Methodenzuordnung wie folgt auf die Gegebenheiten des V-Modell XT Metamodells angepasst bzw erweitert werden:

KB: Konsistenz der Methodenzuordnung

Die Methodenintegration erfolgt über eine Zuordnung der Artefakte der Methoden einer Methodenmenge zu den Produkten oder Themen eines Teilprozessmodells. Artefakte der Methoden einer Methodenmenge dürfen dabei ausschließlich Produkte und Themen des zugeordneten Teilprozessmodells referenzieren. Diese Konsistenzbedingung stellt sicher, dass den Produkten und Themen ausschließlich Artefakte von Methoden zugeordnet werden, die zu einer dem Teilprozessmodell zugeordneten Methodenmengen gehören.

```
context Methodenmenge inv:
    self.Methode→forAll(m | m.Artefakt→forAll( af | self.Teilprozessmodell.Aktivität→exists(ak |
    (ak.Produkt = af.Produkt) or (ak.Produkt.Thema→includes(af.Thema)))
```

KB: Konsistenz der Artefaktzuordnung

Das hier verwendete Vorgehens-Metamodell unterstützt die Verfeinerung von Produkten zu Themen. Die Artefakte einer Methode können entweder einem Produkt im Teilprozessmodell oder einem Thema zugeordnet werden. Diese Konsistenzbedingung stellt sicher, dass ein Artefakt im Rahmen der Zuordnung einer Methodenmenge zu einem Teilprozessmodell niemals beiden Konzepten gleichzeitig zugeordnet werden darf. Es ist jedoch erlaubt, dass ein Artefakt weder einem Produkt noch einem Thema zugeordnet ist.

```
context Methodenmenge inv:
    self.Methode→forAll(m | m.Artefakt→forAll( af | (af.Produkt.isEmpty() or af.Thema.isEmpty())))
```

8.3 Erweiterung des Tailoringmodells

Ein kennzeichnendes Merkmal des V-Modells ist das explizit vorgegebenen Tailoringmodell zur Unterstützung der projektspezifischen Anpassung. Das Tailoringmodell bringt Vorgehensbausteine und Projektdurchführungsstrategien (vgl. Anhang B.3) in einem konkreten Projektkontext, dem Projekttyp zusammen, und erlaubt eine projektspezifische Auswahl über Projektmerkmale. Durch seine Verankerung im Metamodell unterstützt das Tailoringmodell eine automatisierte, werkzeugunterstützte projektspezifische Anpassung des V-Modells (vgl. Tailoringmodell in Anhang B.4).

Ergebnis ist ein in sich vollständiges, konsistentes Vorgehensmodell, das in seinen Inhalten genau auf die Bedürfnisse des jeweiligen Projekts abgestimmt ist.

Die zentralen Tailoringkonzepte im Metamodell sind der Projekttyp und das Projektmerkmal. Ein Projekttyp referenziert eine Menge von Vorgehensbausteinen und eine Menge von Projektdurchführungsstrategien. Dabei kennzeichnet der Projekttyp eine Menge von Vorgehensbausteinen als verpflichtend andere als optional. Die vom Projekttyp referenzierten Projektdurchführungsstrategien stehen bei der projektspezifischen Anpassung prinzipiell alle zur Auswahl. Projektmerkmalswerte können jedoch teilweise auch diese Auswahl beeinflussen und die Verwendung einer spezifischen Projektdurchführungsstrategie erzwingen. Abbildung 8.4 zeigt das Metamodell zum Tailoring im Zusammenhang.

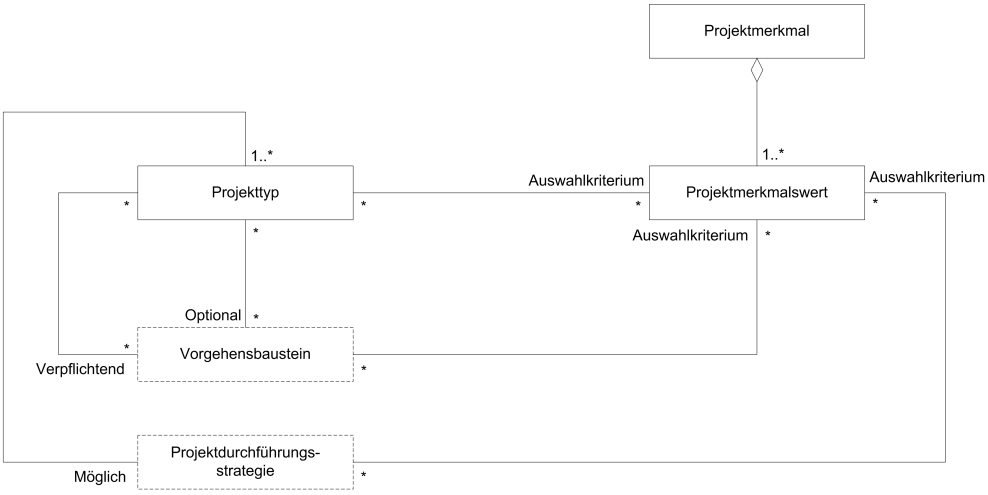

Abbildung 8.4: Metamodell zum Tailoringmodell (Quelle [132], eigene Darstellung)

Dieser Tailoringmechanismus kann nun einfach um Methodenmengen erweitert werden. Einem Teilprozessmodell können im Rahmen der organisationsspezifischen Anpassung mehrere Methodenmengen alternativ zugeordnet werden, d.h. die Artefakte ihrer Methoden sind den Produkten und Themen im Vorgehensmodell zugeordnet. Den Methodenmengen werden entsprechend dem Tailoringmodell im V-Modell Projektmerkmalswerte zugeordnet. Die Projektmerkmalswerte charakterisieren eine Projektsituation, in der die entsprechende Methodenmenge geeignet eingesetzt werden kann. Im Rahmen der projektspezifischen Anpassung kann nun anhand der Projektmerkmalswerte die am besten geeignete der zugeordneten Methodenmengen ausgewählt werden. Abbildung 8.5 zeigt die Integration der Methodenmengen in das Tailoringmodell des V-Modells.

Beispielsweise könnten in einem Integrationsmodell einem Teilprozessmodell *Systementwicklung* eine Methodenmenge zur *Softwareentwicklung* und eine Methodenmenge zur *Hardwareentwicklung* alternativ zur Verfügung stehen. Ein Projektmerkmalswert *Softwareentwicklung* für ein Projektmerkmal *Projektgegenstand* würde automatisch die *Softwareentwicklung* als geeignete Methodenmenge auswählen und *Hardwareentwicklung* nicht mehr zur Auswahl zur Verfügung stellen.

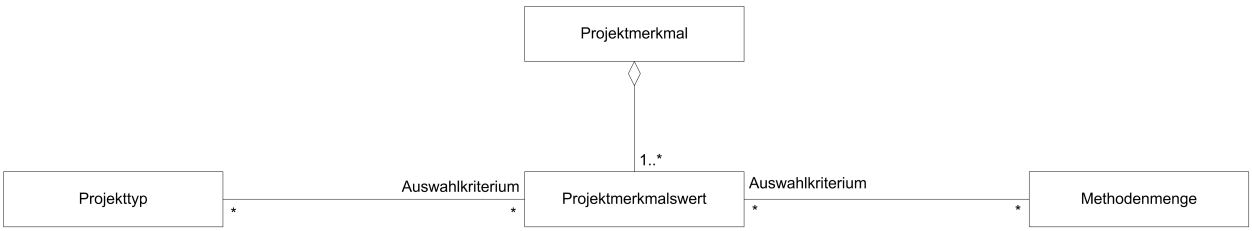

Abbildung 8.5: Integration von Methodenmengen in das Tailoringmodell

8.4 Evaluierung am Beispiel

In diesem Abschnitt wird das Metamodell der Methodenintegration anhand einer Integration der Methoden der in Abschnitt 6.6 entwickelten Beispielbibliothek in das V-Modell XT evaluiert. Der zur Integration betrachtete Ausschnitt aus dem V-Modell betrifft die Projektplanung und umfasst die Produkte *Projektplan* und *Schätzung* mit den Aktivitäten *Projekt planen* und *Schätzung durchführen* (vgl. Instanzmodell in Abbildung 8.6). Die Aktivitäten zu den Produkten und damit die Produkte selbst sind dem Teilprozessmodell *Projektplanung* zugeordnet.

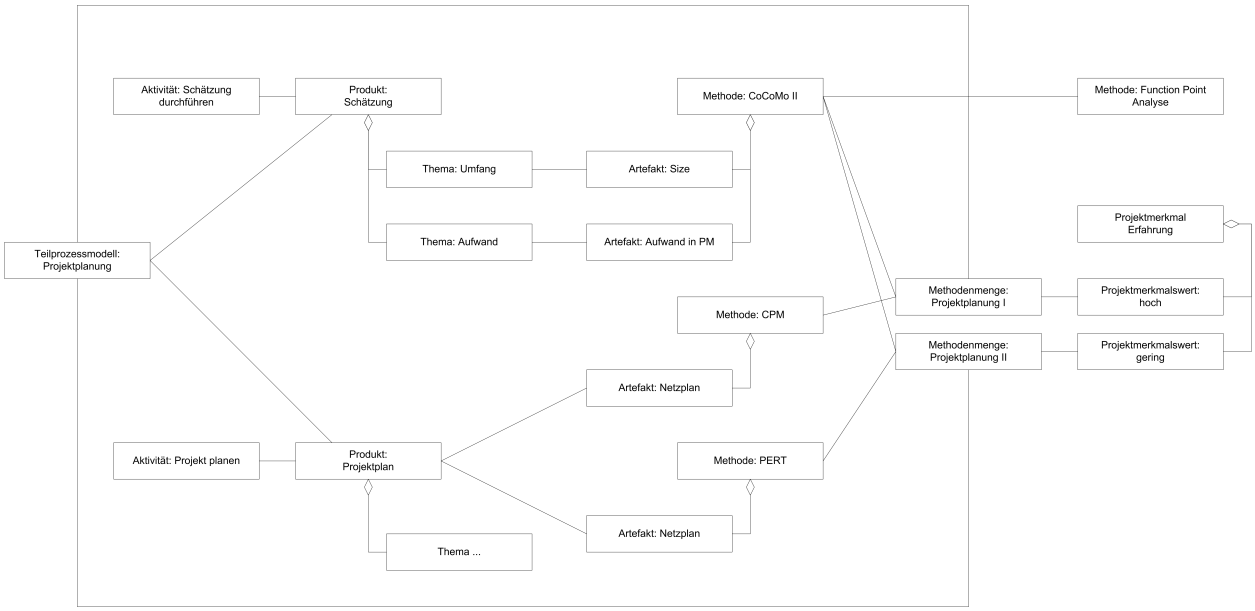

Abbildung 8.6: Integration der Methodenmengen

Zur Zuordnung stehen zwei alternative Methodenmengen zur Verfügung. Die Methodenmenge *Projektplanung I* umfasst die Critical Path Method als Planungsmethode und die Schätzmethode CoCoMo II. CoCoMo II integriert die Methode Function Point Analyse, diese ist jedoch, da keine ihrer Ergebnisse dem Vorgehensmodell zugeordnet werden, nicht Teil der Methodenmenge. Der Netzplan der CPM wird im Beispiel direkt dem Produkt *Projektplan* zugeordnet. Das Produkt *Projektplan* wird im V-Modell um eine Menge von Themen verfeinert, die neben der allgemeinen Projektplanung und Meilensteinplanung zusätzliche Einzelaspekte der Planung abdecken. Diese sind im Artefakt

Netzplan durch Angabe des *Projektantrags* (vgl. Abbildung 6.4) jedoch vollständig abgedeckt. Für das Produkt *Schätzung* ist dagegen eine unmittelbare Zuordnung der Artefakte *Size* und *Aufwand in PM* zu den Themen *Umfangschätzung* und *Aufwandsschätzung* möglich.

Als alternative Methodenmenge steht dem Teilprozessmodell die Methodenmenge *Projektplanung II* zur Zuordnung zur Verfügung. Diese Methodenmenge enthält im Beispiel die Planungsmethode PERT (vgl. Abschnitt A.3) und ebenfalls die Schätzmethode CoCoMo II. Hier wäre auch eine beliebige andere Schätzmethode denkbar. Die Methode PERT kann, wie die Critical Path Method, als Spezialisierung der Netzplantechnik in der Methodenbibliothek modelliert werden. Unterschiede zur CPM betreffen die Berechnung der Dauer eines Vorgangs, sowie die Zuordnung einer alternativen Notation zum Artefakt Netzplan. PERT wird im Gegensatz zur CPM eingesetzt, wenn die Dauer der Vorgänge nicht explizit angegeben werden kann.

Zur Unterscheidung der Methodenmengen wurde das Projektmerkmal *Erfahrung* mit den Werten *gering* und *hoch* im Vorgehensmodell ergänzt. Das Projektmerkmal kennzeichnet den Grad der Erfahrung, die hinsichtlich der zu ermittelnden Dauer von Vorgängen besteht (CPM) bzw. fehlt (PERT). Damit kann bei der projektspezifischen Anpassung eine entsprechende Auswahl der Methodenmenge getroffen werden kann.

8.5 Bewertung des V-Modells XT als Integrationsmodell

Der hier vorgestellte Ansatz lässt die Wahl des Vorgehens-Metamodells frei, solange es die Eigenschaft der Produktorientierung erfüllt. Auch das hier verwendete V-Modell XT Metamodell legt die Wahl des verwendeten Vorgehensmodells nicht fest. Jedes Vorgehensmodell, das gültige Instanz des V-Modell XT Metamodells ist, kann als Vorgehensmodell im Integrationsmodell verwendet werden. Dabei ist zu beachten, dass auch bei gleichem Metamodell sich nicht jedes Vorgehensmodell in gleicher Weise für eine Erweiterung zum Integrationsmodell eignet. So weist beispielsweise das V-Modell XT Eigenschaften auf, die bei einer Methodenintegration zu Problemen führen können. Diese Eigenschaften liegen unmittelbar im Modell, nicht im Metamodell begründet und werden im Folgenden näher erläutert. Sie gelten nicht unbedingt für andere Instanzen des Metamodells.

Wie in Abschnitt 5.2 gezeigt wurde, definieren Vorgehensmodelle über ihre Produkte Anforderungen an Methoden. Dabei unterscheiden sich Vorgehensmodelle hinsichtlich der Detaillierungstiefe, in der sie ihrer Anforderungen definieren. Ein sehr hoher Grad der Detaillierung - eine detaillierte Angabe von Informationstypen - kann zur Folge haben, dass nur sehr wenig Methoden den Anforderungen tatsächlich genügen können. Bleibt ein Vorgehensmodell dagegen allgemein in seinen Anforderungen, ist die Palette der verfügbaren Methoden sehr viel umfangreicher.

Das V-Modell XT definiert seine Anforderungen an Methoden mit unterschiedlichem Detaillierungsgrad. Im Bereich der querschnittlichen Aufgaben wie Qualitätssicherung oder Projektmanagement bleibt das V-Modell XT eher allgemein. Die methodische Ausgestaltung kann hier variabel erfol-

gen, dementsprechend steht eine Vielzahl potentieller Methoden zur Integration zur Verfügung. Es werden lediglich gewisse Grundvoraussetzungen von den Methoden gefordert. Beispielsweise fordert das Produkt *Schätzung* über seine Themen die Informationstypen *Umfangsschätzung* und *Aufwandsschätzung*. Eine geeignete Methode sollte Ausgabeartefakte mit entsprechenden Informationstypen für beide dieser Werte liefern. Dies trifft jedoch im Wesentlichen auf alle bekannten Schätzmethoden zu, so dass hier eine Methodenintegration keine Probleme bereitet. Kritischer ist dagegen der Bereich der Systementwicklung im V-Modell XT. Hier geht das V-Modell in seinen Anforderungen teilweise so stark in Details, dass eine Wahl geeigneter Methoden stark erschwert wird bzw. nicht mehr sinnvoll möglich ist. Dies wird im Folgenden an einem Beispiel illustriert:

Das V-Modell XT unterstützt eine komponentenorientierte Systementwicklung. Von der Idee her wird ein System in Form einer Komponentenhierarchie entworfen, wobei das V-Modell über strukturelle Produktabhängigkeiten Vorgaben für die Komponentenzerlegung macht. Die Komponentenhierarchie wird auf zwei Ebenen beschrieben. Auf Ebene des Systems in der *Systemarchitektur*, auf Ebene der Hardware oder Softwareeinheiten in einer *HW-* bzw. *SW-Architektur*. Schnittstellen und Funktionalität der identifizierten Komponenten werden pro Komponente in einer *Systemspezifikation* (HW-Spezifikation, SW-Spezifikation) konkretisiert. Die Spezifikationstiefe wird dabei dem Architekten überlassen. Abbildung 8.7 fasst die Idee des Systementwurfs im V-Modell zusammen.

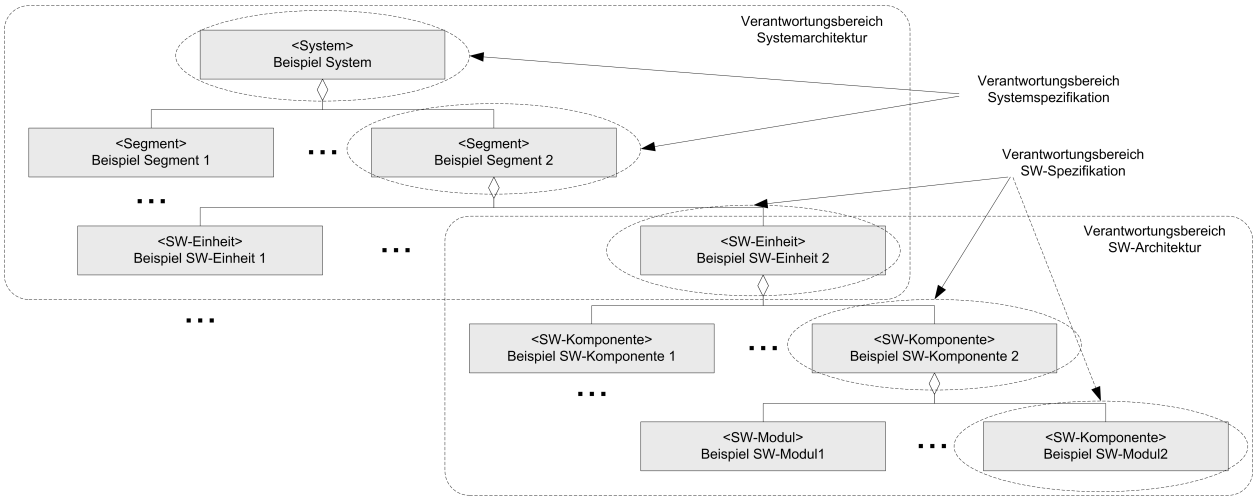

Abbildung 8.7: Korrelation von Systemelementen und Dokumenten im V-Modell XT

Das V-Modell XT macht explizit keine Vorgaben hinsichtlich der Methodik zur Erstellung der Architekturen oder der Spezifikationen, implizit liegt jedoch der hierarchischen Zerlegung des Systems sowie den Themenstruktur der Produkte eine konkrete Philosophie einer geeigneten Methode zu Grunde. So werden eine Reihe sehr konkreter Anforderungen an Entwurfsmethoden gestellt, z.B.

- Die Unterstützung der Zerlegung eines Systems in komponentenähnliche Konzepte.
- Die Unterstützung eines zentralen Architekturdokuments als Beschreibung der Komponentenhierarchie.

- Die Unterstützung einer Hierarchie der Schnittstellenspezifikationen parallel zur Komponentenhierarchie.
- Die Abdeckung der durch Themen geforderten unterschiedlichen Sichten auf das System

Der komponentenbasierte Ansatz des V-Modells XT impliziert, dass komponentenbasierte Methoden geeignet in das V-Modell integriert werden können. Dies ist jedoch bei näherer Betrachtung nicht der Fall. Wie das V-Modell verfolgt auch jede Methode ihre eigene Philosophie, die häufig nicht oder nur schlecht mit der V-Modell Philosophie zusammenpasst. Eine ausführliche Diskussion zu diesem Thema findet sich beispielsweise in [102]. In dieser Arbeit wurde am Beispiel der Methode KobrA [6] aufgezeigt, welche Probleme die Integration einer komponentenorientierten Methode in das V-Modell mit sich bringt. So ist KobrA eine Methode, die ähnlich zum V-Modell eine Systemzerlegung in Komponenten unterstützt und eine Spezifikation für jede Komponente vorsieht. KobrA unterstützt jedoch nicht den Begriff einer übergeordneten Systemarchitektur und definiert keine entsprechenden Artefakte. Bei einer Integration der Methode KobrA in das V-Modell bleiben somit die Architekturprodukte unberücksichtigt. Aus Sicht des V-Modells stehen jedoch gerade die Architekturen im Zentrum des Systementwurfs.

Als weiteres Beispiel kann die Methode von Andresen zur komponentenbasierten Softwareentwicklung (vgl. Abschnitt A.4) genannt werden. Auch bei dieser Methode handelt es sich um eine komponentenorientierte Entwurfsmethode. Die Methode stützt sich jedoch auf einen sichtenbasierten Ansatz zur Beschreibung der Systemarchitektur auf unterschiedlichen Granularitätsebenen. Komponenten werden als Teil der Architektursichten beschrieben. Die Erstellung unabhängiger Spezifikationen für Komponenten ist in dieser Methode dagegen nicht vorgesehen. Auch hier ist eine unmittelbare Integration nicht sinnvoll möglich.

Die Beispiele zeigen, dass die Anforderungen des V-Modells teilweise zu stark ins Detail gehen. Ähnlich zu Vorgehensmodellen, die bereits eine feste Methodik vorgeben, lässt das V-Modell im Bereich des Systementwurfs fast keine existierenden Methoden zu, sondern fordert im Grunde eine explizit auf die Anforderungen hin entwickelte Entwurfsmethode. Ziel sollte es jedoch sein, generische Vorgehensmodelle unabhängig von Methodik und Methoden unabhängig von den Anforderungen der Vorgehensmodelle zu halten um so die Wiederverwendung und Flexibilität bei der Zuordnung zu erhalten. Als mögliche Lösung des Problems wird vorgeschlagen, das Vorgehensmodell auf Produkte zu beschränken, die von Bedeutung für den Projektablauf sind. Dies sind insbesondere alle Produkte, die am Ende einer Phase alle Ergebnisse eines Teilprozesses oder einer Rolle repräsentieren. Diese Produkte werden der Qualitätssicherung unterzogen und definieren Eingangs- und Ausgangskriterien für die Phasen. Dies könnten beispielsweise ein Produkt *Grobentwurf* für das Ende der Entwurfsphase sowie ein Produkt *Feinentwurf* für das Ende des detaillierten Designs darstellen. Jedes der Produkte fasst auf einer allgemeinen Ebene die minimal erwarteten Entwurfsergebnisse der jeweiligen Phase zusammen. Damit wird eine wesentlichen höhere Flexibilität hinsichtlich der Methodenauswahl für das V-Modell erreicht ohne Einbußen der Qualität.

8.6 Konzept zu Umsetzung und Anwendung

In diesem Abschnitt wird eine mögliche Umsetzung und Anwendung des Integrationsmodells auf Basis der Technologien und Werkzeuge des V-Modells XT vorgestellt. Das Metamodell des V-Modells XT ist formal in Form eines XML Schemas definiert. Dies erlaubt die Entwicklung und Manipulation des V-Modells mit Hilfe spezifischer Werkzeuge. Zur Bearbeitung des Modells stehen zwei Werkzeuge zur Verfügung: der V-Modell XT Editor zur Manipulation der Modellinhalte im Rahmen der organisationsspezifischen Anpassung und der Projektassistent zur Bildung von Teilmodellen und zur Projektinitiierung im Rahmen der projektspezifischen Anpassung. Ziel dieses Abschnitts ist es zu zeigen, wie Technologien und Werkzeuge des V-Modells XT für Integrationsmodelle eingesetzt werden können, die auf dem V-Modell Metamodell basieren.

8.6.1 Die Technologien

Das V-Modell XT ist mit seinem Metamodell vollständig in XML Notation beschrieben. Die *eXtensible Markup Language* XML ist eine deskriptiver Sprache, mit deren Hilfe beliebig komplexe Datenstrukturen mit ihren Abhängigkeiten in einem Dokument beschrieben werden können. Als deskriptive Sprache fehlt der XML die Definition einer Semantik. Diese wird in Form eines XML Schemas hinterlegt. XML Schema ist eine auf der XML basierende Sprache zur Beschreibung von XML Typsystemen. Mit einem XML Schema wird für jedes Element in einem XML Dokument ein expliziter Typ mit Namen, Attributen und Wertebereichen definiert. Ein XML Dokument wird als *wohlgeformt* bezeichnet, wenn es die strukturellen Vorgaben der XML Spezifikation allgemein erfüllt. Es ist zusätzlich auch *gültig*, wenn es den Typen seiner XML Schema Beschreibung genügt. An dieser Stelle wird nicht weiter auf die XML Technologien eingegangen, sondern der technische Aufbau des V-Modells und ein Ansatz zur Erweiterung zum Integrationsmodell vorgestellt. Für weitere Informationen zu XML und XML Schema wird auf die Literatur, insbesondere auf die entsprechenden Spezifikationen verwiesen ([121], [133]).

Aus technischer Sicht besteht das V-Modell aus zwei Dateien, einer XML Datei mit dem Modell selbst (V-Modell-XT.xml) und seiner Schema Datei mit der Beschreibung des Metamodells (V-Modell-XT-Metamodell.xsd). Bei Erweiterung des V-Modells XT zu einem Integrationsmodell, übertragen wir diese Struktur auf Methodenbibliothek und Methodenintegration. Abbildung 8.8 zeigt den resultierenden strukturellen Aufbau des Integrationsmodells anhand der relevanten XML und XML Schema Dateien.

Das Integrationsmodell entspricht hier der Datei *Integrationsmodell.xml*. Es inkludiert drei Dateien: *V-Modell-XT.xml* mit dem V-Modell selbst, *Methodenbibliothek.xml* mit der Methodenbibliothek sowie *Methodenintegration.xml* zur Verknüpfung von Bibliothek und Vorgehensmodell. Die entsprechenden Schema Dateien mit den Metamodell-Definitionen finden sich in den Dateien *V-Modell-*

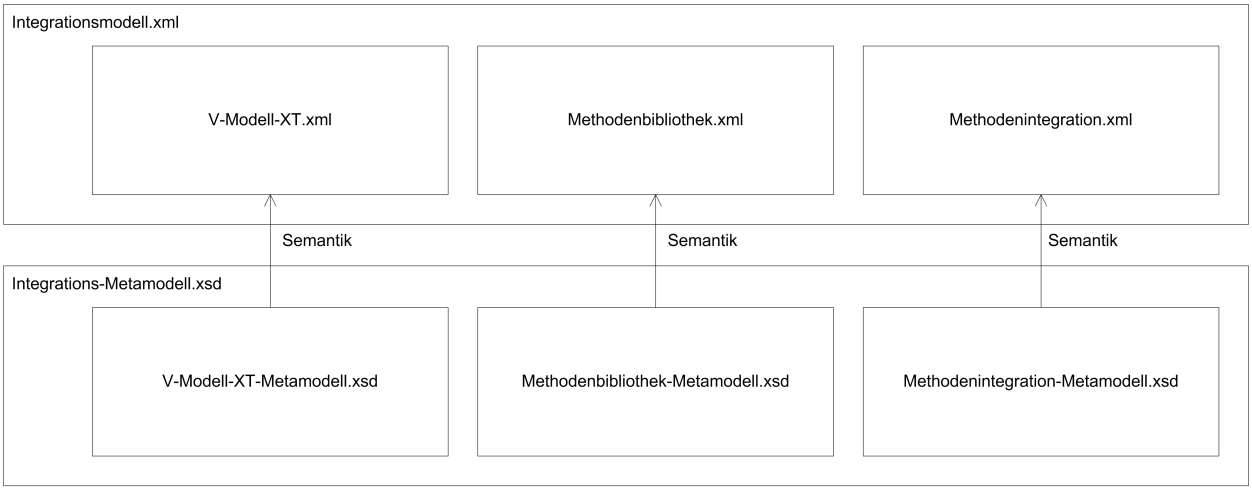

Abbildung 8.8: Technische Umsetzung des Integrationsmodells

XT-Metamodell.xsd, *Methodenbibliothek-Metamodell.xsd* und *Methodenintegration-Metamodell.xsd*. Diese Dateien sind in der Datei *Integrations-Metamodell.xsd* zusammengefasst.

8.6.2 Integrationsmodelle im V-Modell XT Editor

Zur Bearbeitung des V-Modells steht der V-Modell XT Editor zur Verfügung. Der V-Modell Editor ist im Kern ein XML Editor, erweitert um Funktionalitäten zur Textverarbeitung. Als generischer XML Editor kann er alle Arten wohlgeformter und gültiger XML Dateien interpretieren. Insbesondere kann er auch zur Darstellung und Bearbeitung von Integrationsmodellen verwendet werden, die der Struktur in Abbildung 8.8 entsprechen. Abbildung 8.9 zeigt das Integrationsmodell im V-Modell Editor.

In der Navigationsleiste links lässt sich die dreiteilige Struktur des Metamodells erkennen. Ein vollständiges Integrationsmodell umfasst das Vorgehensmodell, die Methodenbibliothek und die Schnittstelle zur Zuordnung der Methoden zum Vorgehensmodell. Der Editor unterstützt konkret die organisationsspezifische Anpassung des Vorgehensmodells im Integrationsmodell (vgl. Abschnitt 7.4), die organisationsspezifische Anpassung und Erweiterung der Methodenbibliothek sowie die Zuordnung der Methodenartefakte zu Produkten im Vorgehensmodell.

Der V-Modell XT Editor kann als allgemeiner XML Editor Integrationsmodelle auf Basis der V-Modell Technologien sowie der in Abbildung 8.8 dargestellten Dateistruktur ohne Probleme darstellen. Nicht im Metamodell definiert sind dagegen die Konsistenzbedingungen. In der aktuellen Version des Editors liegen die Konsistenzbedingungen zum V-Modell im Code implementiert vor. Damit der Editor Integrationsmodelle adäquat bearbeiten und insbesondere ihre Konsistenz prüfen kann, ist zusätzlich eine Implementierung der Konsistenzbedingungen zu Methodenbibliothek und Methodenintegration erforderlich.

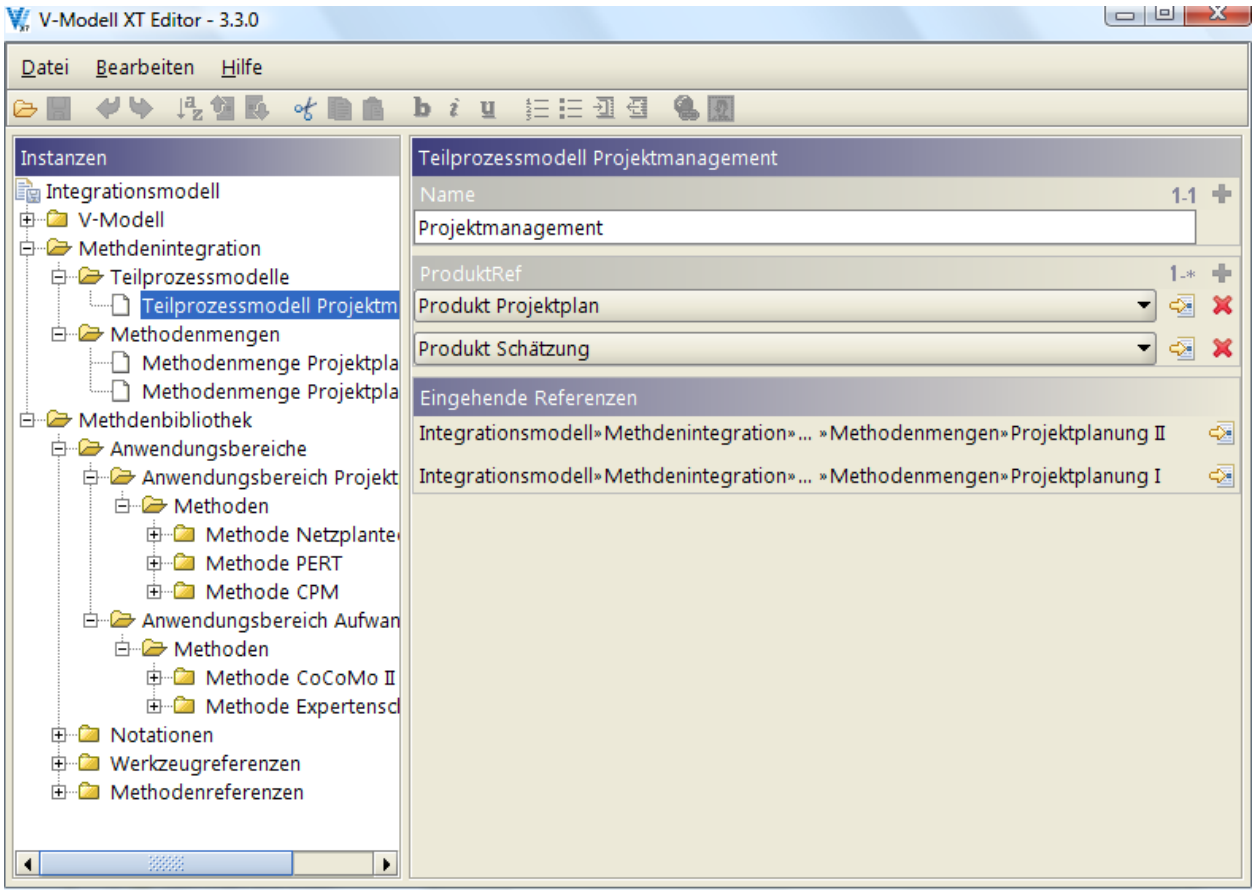

Abbildung 8.9: Das Integrationsmodell im V-Modell XT Editor

8.6.3 Projektspezifische Anpassung mit dem Projektassistenten

Das im V-Modell Editor organisationsspezifisch angepasste Integrationsmodell steht den Projekten der Organisation zur Anwendung zur Verfügung. Erster Schritt ist die projektspezifische Anpassung, mit der das Integrationsmodell unmittelbar auf die Bedürfnisse eines spezifischen Projekts angepasst wird. Als Werkzeug dient hier der V-Modell XT Projektassistent. Der Projektassistent interpretiert ähnlich zum Editor XML und XML Schema Dateien des V-Modells, erlaubt jedoch im Gegensatz zum Editor keine Änderungen am Modell selbst. Die Regeln für die projektspezifische Anpassung sind im Tailoringmodell des V-Modells definiert (vgl. Abschnitt B.4).

Die projektspezifische Anpassung von Integrationsmodellen mit Hilfe des Projektassistenten entspricht weitgehend der projektspezifischen Anpassung des V-Modells. Das Vorgehensmodell im Integrationsmodell wird auf die Bedürfnisse des entsprechenden Projekts angepasst. Hinzu kommt hier die projektspezifische Auswahl von Methodenmengen. Der Projektleiter hat die Möglichkeit, im Integrationsmodell entweder über Projektmerkmale oder manuell die für sein Projekt geeignete Methodenmengen auszuwählen. Abbildung 8.10 zeigt einen Screenshot des Projektassistenten.

Nach erfolgter projektspezifische Anpassung unterstützt der Projektassistent die Initialisierung des Projekts. Konkret erfolgt mit Hilfe des Assistenten die Generierung eines initialen Projektplans, die

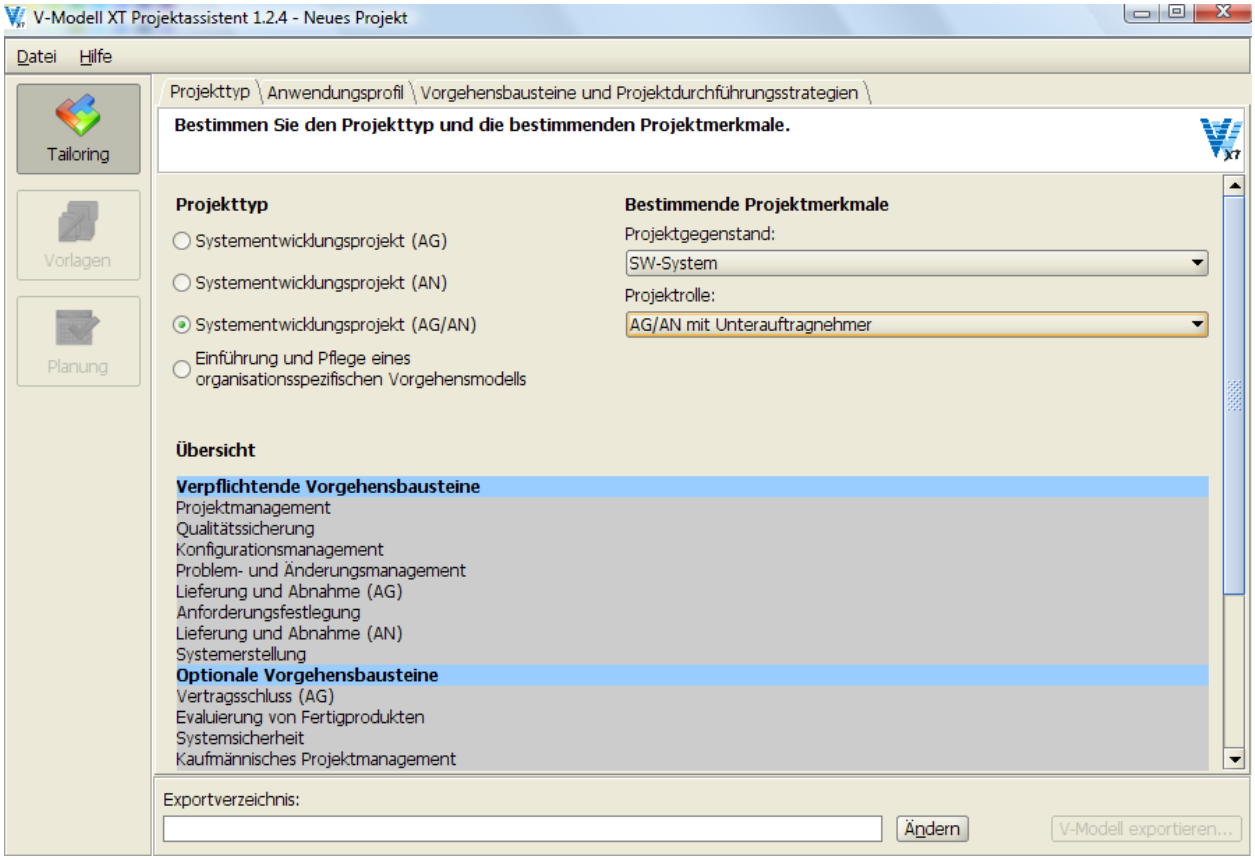

Abbildung 8.10: Der V-Modell XT Projektassistent (aktueller Stand)

Generierung von projektspezifischen Vorlagen sowie die Generierung einer sichtenbasierten Dokumentation des projektspezifischen V-Modells. Die Integration von Methoden wirkt sich insbesondere auf die Generierung von Vorlagen und Dokumentation aus. Jede generierte Vorlage entspricht einem Produkt im Vorgehensmodell. Die Kapitelstruktur der Vorlage bildet die Themenstruktur im Vorgehensmodell nach. Produkt- und Themenbeschreibungen geben vor, welche Inhalte in dieser Vorlage zu dokumentieren sind. Die Beschreibungen der dem Produkt bzw. den Themen zugeordneten Artefakte liefern methodische Hinweise zur Erarbeitung und Dokumentation der Ergebnisse. In ähnlicher Weise ergänzen Artefakt- und Arbeitsschrittbeschreibungen Produkte und Aktivitäten in der generierten projektspezifischen Dokumentation. Dem Anwender im Projekt steht somit nicht nur eine Beschreibung zur Verfügung was zu erarbeiten ist, sondern er erhält zusätzlich eine durchgängige methodische Hilfestellung, wie die erwarteten Ergebnisse zu erstellen sind. In welcher Form die im Integrationsmodell vorliegenden Informationen bei der Generierung tatsächlich genutzt werden, ist prinzipiell nicht festgelegt. So ist beispielsweise eine getrennte Darstellung von Methoden und Vorgehensmodell mit Verweisen möglich oder eine integrierte Darstellung von Methoden im Vorgehensmodell.

8.6.4 Vorschlag zur rollenbasierten Dokumentation

Zum Abschluss beschäftigt sich dieser Abschnitt mit der Struktur der V-Modell Dokumentation, wie sie aktuell vorliegt und wie sie prinzipiell für Integrationsmodelle übernommen werden kann. Die aktuelle Darstellung geht von einer neunteiligen Struktur der Dokumentation aus. Jeder Teil repräsentiert eine spezifische Sicht auf das V-Modells. Nach IEEE STD 1471-2000 entspricht eine Sicht auf ein Modell der Darstellung des Modells aus einer spezifischen Perspektive. Eine Sicht beschreibt alle Aspekte, die für diese Perspektive von Belang sind und blendet andere Aspekte aus. So repräsentiert ein Teil der V-Modell Dokumentation die Produktsicht auf das V-Modell, ein anderer die Aktivitätssicht. In der Produktsicht sind alle Produkte, Themen und Abhängigkeiten zwischen Produkten im V-Modell dokumentiert. Die Aktivitätssicht fasst Aktivitäten und Teilaktivitäten zusammen. Als weitere Sichten bietet das V-Modell beispielsweise die Rollensicht und die Tailoringsicht.

Wie die Beispiele zeigen, orientiert sich das Sichtenkonzept an der Teilmodellstruktur des V-Modells. Diese Art der Darstellung wird auch von anderen Vorgehensmodellen unterstützt. Die Darstellung der Sichten erfolgt über die entsprechenden Werkzeuge der Vorgehensmodelle. Sichtenbasierte Ansätze sind relativ einfach zu erstellen. Sie helfen die Komplexität eines Vorgehensmodell zu reduzieren und so das Verständnis für den Aufbau des Modells zu verbessern. Sie bieten jedoch nur bedingt Unterstützung für die Anwender des Vorgehensmodells in Projekten. So berücksichtigt der sichtenbasierte Ansatz nicht die Darstellung sichtenübergreifender Informationen in einer Sicht. Dies sind jedoch Informationen, die von den jeweiligen Anwendern des Modells benötigt werden. Aus diesem Grund wird im Folgenden ein Vorschlag für ein Sichtenkonzept vorgestellt, der als Erweiterung des sichtenbasierten Ansatzes eingeordnet werden kann. Der Ansatz orientiert sich an der Rollenstruktur im Vorgehensmodell. Alle Informationen aus allen Sichten, die eine Rolle oder eine Rollengruppe benötigt, wird zusammengefasst dargestellt. Abbildung 8.11 stellt den Ansatz zur Darstellung von Integrationsmodellen beispielhaft dar. Danach erhalten beispielsweise Softwareentwickler in einer Entwicklersicht alle relevanten Informationen aus allen Teilmodellen, die den Prozess zur Entwicklung des Softwaresystems betreffen. Diese Sicht benötigt Informationen aus allen Teilmodelle eines Vorgehensmodells. Informationen, die nicht benötigt werden, wie beispielsweise das Vorgehen zur Projektorganisation, werden jedoch ausgeblendet. Für Projektverantwortliche (Projektleiter, Projektmanager, Lenkungsausschuss) könnten dagegen in einer Projektmanagementsicht alle Informationen im Vorgehensmodell bezüglich Projektsteuerung und Projektmanagement zusammengefasst werden. Zur Darstellung von Methoden im Vorgehensmodell muss die Darstellung entsprechend erweitert werden. Insbesondere müssen Lösungen für folgende Probleme gefunden werden:

- Darstellung einer nahtlosen Integration der Methoden in das V-Modell.
- Geeignete und übersichtliche Darstellungsform der Methoden.

- Umgang mit Artefakten, die einem Produkt zugeordnet wurden.
- Umgang mit Artefakten einer Methode, die keinem Produkt oder Thema zugeordnet wurden.

Die Darstellung des Vorgehensmodells mit integrierten Methoden sollte nahtlos und vollständig sein. Sinnvoll ist die Darstellung der Produkte und Themen mit den zugeordneten Artefakten, die Darstellung der Aktivitäten mit den zugeordneten Arbeitsschritten. Allgemeine Informationen zu den Methoden könnten in einer eigenen Methodensicht zusammengefasst werden. Ergebnis ist ein methodenspezifisches V-Modell, in dem Methoden nahtlos und vollständig in die V-Modell Konzepte integriert sind.

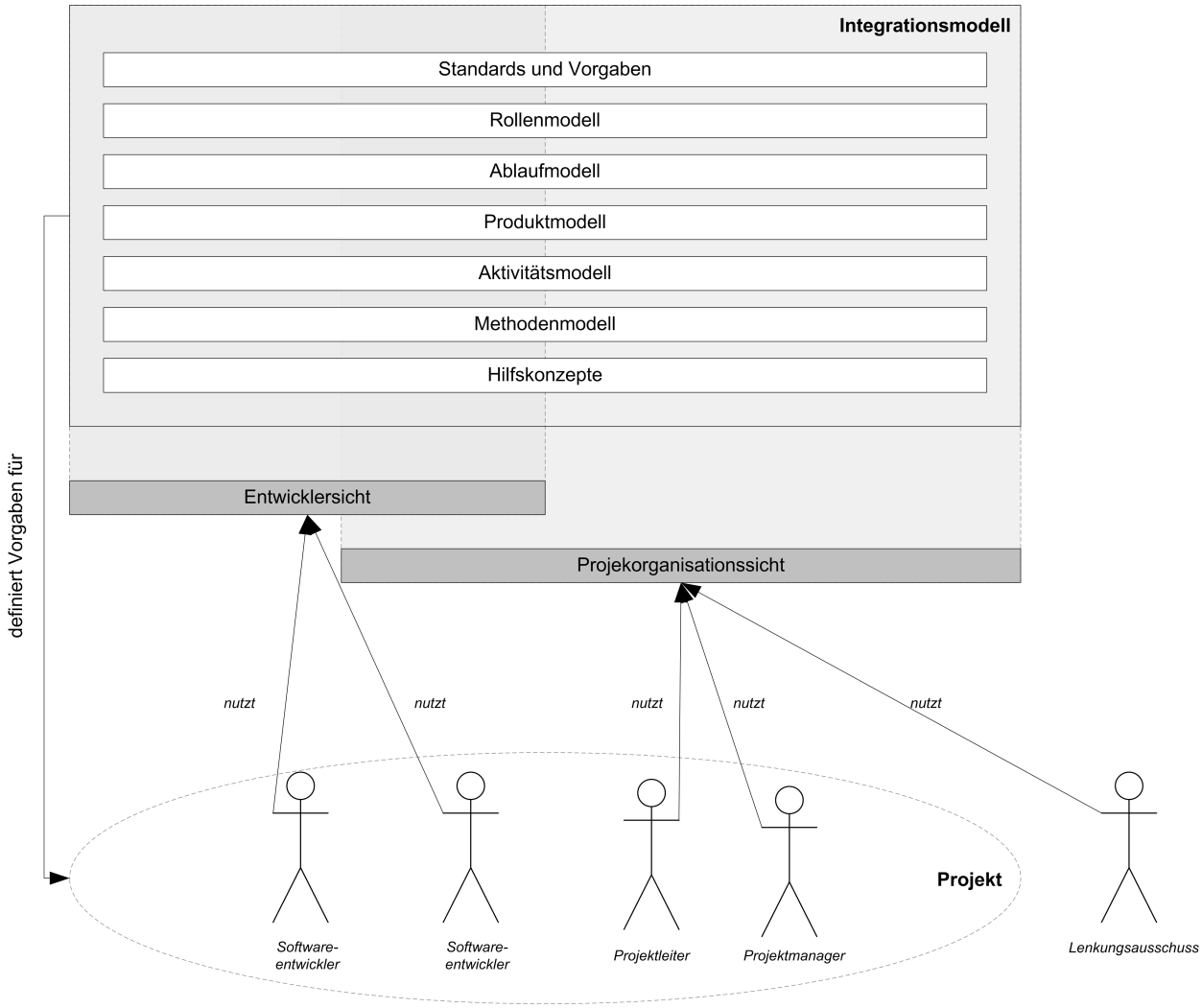

Abbildung 8.11: Rollenbasierte Sichtenstruktur auf Vorgehensmodelle

8.7 Zusammenfassung

In diesem Kapitel wurde am Beispiel des V-Modell XT Metamodell gezeigt, wie ein existierendes Vorgehens-Metamodell um das Konzept der dynamischen Methodenintegration erweitert werden

kann. Es wurde eine konkrete Ausgestaltung der Schnittstelle auf Basis der Konzepte Teilprozessmodell und Methodenmenge aufgezeigt und die Integration des Auswahlmechanismus für Methoden in das Tailoringmodell des V-Modells vorgestellt. Das so erweiterte V-Modell Metamodell wurde anschließend anhand der in Kapitel 6 entwickelten Beispielbibliothek evaluiert. Abschließend wurde anhand der V-Modell XT Technologien und Werkzeuge die Umsetzung und Anwendung des zum Integrationsmodell erweiterten V-Modells gezeigt.

Das hier vorgestellte Integrationsmodell setzt auf dem V-Modell XT als Vorgehensmodell auf. Prinzipiell ist jedoch die Wahl des Vorgehensmodells frei, solange das Metamodell die in Kapitel 6 diskutierten Eigenschaften an der Schnittstelle zu den Methoden erfüllt.

Kapitel 9

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Ansatz entwickelt, der die optimale methodische Anpassung von Vorgehensmodellen in Organisationen unterstützt. Im Zentrum der Arbeit stehen die formale Spezifikation eines allgemeinen Methodenbegriffs sowie die Erweiterung eines Vorgehens-Metamodells hin zu einem Integrations-Metamodells für Vorgehensmodelle mit dynamisch integrierbaren Methoden. Konkret wurden folgende Ergebnisse erarbeitet:

- Anhand einer Menge repräsentativer Methoden wurden strukturelle und inhaltliche Eigenschaften von Methoden identifiziert und informell beschrieben. Als zentrale Elemente einer Methode wurden Verfahren und Domänentheorien ermittelt. Das Verfahren beschreibt den Prozess zur Lösung des von der Methode adressierten Problems, die Domänentheorie liefert über Informationstypen und ihre Beziehungen die Sprache für das Verfahren. Daneben existieren eine Reihe von Konzepten, die Methoden mittelbar oder unmittelbar beeinflussen. Anhand der Ergebnisse wurde eine allgemeine Methodenontologie abgeleitet und eine Methodendefinition gegeben. Des Weiteren wurde eine Methodenklassifikation vorgestellt. Die Klassifikation basiert auf dem Schwerpunkt der Ausarbeitung einer Methode. Danach können theorienahe, verfahrensnahe und anwendungsnahe Methoden unterschieden werden. Theorie- und verfahrensnahe Methoden liefern entsprechend ihrem Namen eine allgemeine Domänentheorie bzw. ein Verfahren und dienen vorrangig als Vorbilder für die Entwicklung anwendungsnaher Methoden.

- Aufbauend auf der informellen Methodendefinition wurde ein formales Methoden-Metamodell mit formaler abstrakter Syntax und Semantik spezifiziert. Ziel des Metamodells war es, die Beziehungen zwischen Methoden und ihrer Domänentheorie formal zu erfassen und Konsistenzbedingungen für Methoden auf Basis der von ihnen verwendeten Informationstypen abzuleiten. Damit wurde die Grundlage für eine formale Erfassung der Schnittstelle von Vorgehensmodell und Methoden, ebenfalls über Spezifikation der abstrakten Syntax und Semantik, gelegt. Hier wurden insbesondere die Wechselbeziehungen zwischen Vorgehensmodell und Methoden an der Schnittstelle beschrieben.

- Auf Basis des formalen Methoden-Metamodells und der formalen Beschreibung der Methodenintegration wurde ein Metamodell zur Erweiterung von Vorgehens-Metamodellen hin zu Integrations-Metamodellen entwickelt. Integrations-Metamodelle bestehen aus drei Teil-Metamodellen, einem Vorgehens-Metamodell, einem Metamodell der Methodenbibliothek und einem Metamodell zur Integration von Methoden der Methodenbibliothek in das Vorgehensmodell. Das Integrations-Metamodell stützt sich auf Erkenntnisse der formalen Methodenintegration, erfasst jedoch lediglich strukturelle Aspekte. Die Wechselbeziehungen von Vorgehensmodell und Methoden mit ihrer Domänentheorie werden, um die Anwendbarkeit des Metamodells zu erhalten, nur in begrenztem Umfang berücksichtigt. Abschließend wurde ein Entwicklungs- und Anpassungsprozess für Integrationsmodelle als Instanzen des Metamodells, sowie eine mögliche Umsetzung des Integrations-Metamodells auf Basis des V-Modell XT Metamodells sowie der V-Modell XT Technologien vorgestellt.

Bei der hier entwickelten Metamodellerweiterung handelt es sich um einen vollkommen neuen Lösungsansatz für das Anpassungsproblem von Vorgehensmodellen. Organisationen, die ein Vorgehensmodell einführen möchten, brauchen sich nicht für ein methodenneutrales Vorgehensmodell oder ein methodenspezifisches Vorgehensmodell, jeweils mit den entsprechenden Vor- und Nachteilen, zu entscheiden. Vielmehr steht mit einem Integrationsmodell ein Vorgehensmodell zur Verfügung, das alle Varianten abdeckt, wobei die Entscheidung auf Projektebene getroffen werden kann.

Bei entsprechender organisationsspezifischer Anpassung eines Integrationsmodells kann im Rahmen der projektspezifischen Anpassung gewählt werden, ob das Vorgehensmodell im Integrationsmodell komplett ohne Methoden, teilweise um Methoden ergänzt oder vollständig um Methoden erweitert angewendet werden soll. Des Weiteren besteht die Möglichkeit, Methodenmengen je nach Projekspezifika auszuwählen.

Wie in Kapitel 6 gezeigt wurde, gibt es verschiedene Ansätze, die das Problem der Methodenintegration adressieren. Diese Ansätze erlauben durchweg eine gewisse Variabilität bei der Methodenzuordnung im Rahmen der organisationsspezifischen Anpassung. Die projektspezifische Anpassung des Vorgehensmodells und die projektspezifische Auswahl von Methoden im Vorgehensmodell wird dagegen nicht ausreichend unterstützt. Der hier vorgestellte Ansatz erlaubt an dieser Stelle eine wesentlich höhere Flexibilität. Kernidee ist die strikte Entkopplung von Vorgehensmodell und Methoden. Erreicht wird dies durch die klare Trennung von Vorgehensmodell und Methoden auf Metamodellebene sowie die Einführung von zwei neuen Konzepten zur Modellierung der Schnittstelle: das Teilprozessmodell und die Methodenmenge. Diese Konzepte stellen den Schlüssel zur Entkopplung dar. Sie unterstützen eine, je nach Projektsituation austauschbare, Zuordnung von Methoden zum Vorgehensmodell, ohne Methoden und Vorgehensmodell selbst zu verändern oder zu beeinflussen.

Aufgrund der Vielschichtigkeit und Heterogenität der hier betrachteten Methodenmenge war ein Kernproblem dieser Arbeit die Abstraktion und Vereinheitlichung der Methoden hin zu einem allgemein gültigen Methoden-Metamodell. Insbesondere existiert zum aktuellen Zeitpunkt kein einheitliches Verständnis darüber, wie eine Methode aufgebaut ist, was eine Methode tatsächlich charakterisiert und wie eine Methode definiert werden kann. So muss an dieser Stelle darauf hingewiesen werden, dass die Entscheidung, was eine Methode ist und was nicht, nur mit einer gewissen Subjektivität getroffen werden kann. Auch bei den hier als empirische Basis angegebenen Methoden kann es unterschiedliche Meinungen dazu geben, ob mit ihnen die Menge der Methoden tatsächlich repräsentativ abgedeckt wird, oder ob gewisse Eigenschaften von Methoden fehlen. Das Metamodell der Methodenbibliothek in Kapitel 6 wurde aus diesem Grund sehr allgemein und flexibel gestaltet. Ziel war es, eine möglichst breite Abdeckung von Methoden zu erreichen, die mit dem Metamodell beschrieben werden können. Dies wird insbesondere im Unterschied zum formalen Methoden-Metamodell in Kapitel 5 deutlich, das in den Details sehr viel präziser spezifiziert wurde.

Neben den Möglichkeiten von Integrations-Metamodellen müssen jedoch auch die Grenzen des hier vorgestellten Ansatzes aufgezeigt werden. Ein Integrations-Metamodell gibt zwar prinzipiell das Vorgehensmodell im Integrationsmodell nicht vor, fordert jedoch ein Vorgehensmodell, dessen Metamodell nach dem resultat- bzw. produktorientierten Ansatz aufgebaut ist. Produktorientierte Ansätze geben keine explizite Reihenfolge für die Erarbeitung von Produkten innerhalb einer Phase vor. Sie kennen somit kein eigenständiges Ablaufmodell für ihre Aktivitäten. Dieser Ansatz ist notwendig um Inkonsistenzen bzw. Widersprüche zwischen Aktivitäten im Vorgehensmodell und dem Verfahren der Methoden zu vermeiden. Aktivitätsorientierte Vorgehensmodell stellen dagegen die Aktivitäten und ihre Durchführungsreihenfolge in den Vordergrund. Hier sind bei der Methodenintegration Konflikte zwischen Vorgehensmodell und Methoden unvermeidbar. Dies bedeutet in letzter Konsequenz, dass für die verhältnismäßig große Anzahl der aktivitätsorientierten Vorgehensmodell, dies sind insbesondere alle Vorgehensmodelle, die auf dem SPEM [88] basieren, der hier entwickelte Ansatz nicht unmittelbar anwendbar ist.

Als ein weiteres eigenständiges Ergebnis dieser Arbeit kann abschließend das Metamodell der Methodenbibliothek selbst genannt werden. Das Metamodell repräsentiert einen einheitlichen Entwicklungsrahmen für Methoden. Im stark heterogenen Bereich der Methoden unterstützt das Metamodell eine einheitliche und standardisierte Darstellungsform für Methoden. Vorteile bringt dies sowohl für die Entwicklung und Dokumentation der Methoden selbst, wie auch für ihre Anwendung. Methodenautoren erhalten mit dem Metamodell ein Werkzeug zur übersichtlichen, vollständigen und strukturierten Methodenentwicklung und -dokumentation. Lücken, Unklarheiten, Redundanzen und Widersprüche werden durch die vorgegebenen Methodenstruktur weitgehend vermieden. Für den Methodenanwender erleichtert ein einheitliches Methodenformat dagegen das Verständnis und die Anwendung der Methode. Neben ihrem Einsatz in Vorgehensmodellen sind daher auch andere Anwendungsszenarien für Methodenbibliotheken denkbar. So könnten vordefinierte Methodenbibliotheken, nach dem Vorbild von Open Source Komponentenbibliotheken, in Communities

entwickelt oder kommerziell zur Verfügung gestellt werden. Methodenbibliotheken eignen sich beispielsweise als eigenständige Nachschlagewerke für Methoden und würden innerhalb der Community aktuell gehalten, was einen nicht zu unterschätzenden Vorteil gegenüber schnell veralternden Büchern darstellen würde.

Anhang A

Beispielmethoden

Im Folgenden werden einige Methoden beschrieben, die im Rahmen der Arbeit als Beispiele oder Referenzen verwendet werden.

A.1 Function Point Analyse

Bei der Function Point Analyse (FPA) handelt es sich um eine Methode zur Ermittlung eines objektiven Größenmaßes für Softwareanwendungen oder Projekte. Dokumentiert ist die FPA im Function Point Counting Manual [58]. Verwaltet und weiterentwickelt wird die Methode von der International Function Point User Group (IFPUG). Mittlerweile ist die FPA auch als ISO Standard anerkannt (ISO 20926). Der Standard liegt englischer Form vor; bei den im Folgenden verwendeten deutschen Begriffen wurde die Übersetzung von [95] verwendet.

Als allgemeine Messmethode kann die FPA in verschiedenen Projektphasen zu unterschiedlichen Zwecken eingesetzt werden. [95] nennt als Beispiele die Anwendungsbereiche Projektmanagement (Aufwandsschätzung), Controlling (Leistungsbewertung) und Einkauf (Bewertung und Vergleich von Angeboten). Als Messgrundlage verwendet die FPA Elementarfunktionen und Datenbestand einer Anwendung. Als Elementarfunktionen bezeichnet der Standard atomare Funktionalität bzw. atomare Datengruppen, die dem Anwender eine zusammenhängende fachliche Information oder Funktionalität zur Verfügung stellen. Die FPA unterscheidet drei Arten von Funktionen:

- Eingabe: fachliche Daten kommen von außerhalb in die zu bewertende Anwendung hinein.
- Ausgabe: fachliche Daten verlassen die zu bewertende Anwendung.
- Abfrage: die Anwendung stellt fachliche Daten bereit.

sowie zwei Arten von Datenbeständen

- Interner Datenbestand: Fachliche Daten, die von der Anwendung selbst angelegt, verändert, gelöscht werden.

- Referenzdatenbestand: Fachliche Daten, die von der Anwendung ausschließlich gelesen werden.

Zur Bewertung müssen alle relevanten Elementarfunktionen und Datenbestände der jeweiligen Anwendung identifiziert werden. Jeder ermittelten Elementarfunktion und jedem ermittelten Datenbestand wird anhand der im Manual definierten Regeln ein expliziter Function Point Wert zugeordnet. Mit der Summe der Function Point Werte erhält man die funktionale Größe eines IT Systems. Function Points dienen so als objektive Kennzahlen für die quantitative Beschreibung der Leistung einer Software.

Die so ermittelten Function Points werden auch als unadjusted Function Points (uFP) bezeichnet. Diese sind zu unterscheiden von den adjusted Function Points (aFP). Adjusted Function Points entsprechen unadjusted Function Points multipliziert mit einem Wertfaktor. Der Wertfaktor berücksichtigt zusätzlich qualitative Leistungsmerkmale eines Systems. Als qualitative Systemmerkmale nennt der Standard beispielsweise die verteilte Verarbeitung, die Wiederverwendbarkeit oder die Leistungsanforderungen. Der Wertfaktor (Value Adjustment Factor (VAF)) wird anhand von insgesamt vierzehn Systemmerkmale bestimmt. Jedem Systemmerkmal wird ein Wert zwischen 0 (niedriger Einfluss) und 5 (hoher Einfluss) zugeordnet werden. Das Resultat wird als gesamter Einflussfaktor (Total Degree of Influcence (TDI)) bezeichnet. Der Wertfaktor VAF wird entsprechend dem Standard nach folgende Formel berechnet:

$$VAF = TDI * 0,01 + 0,65$$
$$aFP = uFP * VAF$$

Die Durchführung einer Function Point Analyse erfolgt in drei Schritte:

- Bestimmung der Zielsetzung der Analyse: Im Function Point Counting Manual stehen drei Formen der Analyse zur Auswahl: die Analyse einer Anwendung, eines Neutentwickungsprojekts oder eines Weiterentwicklungsprojekts. Bei der Analyse selbst wird zwischen einer Schätzung (Funktionsumfang einer noch zu entwickelnden Anwendung) und einer Zählung (Funktionsumfang einer bestehenden Anwendung) unterschieden.

- Abgrenzung der Analyse: In diesem Schritt muss der Analyseumfang ermittelt werden. Dieser ergibt sich bei der Analyse eines Projekts aus dem Projektantrag, bei der Analyse einer existierenden Anwendung aus den Anwendungsgrenzen. Die Bestimmung der Anwendungsgrenzen erfolgt nach den im Function Point Manual definierten Regeln.

- Die Analyse selbst: Im Rahmen der Analyse wird anhand der Elementarprozesse ein objektiver Function Point Wert ermittelt, der quantitative und qualitative Leistungsmerkmale des zu analysierenden Gegenstands ermittelt. Zur Durchführung der Analyse werden folgende Schritte durchgeführt:

- Ermittlung und Bewertung der Elementarprozesse
- Ermittlung und Bewertung der Datenbestände
- Berechnung des unadjusted Function Point Wertes
- Ermittlung des Wertefaktors
- Ermittlung des adjusted Function Point Wertes

Regeln, Bestimmungen und Definitionen zu allen der genannten Schritte gibt das Function Point Counting Manual vor.

A.2 CoCoMo II

Bei CoCoMo (Constructive Cost Model) [19] handelt es sich um ein Schätzmodell mit Methode zur Ermittlung des Schätzaufwands. Entwickelt wurde das Modell von Barry Boehm, die Weiterentwicklung und Pflege des Modells liegt in der Verantwortung des Center of Software Engineering (CSE) [136] an der University of Southern California. Aktuelle Version des Modells ist CoCoMo II.

Ziel von CoCoMo ist es, anhand einer kontinuierlich gepflegten Erfahrungsdatenbasis zu Softwareprojekten eine zuverlässige Basis für die Schätzung von Projektaufwänden bereitzustellen. Zur Berechnung von Projektaufwänden stellt das Modell eine Formel bereit. Die Formel bringt Aufwand und Projektgröße in einen Zusammenhang, wobei für die Anwendung in Organisationen Kalibrierungsfaktoren zur Verfügung stehen. Die Formel wird wie folgt angegeben:

$$PM = A * K * (Size)^S$$

PM entspricht in dieser Formel dem zu berechnenden Aufwand in Personenmonaten. Mit *Size* wird die Größe der Anwendung angegeben. Zur Berechnung der Größe kann die Function Point Analyse verwendet werden. CoCoMo erwartet als Größenangabe für die Variable *Size* die Einheit *Source Lines of Code (SLoC)* und stellt eine Umrechnungstabelle von unadjusted Function Points zu LoCs, abgestimmt auf die Eigenschaften verschiedener Programmiersprachen, zur Verfügung.

Die Variable S dient zur Skalierung der Formel. Sie entscheidet darüber, ob der Aufwand mit der Größe eines Projekts über- oder unterproportional wächst. Für S = 1 wächst das Verhältnis von Aufwand zu Projektgröße linear. Bei S < 1 wächst der Aufwand geringer als die Projektgröße. Bei S > 1 wächst der Aufwand stärker als die Projektgröße. Berechnet wird S auf der Basis von fünf Skalenfaktoren:

- Precedentedness (PREC): Vorerfahrung einer Organisation
- Development Flexibility (FLEX): Wie flexibel ist der Entwicklungsprozess in einer Organisation

- Risk Resolution (RESL): Wird in der Organisation Risikomanagement betrieben
- Team Cohesion (TEAM): Wie ist die Zusammenarbeit der Teammitglieder
- Process Maturity (PMAT): Wie hoch ist die Prozessreife in der Organisation (gemessen anhand von CMMI)

Zur Berechnung des Aufwands wird jeder Skalenfaktor von *Very Low* bis *Very High* bewertet und mit einem numerischen Wert gewichtet. Zur Ermittlung der Bewertung definiert das CoCoMo Modell explizite Regeln. Die Berechnung des Faktors S erfolgt dann anhand der im Manual vorgegebenen Formel:

$$S = 0,91 + 0,01 * \sum_{i=1}^{n} w_i$$

Die gewichteten Skalenfaktoren w_i werden aufsummiert und gehen so in die Berechnung der Variable ein.

Neben den Skalenfaktoren wirken sich noch eine Reihe weiterer Kostentreibern auf die Aufwände in einem Projekt aus, die ebenfalls im CoCoMo Modell berücksichtigt werden. Das Modell nennt 17 Kostentreiber, die sich auf die Aufwandsberechnung auswirken. In der Formel werden die Kostentreiber durch die Variable K repräsentiert.

$$K = \prod_{i=1}^{n} K_i$$

Als Kostentreiber nennt das Modell folgende Faktoren: Zuverlässigkeit, Mengengerüst, Komplexität (Programmflusskontrolle, Hardwaresteuerung, Datenmanagement, Benutzerschnittstelle), Wiederverwendbarkeit, Qualität der Dokumentation, CPU Zeit Beschränkungen, Hauptspeicheranforderungen, Plattformstabilität, Analytische Fähigkeiten des Projektteams, Fluktuation, Erfahrungen mit der Plattform, Erfahrungen mit der Technologie, Werkzeuge, räumliche Situation (Standort, Kommunikation), zeitliche Beschränkung. An dieser Stelle wird darauf verzichtet, die Kostentreiber, ihre Fragestellungen und der jeweilige Wertebereich im Einzelnen vorzustellen. Statt dessen wird hier auf die Literatur, insbesondere das CoCoMo II Manual [107] verwiesen.

Neben den genannten Variablen sieht das CoCoMo Modell eine Konstanten A in der Berechnungsformel vor, die es den Unternehmen erlauben, die Formel bei Bedarf weiter zu kalibrieren. Mit der Konstante A können Multiplikatoreffekte bei wachsender Projektgröße mit berücksichtigt werden.

Bei CoCoMo handelt es sich einerseits um ein Schätzmodell, das als Erfahrungsdatenbasis bei der Aufwandsschätzung herangezogen werden kann. CoCoMo bietet jedoch zusätzlich eine Berechnungsformel zur Anwendung des Schätzmodells und wird so zur vollwertigen Schätzmethode.

A.3 Netzplantechnik

Bei der Netzplantechnik handelt es sich um eine Methode zur Planung von Projekten und zur Überwachung ihrer Ausführung. Die Wurzeln der Netzplantechnik reichen bis in die 50er Jahre zurück. 1970 wurden schließlich die verschiedenen Strömungen und Entwicklungen im Bereich der Netzplantechnik gebündelt und als DIN 69900 normiert. Bestandteil der Normierung sind die Begrifflichkeiten (DIN 69900-1) und Darstellungstechniken (DIN 69900-2) zur Netzplantechnik. Die aktuelle Ausgabe der Norm stammt von 1987, eine Überarbeitung ist in Planung, aber noch nicht abgeschlossen.

Die Netzplantechnik definiert Grundlagen und Vorgehen zur Erstellung von Netzplänen. Ein Netzplan ist die graphische Darstellung von Ablaufstrukturen, die die logische und zeitliche Aufeinanderfolge von Vorgängen veranschaulichen. Elemente eines Netzplans sind Vorgänge, Ereignisse und Beziehungen. Ein *Vorgang* ist eine abgegrenzte Arbeitseinheit, die zu einem bestimmten Zeitpunkt begonnen und einem bestimmten späteren Zeitpunkt beendet wird. In der Projektplanung wird dabei üblicherweise nicht von Vorgängen, sondern von Arbeitspaketen gesprochen. Ein Vorgang wird durch folgende Eigenschaften charakterisiert:

- Vorgangsnummer
- Vorgangsbeschreibung
- Dauer
- Gesamtpuffer
- Frühestmöglicher Anfangszeitpunkt (FAZ): Zeitpunkt, zu dem der Vorgang frühestens beginnen kann.
- Spätestmöglicher Anfangszeitpunkt (SAZ): Zeitpunkt, zu dem der Vorgang spätestens beginnen muss.
- Frühestmöglicher Endzeitpunkt (FEZ): Zeitpunkt, zu dem der Vorgang frühestens beendet werden kann.
- Spätestmöglicher Endzeitpunkt (SEZ): Zeitpunkt, zu dem der Vorgang spätestens beendet sein muss.

Die Werte werden teilweise einem Vorgang zugewiesen, teilweise werden sie berechnet.

Ein *Ereignis* signalisiert das Eintreten eines definierten und beschreibbaren Zustandes im Projektablauf. Ein Ereignis hat keine zeitliche Ausdehnung. Beispiel für ein Ereignis in der Projektplanung ist ein Meilenstein.

Zwischen Vorgängen und Ereignissen bestehen logische Abhängigkeiten hinsichtlich der Ausführungsreihenfolge. Diese werden in der Netzplantechnik in Form von *Anordnungsbeziehungen* beschrieben. Die Netzplantechnik unterscheidet vier Arten von Anordnungsbeziehungen:

- Ende-Start: Ein Vorgang/Ereignis A kann begonnen werden, sobald Vorgang/Ereignis B beendet ist (EA-Beziehung oder Normalfolge)
- Start-Start: Ein Vorgang/Ereignis A kann begonnen werden, sobald Vorgang/Ereignis B begonnen wurde (AA-Beziehung oder Anfangsfolge)
- Start-Ende: Ein Vorgang/Ereignis A kann beendet werden, sobald Vorgang/Ereignis B begonnen wurde (AE-Beziehung oder Sprungfolge)
- Ende-Ende: Ein Vorgang/Ereignis A kann beendet werden, sobald Vorgang/Ereignis B beendet wurde (EE-Beziehung oder Endfolge)

Die Planung erfolgt nun in zwei Schritten: der Strukturanalyse und der Zeitanalyse.

Im Rahmen der Strukturanalyse wird ein Projekt in seine verschiedenen Teilaufgaben (Vorgänge) aufgegliedert. Zu jedem Vorgang werden Vorgangsnummer, Vorgangsbeschreibung und die Dauer des Vorgangs angegeben. Anschließend werden Vorgänge und Ereignisse über Anordnungsbeziehungen in eine logische Reihenfolge gebracht.

Im Rahmen der Zeitanalyse werden nun die zu den Vorgängen noch fehlenden Werte berechnet. Zur Berechnung kennt die Netzplantechnik zwei Verfahren: die Vorwärtsplanung und die Rückwärtplanung. Die Vorwärtsplanung beginnt mit dem Starttermin des ersten Vorgangs und berechnet anhand der Dauer der einzelnen Vorgänge den FAZ sowie den frühestmöglichen FEZ der Vorgänge. Die Rückwärtsplanung beginnt mit dem Endtermin des letzten Vorgangs und berechnet anhand der Dauer den SEZ und den SAZ der Vorgänge. Die Angabe von frühestmöglichen und spätestmöglichen Terminen führt zu Puffern in der Planung. Die Netzplantechnik unterscheidet vier Arten von Puffern:

- Gesamtpuffer: Der Gesamtpuffer eines Vorgangs errechnet sich aus der Differenz von spätesten Anfangszeitpunkt und frühesten Anfangszeitpunkt, sowie spätesten Endzeitpunkt und frühesten Endzeitpunkt. Der Gesamtpuffer gibt an, um wieviel ein Vorgang sich verschieben lässt ohne das Projektende zu gefährden.
- Freier Puffer: Der Freie Puffer ist die Zeit, die den frühestmöglichen Beginn bzw. das frühestmögliche Ende des Nachfolgevorgangs nicht gefährdet. Die Berechnung unterscheidet sich je nach Anordnungsbeziehung. Berechnet wird der freie Puffer bei einer EA Beziehung durch die Differenz vom frühestmöglichen Ende des betrachteten Vorgangs und dem

frühestmöglichen Beginn des Nachfolgevorgangs, bei einer AA Beziehung durch die Differenz der frühestmöglichen Anfangstermine, bei einer EE Beziehung durch die Differenz der frühestmöglichen Endtermine.

- Freier Rückwärtspuffer: Ein freier Rückwärtspuffer ist die maximale Zeitspanne, um die ein Vorgang von seinem frühestmöglichen Anfangszeitpunkt verschoben werden kann. Bedingung ist, dass alle vorhergehenden Vorgänge auf dem spätestmöglichen Termin liegen.
- Unabhängiger Puffer: Ein unabhängiger Puffer ist die maximale Zeitspanne, die ein Vorgang verschoben werden darf, wenn alle vorhergehenden Vorgänge zum spätestmöglichen Termin enden und alle nachfolgenden Vorgänge zum frühestmöglichen Termine beginnen. Die Nutzung des unabhängigen Puffer hat keine Auswirkungen auf die Start und Endtermine der Vorgänger und Nachfolger.

Anhand der Vorwärts- und Rückwärtsplanung kann der so genannte kritische Pfad in einem Netzplan identifiziert werden. Der Kritische Pfad ist nach DIN 69900-1 der Weg von Anfang bis zum Ende des Netzplans, auf dem die Summe aller Pufferzeiten minimal wird. Ereignisse oder Vorgänge, die auf dem Kritischen Weg liegen, erhalten ebenfalls die Bezeichnung kritisch. Bei einem Vorgang, der auf dem kritischen Weg liegt, ist der früheste Anfangszeitpunkt (FAZ) gleich dem spätesten Anfangszeitpunkt (SAZ) und der früheste Endzeitpunkt (FEZ) ist gleich dem spätesten Endzeitpunkt (SEZ). Eine Verlängerung der Vorgangsdauer hat somit unmittelbare Folgen auf die Gesamtprojektdauer.

Zur Netzplantechnik sind unterschiedliche Darstellungsarten möglich. Zu den bekanntesten Darstellungsarten zählen:

- Der Vorgangspfeilnetzplan (VPN): Bei einem Vorgangspfeilnetzplan werden Vorgänge als Pfeile dargestellt. Beginn und Ende von Vorgängen entsprechen Knoten. Die logische Reihenfolge der Vorgänge geht aus der Anordnung der Knoten hervor.
- Der Ereignisknotennetzplan (EKN): Bei einem Ereignisknotenplan werden die Ereignisse als Knoten und die zeitlichen Abhängigkeiten als Pfeile dargestellt.
- Der Vorgangsknotennetzplan (VKN): Bei einem Vorgangsknotennetzplan werden Vorgänge als Knoten dargestellt. Pfeile stellen die Reihenfolgebeziehungen abgeleitet aus den Abhängigkeiten zwischen den Knoten dar.

Spezialisierungen der Netzplanttechnik sind die Critical Path Method (CPM), die Project Evaluation and Review Technique (PERT) und die Metra Potenzial Methode (MPM). Diese Methoden unterscheiden sich in der jeweils gewählten Darstellungstechnik, sowie teilweise bei der Berechnung der Vorgangsdauer. CPM verwendet als Darstellungstechnik den Vorgangspfeilnetzplan. Bei der Ermittlung der Vorgangsdauer wird die häufigste Dauer zu Grunde gelegt. Zur Anwendung gibt

die Methode eine Reihe von Regeln vor. Diese verbieten beispielsweise das Auftreten von Zyklen im Netzplan. CPM wird eingesetzt für Projekte, deren Vorgänge aus Erfahrung anderer Projekte weitgehend bekannt sind und zu denen vergleichsweise wenig Unsicherheit bezüglich der Vorgangsdauer besteht. Die Methode PERT nutzt dagegen als Darstellungstechnik den Ereigniskontennetzplan. Zur Berechnung der Dauer verwendet PERT keine skalaren Werte, statt dessen wird einem Vorgang eine mittlere Dauer zugeordnet. Diese wird berechnet anhand einer optimistisch geschätzte Dauer D_{min}, die häufigste Dauer D_{norm} und die maximale oder pessimistisch geschätzte Dauer D_{max}. Die Berechnung der mittleren Dauer erfolgt entsprechend der Formel:

$$D_{mittel} = \frac{D_{min}+4*D_{norm}+D_{max}}{6}$$

PERT eignet sich zur Planung von Projekten bei denen noch geringe Erfahrung und hohe Unsicherheit bezüglich der zu erwartenden Zeitaufwände besteht. Bei der MPM handelt es sich schließlich um die heute am häufigsten eingesetzten Netzplantechnik. Die Methode verwendet den Vorgangsknotennetzplan als Darstellungsform. Vorteil der MPM ist die vergleichsweise große Menge an Informationen, die über einen Vorgangsknotennetzplan dargestellt werden kann. Eine üblicherweise verwendete Notation für Vorgangskontennetzpläne sind beispielsweise Balkendiagramme, auch als Gantt-Charts bekannt.

A.4 Komponentenbasierte Softwareentwicklung nach Andresen

Bei der Entwurfsmethode von Andresen [5] handelt es sich um eine komponentenbasierte Methode für Entwurf und Entwicklung verteilter, softwareintensiver Systeme. Die Methode geht nach dem Entwurfsansatz der Modell Driven Architecture (MDA) [125] vor. Bei der MDA handelt es sich um einen durchgängigen modellbasierten Ansatz zur Systementwicklung. So sieht MDA die Entwicklung von drei Architekturmodellen im Entwurfsprozess vor:

- **Das Computation Independent Model (CIM):** Ein CIM beschreibt ein System aus Business Sicht. Im CIM sind unter anderem die Anforderungen an das zu entwickelnde System modelliert, sowie die zu Grunde liegenden Geschäftsprozesse und die geplante Einsatzumgebung mit ihren Nachbarsystemen.
- **Das Platform Independent Model (PIM):** Das PIM liefert die logische Architektur des zu entwickelnden Systems. Es beschreibt Software-Komponenten und ihre Interaktionen. Das resultierende Modell ist noch unabängig von einer spezifischen Plattform.
- **Das Platform Specific Model (PSM):** Mit dem PSM wird das PIM an eine spezifische Plattform (z.B. Corba oder J2EE) angepasst. Das PSM dient auf unterster Granularitätsebene als Grundlage für die Codegenerierung. Der resultierende Code kann von Entwicklern entsprechend den Anforderungen verfeinert und angereichert werden.

Ausgehend vom CIM werden alle weiteren Architekturmodelle durch Transformationsschritte erstellt: Das PIM wird durch Transformation aus dem CIM abgeleitet, das PSM durch Transformation aus dem PIM. Jedes Modell kann dabei um weitere Informationen angereichert werden. Ziel des MDA Ansatzes ist die Trennung der Spezifikation von Systemfunktionalität von der Spezifikation pattformspezifischer Eigenschaften. So sind CIM und PIM vollständig plattformunabhängig. Erst mit dem PSM werden Plattformspezifika in die Architekturspezifikation mit aufgenommen. Dies erlaubt die Transformation der fachlichen Architekturmodelle auf unterschiedliche Plattformen. Als Notation für die Beschreibung der verschiedenen Modelle sieht die MDA die UML vor.

Die Methode von Andresen kann als eine konkrete Ausgestaltung des MDA Ansatzes eingeordnet werden. Die Methode sieht vier hierarchisch aufgebaute Modelle als Kernergebnisse vor. Drei der Modelle repräsentiert jeweils genau ein Modell der MDA:

- **Die Business-Architektur als Repräsentant des CIM:** die Business-Architektur ist eine Gesamtsicht aller relevanten Stakeholder auf das System. Inhalte der Business-Architektur sind das Business Konzept, Business Cases, funktionale und nichtfunktionale Anforderungen, Use Cases, Geschäftsprozesse und eine erste logische Komponentenarchitektur.

- **Die Referenz-Architektur als Repräsentant des PIM:** Die Referenz-Architektur dient zur Identifikation und Spezifikation von Komponenenten und Systemen aus fachlicher Sicht. Die Modelle sind unabhängig von den Belangen einer spezifischen Plattform. Inhalte der Referenz-Sicht sind Prozess, Entitäts- und Service-Komponenten, Schnittstellen und Interaktionen, System- und Komponentenhierarchien sowie Nutzungs- und Realisierungsverträge zwischen Komponenten und Systemen.

- **Die Anwendungs-Architektur als Repräsentant des PSM:** Die Anwendungs-Architektur dient der konkreten Abbildung der Referenz-Architektur auf eine spezifische Plattform. Neben der Abbildung der logischen Komponenten auf ein konkretes Komponentenmodell wird die Referenz-Architektur um eine Festlegung der konkreten Tiers, um Spezifikationen der zu integrierenden Systeme mit ihren Daten, sowie um Spezifikationen der Datenbankanbindungen erweitert.

- **Die System-Architektur:** Die Systemarchitektur befasst sich mit der Beschreibung der konkreten Zielplattform. Spezifiziert werden die Infrastruktur, die technische Anbindung an Fremdsysteme, das Laufzeitverhalten der Anwendung und die Deploymentstruktur.

Jedes der genannten Modell ist hierarchisch aus Teilmodellen, so genannten Sichten, aufgebaut. Eine Sicht repräsentiert einen Ausschnitt der Architektur aus einem spezifischen Blickwinkel. Abbildung A.1 zeigt die vier Basismodelle der Methode sowie beispielhaft den hierarchischen Aufbau der Business-Architektur.

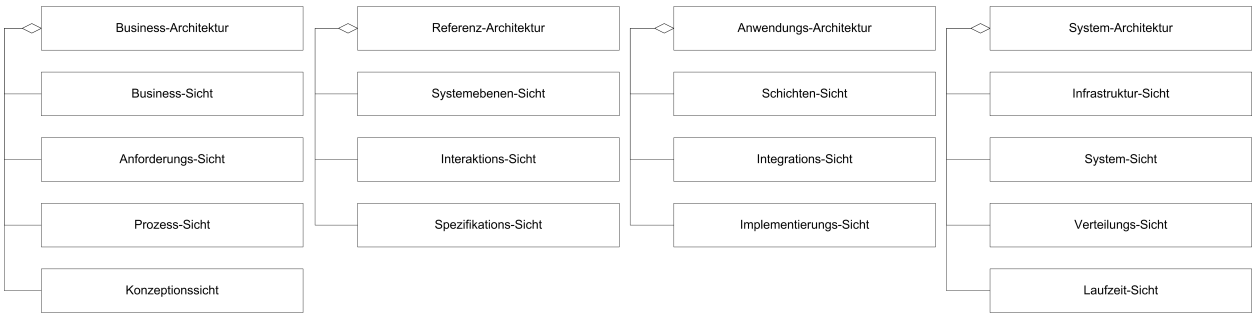

Abbildung A.1: Modelle und Sichten der Methode von Andresen

Die Methode ist vollständig auf die Erarbeitung dieser vier Modelle mit ihren Sichten ausgerichtet. Entsprechend den Vorgaben der MDA beginnt der Systementwurf mit der Erstellung der *Business-Architektur*. Konkret wird eine Business-Sicht mit Business Cases und einem Business-Konzept, eine Anforderungs-Sicht mit den funktionalen und nicht-funktionalen Anforderungen, eine Prozess-Sicht mit den anzupassenden Geschäftsprozessen sowie eine Konzeptions-Sicht mit einem ersten Architekturvorschlag erarbeitet.

Mit der *Referenz-Architektur* wird ein plattformunabhängiger Architekturvorschlag erarbeitet, der das oder die zu entwickelnden bzw. anzupassenden Systeme berücksichtigt. Es werden fachliche Komponenten identifiziert und in der Spezifikations-Sicht ihre Schnittstellen spezifiziert. Fachliche Interaktionen zwischen Komponenten bzw. zwischen Komponenten und Systemen werden in der Interaktions-Sicht beschrieben.

Mit der *Anwendungs-Architektur* wird schließlich die jeweilige Plattform berücksichtigt. Es wird eine der Plattform entsprechende Schichtenarchitektur entworfen, die Integrationstechnologien werden festgelegt und ein Implementierungsmodell auf der Basis von Klassen entworfen.

Die *System-Architektur* sieht schließlich die Spezifikation der verschiedenen Verteilungsaspekte vor. In der Infrastruktur-Sicht werden die zu verwendenden Middleware- und Server-Technologien festgelegt, die System-Sicht beschreibt die Umsetzung nicht-funktionaler Anforderungen wie Sicherheitsaspekte oder Transaktionsverwaltung. In der Verteilungs-Sicht wird die Deploymentstruktur für den Code festgelegt. Das Laufzeitverhalten sowie Anwendungsstart und -stop beschreibt die Laufzeit-Sicht.

Zur Darstellung der verschiedenen Sichten und Modelle sieht die Methode vorrangig die verschiedenen Diagrammtypen der UML 2.0 vor, bei Bedarf ergänzt um OCL Constraints. Zur Beschreibung technischer Aspekte verwendet die Methode freien Text.

Die Methode von Andresen ist eingebettet in ein agiles Vorgehensmodell. Das Vorgehensmodell ist durch ein iteratives Vorgehen gekennzeichnet. Als Phasen, die in einer Iteration zu durchlaufen sind, nennt das Vorgehensmodell: Anforderungen, Architektur, Entwurf, Implementierung, Qualitätssicherung, Test und Verteilung. Diese sind jedoch nur als grobe Richtlinie zu verstehen.

Der Prozess zur Anwendung der Methode wird über so genannte Workflows definiert. Ein Workflow ist ein zusammenhängender Arbeitsablauf, der eine Reihe von Aktivitäten in eine logische Ablauffolge bringen. Ein Workflow modelliert somit einen in sich geschlossenen Teilablauf innerhalb der Methode. Beispielsweise beschreibt der Konzeptions-Workflow den Prozess von der Identifikation der Komponenten-Kandidaten bis hin zum Entwurf einer Referenzarchitektur. Als weitere Workflows nennt die Methode beispielsweise einen Spezifikations-Workflow oder einen Realisierungs-Workflow. Eine Anpassung der Methode auf unterschiedliche Projektbedürfnisse erfolgt durch eine Anpassung der Workflows, ihrer Aktivitäten und der durch sie zu erstellenden Artefakte.

A.5 UfAB

UfAB (Unterlagen für Ausschreibung und Bewertung von IT-Leistungen) ist ein vom Bundesministerium des Inneren herausgegebener Methodenstandard für die Durchführung von Vergabeprojekten im öffentlichen Bereich. Die Methode definierten den groben Ablauf einer Vergabe von der Ausschreibung über die Bewertung der Angebote bis hin zum Vertrag, wobei unterschiedliche Vergabearten und Vergabeverfahren berücksichtigt werden. UfAB ist einer praxisnaher Leitfaden. Die Methode zeigt, wie die in Behörden im Rahmen der Vergabe zu berücksichtigenden Vorschriften (z.B. VOL/A, VOL/B, EVB-IT) sinnvoll angewendet werden können. Die Methode unterscheidet zwei Verfahrensschritte: den Beschaffungsvorlauf und die Durchführung der Beschaffung. Für jeden der Verfahrensschritt werden explizite Teilschritte angegeben.

Beschaffungsvorlauf

Der Beschaffungsvorlauf umfasst alle Aktivitäten, die vor Durchführung einer Beschaffungsmaßnahme zwingend zu erledigen sind. Grundlage für die Durchführung eines neuen IT-Vorhabens ist eine genaue Planung der vorhandenen und benötigten Ressourcen und Mittel. Diese erfolgt mittels einer Ist-Analyse und eines Soll-Konzepts. Das Soll-Konzept beschreibt die jeweiligen organisatorischen, technischen und personellen Bedarfe und gibt alternative Möglichkeiten zur Bedarfsdeckung an. Der Bedarf ist genau zu definieren und an den minimalen Anforderungen zu orientieren.

Anhand des festgestellten Bedarfs wird eine Wirtschaftlichkeitsanalyse durchgeführt. Hier gibt der Standard die in Behörden etablierte Methode zur Wirtschaftlichkeitsbetrachtung, WiBe 4.0, als verpflichtend vor. Wurde die Wirtschaftlichkeit des Vorhabens bestätigt, ist eine Klärung der Haushaltsmittel erforderlich. Zusätzlich sind alle sonstigen einzuhaltenden Vorschriften zu berücksichtigen.

Durchführung der Beschaffung

Mit dem Beschaffungsvorlauf wurde die Notwendigkeit und Durchführbarkeit des Vorhabens geprüft. Nun erfolgt die eigentliche Vergabe. Die im Folgenden beschriebenen Arbeitsschritte geben

die notwendigerweise durchzuführenden Schritte im Vergabeverfahren an. Teilweise hängen Folgeschritte von Zwischenentscheidungen ab, beispielsweise der gewählten Vergabeart oder des gewählten Vergabeverfahrens.

Anlegen der Vergabeakte: Erster Schritt zur Durchführung einer Vergabe ist die Erstellung einer Vergabeakte, in die lückenlos alle Entscheidungen und wichtige Dokumente eingehen. Die Vergabeakte dient der vollständigen Dokumentation des Vergabeprozesses.

Schätzung des Auftragswerts: Der Auftragswert wird anhand von Bedarfsanalysen, auf der Basis eigener Erkenntnisse oder anhand der Ergebnisse der WiBe ermittelt.

Festlegung des Vergabeverfahren: Anhand des ermittelten Auftragswerts wird darüber entschieden, ob ein nationales oder ein EU-weites Verfahren angewendet wird.

Festlegung der Vergabeart: Je nach gewählten Verfahren muss eine entsprechende Vergabeart gewählt werden. Für nationale Verfahren sind folgende Vergabearten möglich: Öffentliche Ausschreibung, Beschränkte Ausschreibung, Freihändige Ausschreibung. Im EU-weiten Verfahren gibt es dagegen: Offenes Verfahren, Nichtoffenes Verfahren, Verhandlungsverfahren. Die Auswahlkriterien für die jeweiligen Vergabearten und Vergabeverfahren sind explizit in der Methode festgelegt.

Erstellung eines Zeitplans: Abhängig von Vergabeart und Vergabeverfahren wird eine Zeitplan für die Vergabe erstellt. Im Zeitplan müssen alle jeweils geltenden gesetzlich geregelten Fristen berücksichtigt werden.

Durchführung des Verfahrens: Nachdem die Rahmenbedingungen festgelegt wurden, kann die Ausschreibung vorbereitet und durchgeführt werden. Das Ausschreibungsverfahrens unterscheidet sich je nach gewählter Vergabeart und gewähltem Vergabeverfahren. Die Methode gibt eigenständige Verfahren für folgende Vergabetypen vor:

- Nationales Verfahren: Öffentliche Ausschreibung
- Nationale Verfahren: Beschränkte Ausschreibung mit Teilnahmewettbewerb
- Nationale Verfahren: Beschränkte Ausschreibung ohne Teilnahmewettbewerb
- Nationale Verfahren: Freihändige Vergabe mit Teilnahmewettbewerb
- Nationale Verfahren: Freihändige Vergabe ohne Teilnahmewettbewerb
- EU-weites Verfahren: Offenes Verfahren
- EU-weites Verfahren: Nichtoffenes Verfahren mit Teilnamewettbewerb
- EU-weites Verfahren: Nichtoffenes Verfahren ohne Teilnamewettbewerb
- EU-weites Verfahren: Verhandlungsverfahren mit Teilnamewettbewerb
- EU-weites Verfahren: Verhandlungsverfahren ohne Teilnamewettbewerb

Auf die Durchführung der einzelnen Vergabetypen sowie auf ihre Unterschiede, wird an dieser Stelle nicht weiter eingegangen. Die entsprechenden Informationen können in [74] nachgelesen werden.

Versand der Verdingungsunterlagen: Je nach gewähltem Vergabetyp wurden spezifische Verdingungsunterlagen erstellt. Diese werden nun in doppelter Ausfertigung den Bietern übermittelt. Unterlagen, die nicht vervielfältigt werden können, müssen zur Einsichtnahme zur Verfügung gestellt werden. Geklärt ist ausserdem die Behandlung von Bewerberfragen, sowie das Vorgehen zur Angebotsöffnung.

Bewertung der Angebote: Zur Bewertung der Angebot wird ein vierstufiges Verfahren gefordert: formale Prüfung, Eignungsprüfung der Bieter, Prüfung der Angemessenheit des Preises, Wirtschaftlichkeitsprüfung. Die Stufen müssen in dieser Reihenfolge durchgeführt werden. Sie dürfen nicht vermischt oder ausgetauscht werden. Ziel der Angebotsbewertung ist die Ermittlung des wirtschaftlichsten Angebots.

Zuschlagsentscheidung: Nach Ermittlung des wirtschaftlichsten Angebots wird eine Zuschlagsentscheidung getroffen und der entsprechende Bieter informiert. Die Gründe für die Entscheidung, sowie die Gründe für die Ablehnung der restlichen Bieter werden in der Vergabeakte dokumentiert. Die abgelehnten Bieter werden ebenfalls über das Ergebnis informiert.

Vertragserstellung: Der Inhalt der Vertragsurkunde ergibt sich im Wesentlichen aus den Verdingungsunterlagen und dem Angebot. Die Unterzeichnung der Vertragsurkunde durch die Vertragspartner dient der Beweissicherung.

Das hier skizzierte Verfahren soll das Vorgehen zur Vergabe nach UfAB verdeutlichen. Die Methode definiert zusätzliche Vorgaben (als Module bezeichnet), die aus den zu berücksichtigenden gesetzlichen Vorgaben abgeleitet wurden. Die Module sind jeweils den einzelnen Verfahrensschritten zugeordnet. Sie enthalten methodische Vorgaben, gesetzliche Vorgaben oder auch allgemeine Hinweise, die bei der Durchführung der Verfahrensschritte zu beachten sind.

A.6 Focus

Bei Focus handelt es sich um eine formale Entwurfsmethode zur Spezifikation und schrittweisen Entwicklung von komponentenbasierten, interaktiven Systemen. Das Verhalten einer Komponenten wird ausschließlich über den Strom der an der Komponentenschnittstelle ausgetauschten Nachrichten spezifiziert. Konkret unterstützt die Methode die Spezifikation der Nachrichtenströme zwischen den Komponenten eines Systems, sowie zwischen dem System selbst und seiner Umgebung. Kernkonzepte zur Spezifikation sind Eingabe- und Ausgabekanäle sowie Ströme von Nachrichten, die auf den Eingabe- und Ausgabekanälen verschickt werden.

Die Methode deckt prinzipiell den gesamten Entwicklungsprozess ausgehend von der Anforderungsspezifikation bis hin zu einer Systemspezifikation in Form einer abstrakten Programmierung ab. Der Prozess selbst stützt sich auf das Spiralmodell von Boehm. Jede Iteration im Spiralmodell ent-

spricht einem Verfeinerungsschritt der Systemspezifikation. So geht der Entwicklungsprozess von einer formalen Spezifikation der Anforderungen aus, die in iterativen Schritten zum System und zu Komponenten verfeinert werden. Ergebnis des iterativen Spezifikationsprozesses ist eine Spezifikation des Systems in einer abstrakten Programmiersprache, die unmittelbar in eine konkrete Programmiersprache übersetzt werden kann. Kern der Methode sind eine Menge von Spezifikationstechniken, auch Stile genannt, die je nach Bedarf alternativ eingesetzt werden können:

- Der *Equational Style* beschreibt das Systemverhalten in Form von Gleichungen. Diese Form der Spezifikation eignet sich insbesondere gegeben Ende des Spezifikationsprozesses als Basis der Codegenerierung.
- Der *Assumption/Guarantee Style* erlaubt die Spezifikation von Komponenten oder Systemen über ihre Verhalten an der Schnittstelle. Assumptions entsprechen Vorbedingungen, die Komponente oder Umgebung erfüllen müssen. In diesem Fall wird das über die Guarantee Phrase spezifizierte Verhalten der Komponente garantiert.
- Der *Graphical Style* erlaubt eine graphische Strukturierung der Spezifikationen über Tabellen und Diagramme. Diese graphischen Stilmittel steuern selbst keinerlei Semantik zur Spezifikation bei, sie unterstützen jedoch die Lesbarkeit der Spezifikationen.
- Der *Relational Style* unterstützt die Modellierung des Komponentenverhaltens über die Spezifikation der Beziehungen zwischen Eingabe- und Ausgabeströmen an der Schnittstelle.

Die Anwendung der Spezifikationsstile im Entwurfsprozess wird von der Methode nicht explizit vorgegeben. Dem Anwender wird mit Focus vielmehr eine formal fundierte Domänentheorie sowie ein ein Baukasten an Spezifikationstechniken zur Entwicklung konkreter Entwurfsmethoden zur Verfügung gestellt.

Anhang B

Das V-Modell XT

Im Folgenden wird das V-Modell XT mit seinen Kernkonzepten und mit seinem Metamodell vorgestellt. Ziel ist es, eine auf die Ziele dieser Arbeit ausgerichtete kompakte Einführung in Modell und Metamodell sowie eine Erläuterung der zentralen Ideen.

Als generisches Vorgehensmodell ist das V-Modell XT, im folgenden kurz als V-Modell bezeichnet, methodenneutral und unabhängig von spezifischen Organisationskontexten. Es definiert einen durchgängigen Prozess zur risikominimierten, qualitätsgesicherten Projektdurchführung. Das V-Modell unterliegt seit seiner Veröffentlichung einem kontinuierlichen Pflege- und Weiterentwicklungsprozess. Aktuelles Release (Stand Juli 2007) und Grundlage dieser Arbeit ist das V-Modell 1.2.

Das V-Modell deckt in seinen Beschreibungen eine breite Palette an Teilprozessen ab, die typischerweise in Entwicklungsprojekten Anwendung finden. Die zur Beschreibung verwendeten Modellierungskonzepte sind *Projekttypen*, *Vorgehensbausteine* und *Projektdurchführungsstrategien*.

B.1 Projekttypen

Ein *Projekttyp* ist das Modell einer Menge von Projekten mit ähnlichen Eigenschaften. Er repräsentiert einen in sich geschlossenen und konsistenten Ausschnitt aus dem V-Modell und kann auch als Vorgehensmodell für Projekte des entsprechenden Typs betrachtet werden. Das V-Modell unterscheidet vier Projekttypen:

- **Systementwicklungsprojekt (AG)**: Dieser Projekttyp modelliert die Durchführung von Vergabeprojekten auf Seiten des Auftraggebers. Zentrale Elemente des Projekttyps sind die Anforderungsfestlegung, die Ausschreibung, die Angebotsbewertung, die Projektbegleitung und schließlich die Abnahme.

- **Systementwicklungsprojekt (AN)**: Dieser Projekttyp modelliert das Systementwicklungsprojekt zu einem Vergabeprojekt auf Auftragnehmerseite. Zentrale Elemente des Projekttyps sind die Angebotserstellung, die Systementwicklung und die Lieferung.
- **Systementwicklungsprojekt (AG/AN)**: Dieser Projekttyp modelliert die Durchführung von Systementwicklungsprojekten ohne explizite Vergabe. Auftraggeber und Auftragnehmer führen ein gemeinsames Projekt durch. Der Projekttyp verbindet Ausschreibung und Systementwicklung in einem Projekt und unterstützt keine Ausschreibung- und Angebotsaktivitäten.
- **Organisationsspezifische Einführung eines Vorgehensmodells**: Dieser Projekttyp modelliert Projekte zur Einführung eines Vorgehensmodells in Organisationen. Der Projekttyp kann auf das V-Modell selbst angewandt werden, ist jedoch nicht darauf beschränkt. Im Kern handelt es sich um einen Projekttyp zur Prozesseinführung und kontinuierlichen Prozessverbesserung in einer Organisation.

Die Unterstützung von Vergabeprojekten ist ein Alleinstellungsmerkmal des V-Modells innerhalb der heute verfügbaren Vorgehensmodelle. Im Rahmen einer Vergabe beauftragt ein Auftraggeber die Entwicklung eines Systems durch einen oder mehrere Auftragnehmer. Die einzelnen Schritte in einem Vergabeprojekt sind:

1. Die Ausschreibung des Systems durch den Auftraggeber auf Basis der Anforderungen,
2. die Erstellung eines Angebots zur Ausschreibung durch mehrere Bieter,
3. die Beauftragung eines geeigneten Auftragnehmers auf Basis des Angebots,
4. die Projektbegleitung aus Sicht des Auftraggebers,
5. die Projektdurchführung aus Sicht des Auftragnehmers,
6. die Lieferung des entwickelten Systems durch den Auftragnehmer,
7. die Abnahme des beauftragten Systems durch den Auftraggeber.

Der tatsächliche Entwicklungsprozess findet im V-Modell bei Vergabeprojekten ausschließlich auf Seiten des Auftragnehmers statt. Der Auftraggeber ist jedoch für die Bereitstellung der entsprechenden Anforderungen verantwortlich. Auftraggeberprojekt und Auftragnehmerprojekt einer Vergabe sind im V-Modell als unabhängige Projekttypen modelliert. Die entsprechenden Projekte werden immer parallel durchgeführt.

Ein Projekttyp gibt eine Menge von *Vorgehensbausteinen* und eine Menge von *Projektdurchführungsstrategien* vor. Welche Vorgehensbausteine und Projektdurchführungsstrategien ein Projekttyp umfasst, orientiert sich an den Teilprozessen, die in entsprechenden Projekten üblicherweise Anwendung finden. Im Folgenden werden diese Konzepte im Detail vorgestellt.

B.2 Vorgehensbausteine

Ein Vorgehensbaustein ist ein inhaltlich geschlossenes, wiederverwendbares Modul im V-Modell. Er fasst alle Produkte, Rollen und Aktivitäten zusammen, die zur Modellierung eines spezifischen Teilprozesses von Bedeutung sind. Das V-Modell in Version 1.2 kennt 21 Vorgehensbausteine. Mit ihnen werden die meisten der typischen Teilprozesse in Entwicklungsprojekt abgedeckt.

Abbildung B.1 zeigt die Vorgehensbausteine im V-Modell nach inhaltlichen Gesichtspunkten gruppiert. Wie die Abbildung illustriert, reicht das Spektrum des V-Modells inhaltlich von allgemeinen querschnittlichen Managementaufgaben über den Entwurfs- und Entwicklungsprozess bis hin zum Vergabeprozess und zu vielen weiteren Teilprozessen. Bei den Vorgehensbausteinen selbst handelt es sich um reine Strukturierungskonzepte. Im Gegensatz zu Produkten, Aktivitäten und Rollen spielen sie für die konkrete Projektdurchführung keine Rolle. Im Folgenden werden die Vorgehensbausteine im Überblick vorgestellt. Die Beschreibung orientiert sich an den in Abbildung B.1 eingeführten inhaltlichen Gruppen.

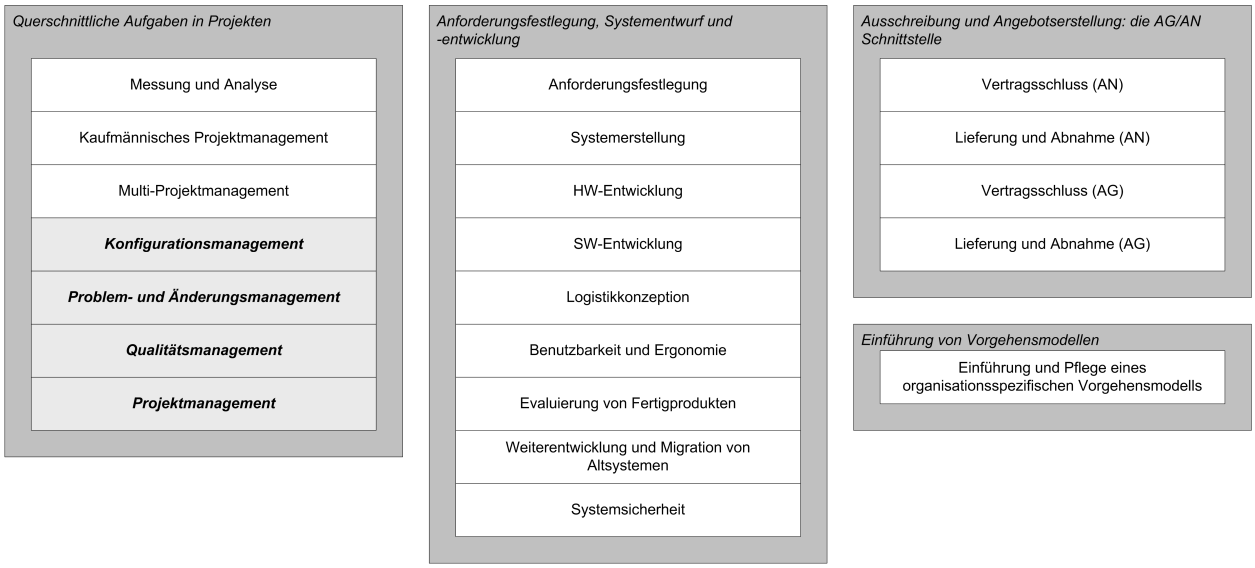

Abbildung B.1: Vorgehensbausteine im V-Modell

Querschnittliche Aufgaben

Vier Vorgehensbausteine (in der Graphik grau hinterlegt) spielen eine herausragende Rolle im V-Modell. Sie werden auch als der 'V-Modell Kern' bezeichnet. Hierbei handelt es sich um die Vorgehensbausteine *Projektmanagement*, *Qualitätssicherung*, *Problem- und Änderungsmanagement* und *Konfigurationsmanagement*. Diese Vorgehensbausteine decken mit ihren Produkten, Aktivitäten und Rollen querschnittliche Aufgaben ab, die in jedem Projekt nach entsprechenden Vorgaben durchzuführen sind, um einen definierten Standard in der Qualität der Projektdurchführung zu erreichen.

Der Vorgehensbaustein *Projektmanagement* deckt den Teilprozess der Projektsteuerung und Projektplanung ab. Produkte im Vorgehensbaustein sind beispielsweise der Projektplan, das Projekthandbuch oder die Risikoliste. Als Rollen nennt der Vorgehensbaustein unter anderem den Projektleiter. Dieser hat die Aufgabe, anhand der entsprechenden Aktivitäten verantwortlich die Produkte zu erstellen. Der Vorgehensbaustein enthält eine Vielzahl weiterer Produkte mit ihren Aktivitäten und Rollen, die alle in irgendeiner Form den Teilprozess der Projektplanung und -steuerung in einem Projekt unterstützen.

Neben Projektmanagement spielt vor allem die Qualitätssicherung eine zentrale Rolle in Projekten. Im Vorgehensbaustein *Qualitätssicherung* finden sich entsprechende Produkte, Aktivitäten und Rollen, die zur Durchführung eines geeigneten Qualitätssicherungsprozesses in Projekten erforderlich sind. Zu nennen sind hier beispielsweise das QS-Handbuch und der QS-Bericht als Produkte. Im QS-Handbuch werden alle für ein Projekt relevanten Maßnahmen hinsichtlich des Qualitätssicherungsprozesses dokumentiert, wie beispielsweise Vorgaben bezüglich der im Projekt durchzuführenden Tests. Diese Vorgaben gelten als verbindlich für das Projekt. Im QS-Bericht werden dagegen die tatsächlich durchgeführten Qualitätssicherungsmaßnahmen protokolliert und in Berichtsform zusammengefasst. Verantwortliche Rolle für diesen Teilprozess und damit für die Erstellung der Produkte ist der QS-Verantwortliche.

Die Vorgehensbausteine *Konfigurationsmanagement* und *Problem- und Änderungsmanagement* modellieren in ähnlicher Weise die ihnen entsprechenden Teilprozesse über Produkte, Aktivitäten und Rollen. Produkte im Konfigurationsmanagement sind beispielsweise die Produktbibliothek als Dokumentenbasis eines jeden Projekts sowie die Konfiguration zur Kennzeichnung zusammengehöriger konsistenter Produktstände innerhalb der Produktbibliothek. Das Problem- und Änderungsmanagement beschäftigt sich dagegen mit der Verwaltung von Problem- und Änderungsanträgen, wie sie beispielsweise als Reaktion auf Anforderungsänderungen oder Fehlermeldungen in einem Projekt auftreten. Die Anträge müssen bewertet und ein Entschluss gefasst und dokumentiert werden. Die Verwaltung der Änderungsanträge findet im Produkt Änderungsstatusliste statt.

Als optionale Ergänzung zum Vorgehensbaustein Projektmanagement bietet das V-Modell zwei weitere Vorgehensbausteine an, *kaufmännisches Projektmanagement* und *Multiprojektmanagement*. Während sich der Vorgehensbaustein kaufmännisches Projektmanagement mit dem Thema Projektcontrolling befasst, unterstützt der Vorgehensbaustein Multiprojektmanagement die Durchführung mehrerer parallel ablaufender Vergabeprojekte aus Sicht eines Auftraggebers. Der Vorgehensbaustein *Messung und Analyse* befasst sich schließlich mit der Erfassung und Auswertung von Metriken in Projekten, aber auch innerhalb der gesamten Organisation.

Ausschreibung und Angebotserstellung

Neben den querschnittlichen Aufgaben unterstützt das V-Modell einen expliziten Vergabe- sowie Abnahmeprozess. Bei einer Vergabe erstellt ein Auftraggeber eine Ausschreibung. Potentielle Auf-

tragnehmer können auf die Ausschreibung hin ein Angebot abgeben. Die eingegangenen Angebote werden geprüft und einer der Anbieter erhält den Zuschlag. Vergabeprojekte berücksichtigen neben der Ausschreibung auch die Projektbegleitung aus Sicht des Auftraggebers sowie die Abnahme des gelieferten Systems.

Modelliert wird dieser Prozess im V-Modell über die Vorgehensbausteine *Vertragsschluss* sowie *Lieferung und Abnahme*. Diese Vorgehensbausteine liegen je zweimal im V-Modell vor, einmal für den Auftraggeber (Vertragsschluss (AG), Lieferung und Abnahme (AG)), einmal für den Auftragnehmer (Vertragsschluss (AN), Lieferung und Abnahme (AN)). Sie enthalten die jeweils für den Auftraggeber bzw. den Auftragnehmer relevanten Produkte, Aktivitäten und Rollen im Vergabeprozess. Auf Auftraggeberseite sind dies beispielsweise die Produkte Ausschreibung, Angebotsbewertung und Abnahmeprotokoll. Als verantwortliche Rolle nennt das V-Modell den Ausschreibungsverantwortlichen (AG). Auf Auftragnehmerseite finden sich dagegen die Produkte Ausschreibungsbewertung, Angebot und Lieferung. Verantwortliche Rolle hier ist der Akquisiteur bzw. der Projektleiter.

Anforderungserfassung, Systementwurf und -entwicklung

Der gesamte Prozess zur Erstellung eines neuen Softwaresystems kann nach den drei Bereichen Anforderungserfassung, Systementwurf und Systementwicklung unterteilt werden. Bei Vergabeprojekten findet die Anforderungserfassung ausschließlich auf Auftraggeberseite statt, Systementwurf und -entwicklung liegen dagegen in der Verantwortung des Auftragnehmers. Zentrales Produkt im Vorgehensbaustein *Anforderungserfassung* ist das Lastenheft (Anforderungen (Lastenheft)). Im Lastenheft werden alle Anforderungen an das System dokumentiert. Es dient als Grundlage der Ausschreibung.

Der Vorgehensbaustein *Systemerstellung* enthält als zentrales Produkt die Gesamtsystemspezifikation (Pflichtenheft). Der Vorgehensbaustein steht im Zentrum eines jeden Auftragnehmer-Entwicklungsprojekts und stellt das direkte Gegenstück zum Vorgehensbaustein *Anforderungserfassung* dar. Die Anforderungen im Lastenheft werden nach der Projektbeauftragung im Pflichtenheft übernommen und verfeinert. Das Pflichtenheft dient so als Basis des gesamten Entwurfs- und Entwicklungsprozesses.

Neben dem Pflichtenheft umfasst der Vorgehensbaustein alle weiteren Produkte, die für das Grobdesign des zu entwickelnden Systems benötigt werden, wie beispielsweise die Systemarchitektur und die Systemspezifikation. Das Feindesign und die Implementierung des Systems wird dagegen durch die Vorgehensbausteine *SW-Entwicklung* und *HW-Entwicklung* abgedeckt. Jeder der beiden Vorgehensbausteine definiert entsprechende Design- und Entwicklungsprodukte, wobei auf dieser Ebene klar zwischen Produkten für Hardware- oder Softwareentwicklung unterschieden wird.

Zur Abdeckung von Teilprozessen, die nicht für jedes Projekt und jedes System eine Rolle spielen, jedoch in bestimmten Fällen benötigt werden, stellt das V-Modell eine Reihe weiterer Vorgehensbausteine zur Verfügung. Der Vorgehensbaustein *Benutzbarkeit und Ergonomie* beschäftigt sich mit

dem Design von Benutzerschnittstelle aus ergonomischer und anwendungsspezifischer Sicht. Produkte im Vorgehensbaustein sind ein Styleguide für das Design der Benutzerschnittstelle sowie eine Anwenderaufgabenanalyse. Der Vorgehensbaustein *Evaluierung von Fertigprodukten* wird benötigt, wenn die Entscheidung getroffen wurde, Fertigprodukte einzukaufen, die in das zu entwickelnde System mit integriert werden. Mit Aspekten zur Sicherheit beschäftigt sich der Vorgehensbaustein *Systemsicherheit*. Er liefert unter anderem Produkte zur Risikoakzeptenz und Risikobewertung. Die Ergebnisse können sich direkt auf das Systemdesign auswirken. Nicht immer handelt es sich bei einem Entwicklungsprojekt tatsächlich um eine reine Neuentwicklung. Häufig wird die Weiterentwicklung eines Systems beauftragt, was unter Umständen Migrationen von Daten und Funktionalität erforderlich macht. Für den entsprechenden Teilprozess liefert das V-Modell den Vorgehensbaustein *Weiterentwicklung und Migration* mit den Produkten Altsystemanalyse und Migrationskonzept.

Einführung von Vorgehensmodellen

Das V-Modell definiert einen eigenständigen Einführungs- und Pflegeprozess für Vorgehensmodelle. Dazu stellt es den Vorgehensbaustein *Einführung eines organisationsspezifischen Vorgehensmodells* mit den entsprechenden Produkten, Aktivitäten und Rollen zur Verfügung. Dieser unterstützt eine Analyse der Ist-Situation bezüglicher des Entwicklungsprozesses in der Organisation, eine Konzeption der geänderten Prozesse auf Basis der identifizierten Probleme und Ziele sowie die tatsächliche Einführung eines neuen Vorgehensmodells. Entsprechende Produkte sind die Bewertung eines Vorgehensmodells, das Verbesserungskonzept für ein Vorgehensmodell und als Ergebnis das organisationsspezifische Vorgehensmodell selbst. Verantwortliche Rolle ist jeweils der Prozessingenieur.

B.3 Projektdurchführungsstrategien

In einem Vorgehensmodell werden Projektabläufe über ein Ablaufmodell modelliert (vgl. Abschnitt 3.1). Das Ablaufmodell im V-Modell stützt sich auf die Konzepte Entscheidungspunkt und Projektdurchführungsstrategie. Ein Entscheidungspunkt modelliert einen Meilenstein in einem Projekt, zu dem über den Projektfortschritt entschieden wird. Grundlage der Entscheidung ist einerseits die ermittelte Qualität der zum Entscheidungspunkt vorzulegenden Produkte, andererseits können sich auch die in den Produkten erarbeiteten Inhalte auf die Entscheidung auswirken.

Eine zulässige Reihenfolge über den zu durchlaufenden Entscheidungspunkten wird über Projektdurchführungsstrategien festgelegt. Eine Projektdurchführungsstrategie fasst eine Menge von Entscheidungspunkten zusammen und beschreibt die Menge der möglichen Pfade durch die Entscheidungspunkte. Ein Projekt wählt zu Projektbeginn genau einen der zulässigen Pfade innerhalb der Projektdurchführungsstrategie und erhält so einen initialen Meilensteinplan. Dieser dient als Basis der weiteren Projektplanung.

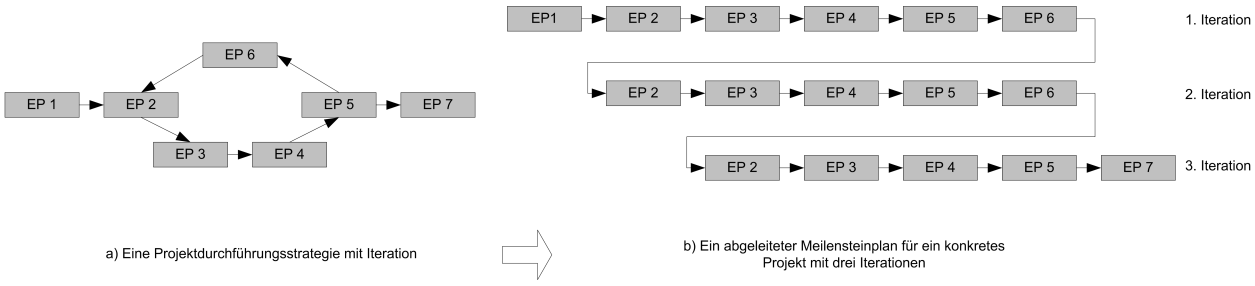

Abbildung B.2: Ableitung eines Meilensteinplans aus einer Projektdurchführungsstrategie

Abbildung B.2 stellt den Zusammenhang zwischen Projektdurchführungsstrategie und Meilensteinplan graphisch dar. Die Projektdurchführungsstrategie (Abbildung B.2 (a)) legt die Menge der Entscheidungspunkte und die erlaubte Durchlaufreihenfolgen fest. Sie dient als Vorlage für eine Menge von Meilensteinplänen. Die Menge aller aus dieser Projektdurchführungsstrategie ableitbaren Meilensteinpläne ist festgelegt durch die Anzahl möglicher Durchläufe. Variationen ergeben sich an Verzweigungen. Eine Verzweigung in einer Projektdurchführungsstrategie tritt immer dann auf, wenn es zu einem Entscheidungspunkt mehrere mögliche Nachfolgeentscheidungspunkte gibt. So hat im Beispiel der Entscheidungspunkt EP 5 zwei mögliche Nachfolger, die Entscheidungspunkte EP 6 und EP 7.

Der Meilensteinplan in Abbildung B.2 (b) stellt genau einen der möglichen Durchläufe durch die Projektdurchführungsstrategie dar. Zu jeder Verzweigung in der Projektdurchführungsstrategie wurde die Entscheidung für genau einen konkreten Nachfolgeentscheidungspunkt getroffen. So wurde im Beispiel für den Entscheidungspunkt EP 5 zweimal der Entscheidungspunkt EP 6 als Nachfolger gewählt und so ein dreistufiger iterativer Ablauf für das Projekt festgelegt. Die Entscheidung für den Entscheidungspunkt EP 7 am Ende der dritten Iteration legt schließlich das Ende des Projekts fest.

Das V-Modell kennt verschiedene Projektdurchführungsstrategien für jeden Projekttyp, wobei teilweise auch Überschneidungen zwischen den Projektdurchführungsstrategien auftreten können. Im Folgenden werden alle im V-Modell definierten Projektdurchführungsstrategien vorgestellt. Zur Vermeidung von redundanten Beschreibungen, die auf Grund der inhaltlichen Überschneidungen existieren, wurde eine nach Teilabläufen geordnete Beschreibungsstruktur für die Projektdurchführungsstrategien gewählt.

Projektdurchführungsstrategien des Auftraggebers

Für den Projekttyp Systementwicklungsprojekt (AG) bietet das V-Modell zwei Projektdurchführungsstrategien an: die Projektdurchführungsstrategie *Vergabe und Durchführung eines Systementwicklungsprojekts* unterstützt die Vergabe von einem Entwicklungsprojekt an genau einen Auftragnehmer, bei der Projektdurchführungsstrategie *Vergabe und Durchführung mehrerer Systementwicklungsprojekt* wird dagegen die Vergabe eines Entwicklungsauftrags auf mehrere Anbieter

verteilt. Jeder Anbieter erhält den Zuschlag für die Entwicklung eines Teilsystems in einem eigenständigen Teilprojekt. Eines der Teilprojekte hat die Aufgabe, alle gelieferten Teilsysteme zu einem Gesamtsystem zusammenzuführen.

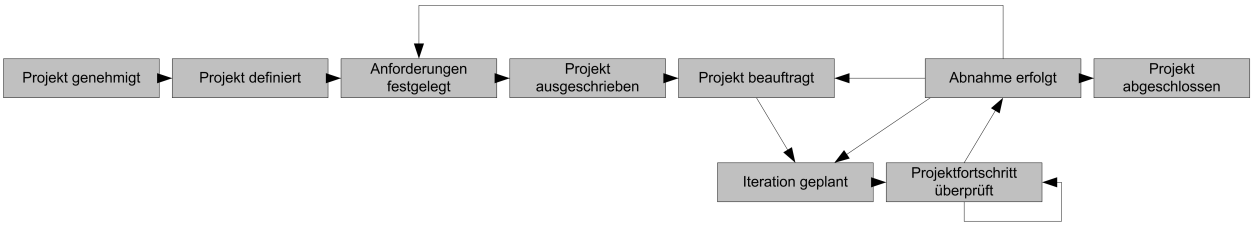

Abbildung B.3: Projektdurchführungsstrategie *Vergabe und Durchführung eines Systementwicklungsprojekts*

Abbildung B.3 zeigt beispielhaft die Projektdurchführungsstrategie zur Vergabe eines Systementwicklungsprojekts. In der Projektdurchführungsstrategie ist jeder Entscheidungspunkt durch einen eindeutigen Namen gekennzeichnet. Der Name charakterisiert das Ende eines spezifischen Projektabschnitts bzw. einer Projektphase und kennzeichnet, welche der in dieser Phase erarbeiteten Produkte vorzulegen sind. Zum Entscheidungspunkt *Anforderungen festgelegt* ist beispielsweise das Produkt *Lastenheft* vorzulegen, zum Entscheidungspunkt *Projekt ausgeschrieben* dagegen das Produkt *Ausschreibung*. Zu jedem Entscheidungspunkt sind im V-Modell die jeweils vorzulegenden Produkte eindeutig festgelegt.

Projektdurchführungsstrategien des Auftragnehmers

Ein Auftragnehmerprojekt läuft immer parallel zu einem Auftraggeberprojekt. Diese Parallelität spiegelt sich in allen Projektdurchführungsstrategien des Projekttyps.

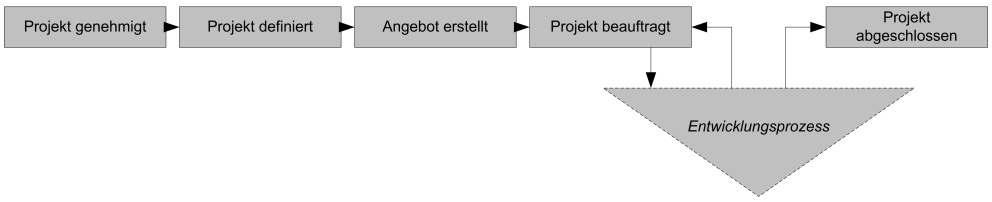

Abbildung B.4: Allgemeiner Teil einer Projektdurchführungsstrategie für Auftragnehmer

Abbildung B.4 zeigt den allen Auftragnehmerstrategien einheitlichen übergeordneten Ablauf. Getriggert wird ein Auftragnehmerprojekt immer durch die Entscheidung einer Organisation, zu einer Ausschreibung ein Angebot zu erstellen. Der Projektbeginn verläuft identisch zu einem Auftraggeberprojekt. Das Projekt wird genehmigt und definiert. Hat ein Auftragnehmer den Zuschlag erhalten, beginnt der eigentliche Entwicklungsprozess. Mit der Lieferung und Abnahme eines (Teil-)Systems endet eine Iteration. Das Projekt kann über eine weitere Iteration fortgesetzt werden oder es wird beendet.

Der Abschnitt in den Auftragnehmerstrategien, der den Entwicklungsprozess modelliert, orientiert sich in seinem Ablauf an den im 'ursprünglichen' V-Modell (vgl. Abbildung 8.1) vorgegebenen Phasen. Auftragnehmerprojekte haben die Möglichkeit, dass 'V' im V-Modell je nach Bedarf in unterschiedlicher Art und Weise zu durchlaufen. Abbildung B.5 stellt die drei verfügbaren Möglichkeiten graphisch nebeneinander dar.

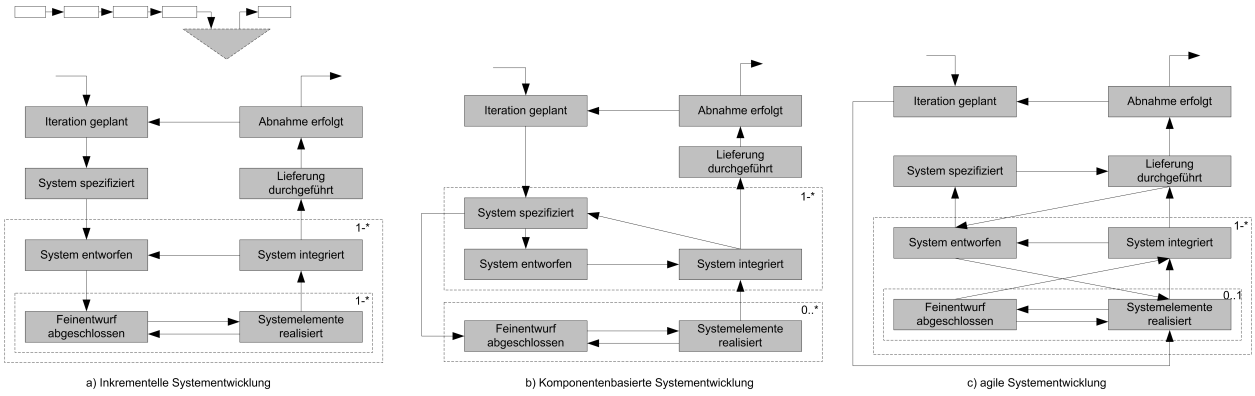

Abbildung B.5: Varianten im Entwicklungsprozess bei Projektdurchführungsstrategien

Die Projektdurchführungsstrategie *Inkrementelle Systementwicklung* (Abbildung B.5 (a)) modelliert einen typischen Entwicklungsablauf, ausgehend vom Grobdesign über das Feindesign bis hin zu Entwicklung und Integration. Inkrementelle Systementwicklung entspricht einer sequentiellen Abfolge der Entwicklungstätigkeiten vom Systementwurf über die Implementierung bis hin zur Abnahme. Die Entwicklungstätigkeiten können in mehreren Iterationen durchlaufen werden. Durch die Möglichkeit der Iterationen erlaubt die Projektdurchführungsstrategie zur Inkrementellen Systementwicklung eine stufenweisen Entwicklung des Systems: ausgehend von einer Teilmenge der wichtigsten Anforderungen, werden in jeder Stufe (Iteration) weitere Anforderungen ausgewählt und umgesetzt, bis alle Anforderungen berücksichtigt wurden und das System fertig gestellt ist.

Die Projektdurchführungsstrategie *Komponentenbasierte Entwicklung* (Abbildung B.5 (b)) modelliert einen Projektablauf, der vor allem den verstärkten Einsatz von Fertigprodukten unterstützt. Während die Entscheidungspunkte in der Projektdurchführungsstrategie selbst identisch zu den Entscheidungspunkten der Inkrementellen Systementwicklung sind, erlaubt die Projektdurchführungsstrategie zur komponentenbasierten Entwicklung alternative Wege durch die Entscheidungspunkte. So hat ein Projekt, dass sich in seiner Projektplanung auf diese Projektdurchführungsstrategie stützt die Möglichkeit bei Bedarf vollständig auf Entwicklungsaktivitäten zu verzichten. Im Fokus steht dagegen die Integration fertiger Komponenten auf Basis einer Systemarchitektur. Dies kann beispielsweise bei reinen Integrationsprojekten sinnvoll sein.

Eine Umkehrung der Reihenfolge von Entwurfs- und Entwicklungstätigkeiten unterstützt die Projektdurchführungsstrategie *Agile Systementwicklung* (Abbildung B.5 (c)). Ihr Ziel ist die Unterstützung von Projekten, in denen es beispielsweise auf Grund der Verwendung neuer, noch unbekannter Technologien nicht ratsam ist, ohne konkrete Implementierungserfahrung einen tragfähigen

Systementwurf zu erstellen. Ziel der Projektdurchführungsstrategie ist es, den Projekten gewisse Freiheiten bei der Entwicklung zu lassen und Entwicklungs- und Entwurfstätigkeiten je nach Bedarf zu vermischen.

Das V-Modell unterstützt nicht nur die Entwicklungsphase eines Systems. Die Projektdurchführungsstrategie zur *Wartung und Pflege von Systemen* wurde speziell zur Durchführung von Wartungsprojekten in das V-Modell eingeführt. Die Projektdurchführungsstrategie entspricht weitgehend der Strategie zur Inkrementellen Entwicklung, erlaubt jedoch bei Bedarf, beispielsweise bei kleinen Änderungen am System, das Überspringen von Spezifikations- und Entwurfsaktivitäten innerhalb einer Iteration im Projekt.

Die Auftraggeber-/Auftragnehmerschnittstelle

Kennzeichen eines Vergabeprojekts im V-Modell ist der parallele Ablauf zweier unabhängiger V-Modell Projekte, eines Auftraggeberprojekts und eines Auftragnehmerprojekts. Vorgaben für die Durchführung des Auftraggeberprojekts definiert der Projekttyp Systementwicklungsprojekt (AG), Vorgabe für die Durchführung des Auftragnehmerprojekts der Projekttyp Systementwicklungsprojekt (AN). Neben den Projekttypen gibt das V-Modell explizit die Schnittstelle zwischen den Projekten vor, im V-Modell als Auftraggeber/Auftragnehmer Schnittstelle (AG/AN Schnittstelle) bezeichnet.

Die AG/AN Schnittstelle umfasst eine Menge von Entscheidungspunkten und Produkten, die gemeinsam die Schnittstelle zwischen den Projekten modellieren. Die Entscheidungspunkte der Schnittstelle stellen Synchronisationspunkte von Auftraggeber- und Auftragnehmerprojekt dar. Sie sind vollständig in den Projektdurchführungsstrategien auf Auftraggeber- wie auch auf Auftragnehmerseite integriert. Zu den Entscheidungspunkten der Schnittstelle fließen jeweils Produkte vom Auftraggeber zum Auftragnehmer oder umgekehrt. Die Produkte triggern jeweils die Fortführung des Partnerprojekts. Entscheidungspunkte und Produkte der Schnittstelle geben so gemeinsam eine zeitlich Synchronisation von Auftraggeber und Auftragnehmerprojekt vor. Die Produkte der Schnittstelle sind in den Vorgehensbausteinen Vertragsgestaltung und Lieferung sowie Abnahme enthalten, jeweils abgestimmt auf Auftraggeberprojekt und Auftragnehmerprojekt. Abbildung B.6 stellt die AG/AN Schnittstelle mit den Entscheidungspunkten und den Schnittstellenprodukte dar.

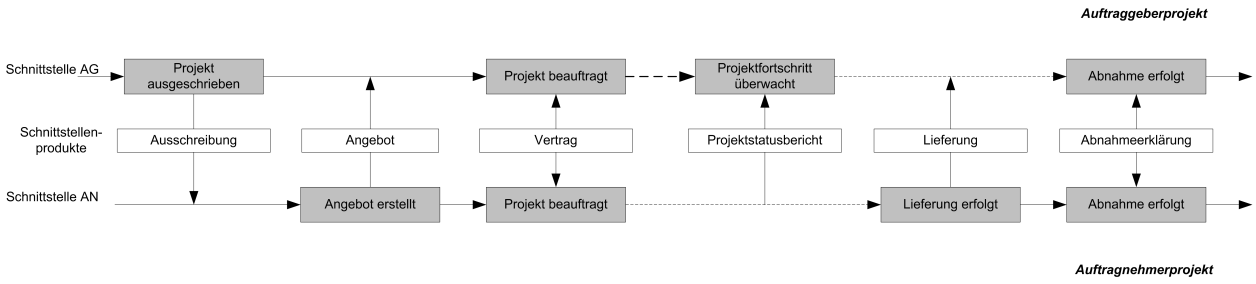

Abbildung B.6: Die Auftraggeber/Auftragnehmerschnittstelle

Ein Auftragnehmer gibt erst ein Angebot ab, wenn er zuvor vom Auftraggeber eine Ausschreibung erhalten hat. Der Vertrag wird von Auftraggeber und Auftragnehmer gemeinsam geschlossen. Während der Projektlaufzeit erhält der Auftraggeber zu definierten Zeitpunkten vom Auftragnehmer Projektstatusberichte. Mit der Lieferung geht das System vom Auftragnehmer an den Auftraggeber über. Mit dem Abnahmeerklärung wird das System offiziell vom Auftraggeber abgenommen.

Projektdurchführungsstrategien für Interne Projekte

Neben Vergabeprojekten unterstützt das V-Modell auch die Durchführung von Entwicklungsprojekten ohne explizite Vergabe. Der entsprechende Projekttyp wird im V-Modell mit dem Namen *Systementwicklungsprojekt (AG/AN)* bezeichnet. In einem AG/AN Projekt arbeiten Auftraggeber und Auftragnehmer gemeinsam in einem Projekt. Die Projektdurchführungsstrategien des Projekttyps entsprechen weitgehend den Projektdurchführungsstrategien eines Auftragnehmerprojekts. Es entfallen jedoch die Entscheidungspunkte der AG/AN Schnittstelle. Hinzu kommt dagegen der Entscheidungspunkt *Anforderungen festgelegt*.

Vergabe von Unteraufträgen

Die Projekttypen Systementwicklungsprojekt (AN) und Systementwicklungsprojekt (AG/AN) unterstützten die Vergabe von Unteraufträgen im Rahmen eines Entwicklunsprojekts. Bei einem Unterauftrag wird eine Teilkomponente des zu entwickelnden Systems an einen Unterauftragnehmer vergeben. Der Auftragnehmer nimmt für die Dauer des Entwicklungsauftrags dem Unterauftragnehmer gegenüber die Rolle eines Auftraggebers ein.

Die Schnittstelle zwischen Auftragnehmerprojekt und Unterauftragnehmerprojekt entspricht vollständig der AG/AN Schnittstelle mit allen Entscheidungspunkten und Produkten: Über eine Ausschreibung werden Angebote möglicher Bieter eingeholt. Diese werden bewertet und ein Bieter erhält den Zuschlag. Projektbegleitung und Abnahme erfolgen ebenfalls entsprechende der AG/AN Schnittstelle.

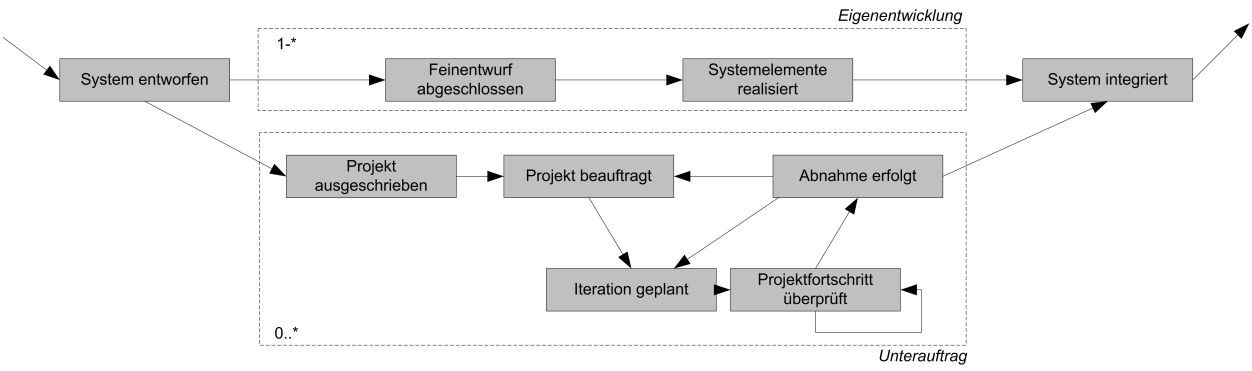

Abbildung B.7: Die Schnittstelle zum Unterauftragnehmer

Abbildung B.7 zeigt am Beispiel der Projektdurchführungsstrategie zur Inkrementellen Systementwicklung die Integration eines Unterauftrags. Ein Unterauftrag startet im V-Modell generell erst nach Durchlaufen des Entscheidungspunkts System entworfen. Zu diesem Entscheidungspunkt wurden die Schnittstellen aller Komponenten in der Systemarchitektur festgelegt und eine Schnittstellenspezifikation erstellt. Die Spezifikationen dienen einerseits als Vorlage für die Entwicklung der Systemkomponenten, andererseits übernehmen sie im Rahmen eines Unterauftrags die Rolle des Lastenhefts. Die auf Basis der Spezifikation entwickelte Komponente wird nach ihrer Lieferung einer Abnahmeprüfung unterzogen und in das System integriert. Alle Projektdurchführungsstrategien zur Systementwicklung erlauben die optionale Integration eines oder auch mehrerer paralleler Unteraufträge im Projektablauf.

Projektdurchführungsstrategie zur Einführung und Pflege eines Vorgehensmodell

Für den Projekttypen Einführung und Pflege eines organisationsspezifischen Vorgehensmodells steht im V-Modell eine gleichnamige Projektdurchführungsstrategie zur Verfügung. Die Strategie sieht einen iterativen Prozess der Softwareprozessverbesserung vor.

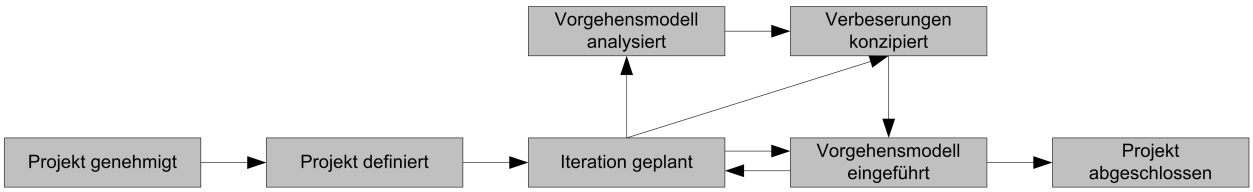

Abbildung B.8: Einführung und Pflege eines Vorgehensmodells

Abbildung B.8 zeigt die Projektdurchführungsstrategie zur Einführung und Pflege eines organisationsspezifischen Vorgehensmodells. Parallel zu den Produkten im entsprechenden Vorgehensbaustein modelliert die Strategie in ihren Entscheidungspunkten die Phase der Analyse des aktuellen Vorgehensmodells mit der Identifikation von Prozessproblemen, die Konzeption von Prozessänderungen und ihre Modellierung im Vorgehensmodell sowie die Einführung des angepassten Vorgehensmodells.

B.4 Projektspezifische Anpassung - das Tailoring

Vor seiner Anwendung in Projekten muss das V-Modell auf die Anforderungen und Bedürfnisse des jeweiligen Projekts angepasst werden. Im V-Modell wird dieser Vorgang der projektspezifischen Anpassung als *Tailoring* bezeichnet.

Das V-Modell unterstützt ein explizites Tailoringmodell auf der Basis von Projekttypen und Projektmerkmalen. Mit der Wahl für einen spezifischen Projekttypen legt ein Projektleiter zu Beginn eines Projekts einen projektspezifischen Ausschnitt aus dem V-Modell fest mit einer initialen Aus-

wahl an Vorgehensbausteinen und einer ausgewählten Menge an möglichen Projektdurchführungsstrategien. Diese Auswahl an Vorgehensbausteinen und Projektdurchführungsstrategien ist durch den Projekttypen eindeutig vorgegeben.

Über Projektmerkmale hat der Projektleiter nun die Möglichkeit, explizit Vorgehensbausteine und Projektdurchführungsstrategien hinzu zu wählen. Ein Projektmerkmal entspricht einer Charakterisierung eines Projekts. Jedem Projektmerkmal ist ein festgelegter Wertebereich zugeordnet. Die Zuordnung eines spezifischen Wertes zu einem Projektmerkmal hat unmittelbare Auswirkungen auf die Menge der Vorgehensbausteine und Projektdurchführungsstrategien des projektspezifische V-Modell. Je nach Projektmerkmalswert wird ein Vorgehensbaustein oder eine Projektdurchführungsstrategie der initialen Auswahl hinzugefügt oder nicht.

Das V-Modell definiert 9 Projektmerkmale. Ein Beispiel eines Projektmerkmals ist der Systemgegenstand. Zur Belegung dieses Projektmerkmals stehen die Werte: 'Entwicklung', 'Wartung' und 'Weiterentwicklung' zur Verfügung. Die Belegung des Merkmals mit dem Wert 'Wartung' führt zur Auswahl der Projektdurchführungsstrategie Wartung und Pflege als passende Strategie für das Projekt. Bei Belegung des Projektmerkmals mit dem Wert 'Entwicklung' oder 'Weiterentwicklung' erhält der Projektleiter dagegen die drei Entwicklungsstrategien: Inkrementelle, Komponentenbasierte und Agile Systementwicklung. Die Strategie zur Wartung entfällt dagegen. Der Wert 'Weiterentwicklung' führt zusätzlich zur Auswahl des Vorgehensbausteins Weiterentwicklung und Pflege.

Dieser Prozess des Tailorings kann auf Grund der formalen Basis des V-Modells mit Hilfe von Werkzeugen durchgeführt werden. Ergebnis des Anpassungsprozess ist ein ideal auf das jeweilige Projekt angepasstes projektspezifisches V-Modell.

B.5 Das V-Modell Metamodell

In diesem Abschnitt wird das Metamodell des V-Modells XT vorgestellt. Die Darstellung lehnt sich an die Paketstruktur der aktuellen Metamodell-Dokumentation zum V-Modell XT 1.2 (vgl. [132]) an. Das hier vorgestellte Metamodell hat nicht den Anspruch der Vollständigkeit, sondern soll lediglich ein Verständnis für die verwendeten Konzepte vermitteln. Da die Schnittstelle des Vorgehensmodells zu den zu integrierenden Methoden sehr klein ist und im Wesentlichen nur das Paket Vorgehensbausteine im Vorgehens-Metamodell betrifft, wird dieser Ansatz als ausreichend für die Ziele diese Arbeit angesehen.

Metamodell der Vorgehensbausteine

Abbildung B.9 stellt das Metamodell eines Vorgehensbausteins dar. Im Zentrum steht das Metamodell des Produktmodells, repräsentiert durch die Elemente *Produktgruppe*, *Produkt* und *Thema*. Eine Produktgruppe fasst eine Menge von Produkten nach inhaltlichen Gesichtspunkten zusam-

men. Produktgruppen dienen ausschließlich zur Strukturierung des Produktmodells im V-Modell. Sie haben keine Relevanz für die Projektdurchführung selbst. Ein Produkt deckt einen thematisch zusammengehörigen Bereich innerhalb der Produktgruppe ab. Es ist gleichzeitig ein im Vorgehensmodell definierter Ergebnistyp. Instanzen von Produkten sind konkrete Projektergebnisse. Ein Produkt kann aus einer Menge von Themen bestehen. Ein Thema deckt einen thematischen Teilbereich innerhalb eines Produkts ab.

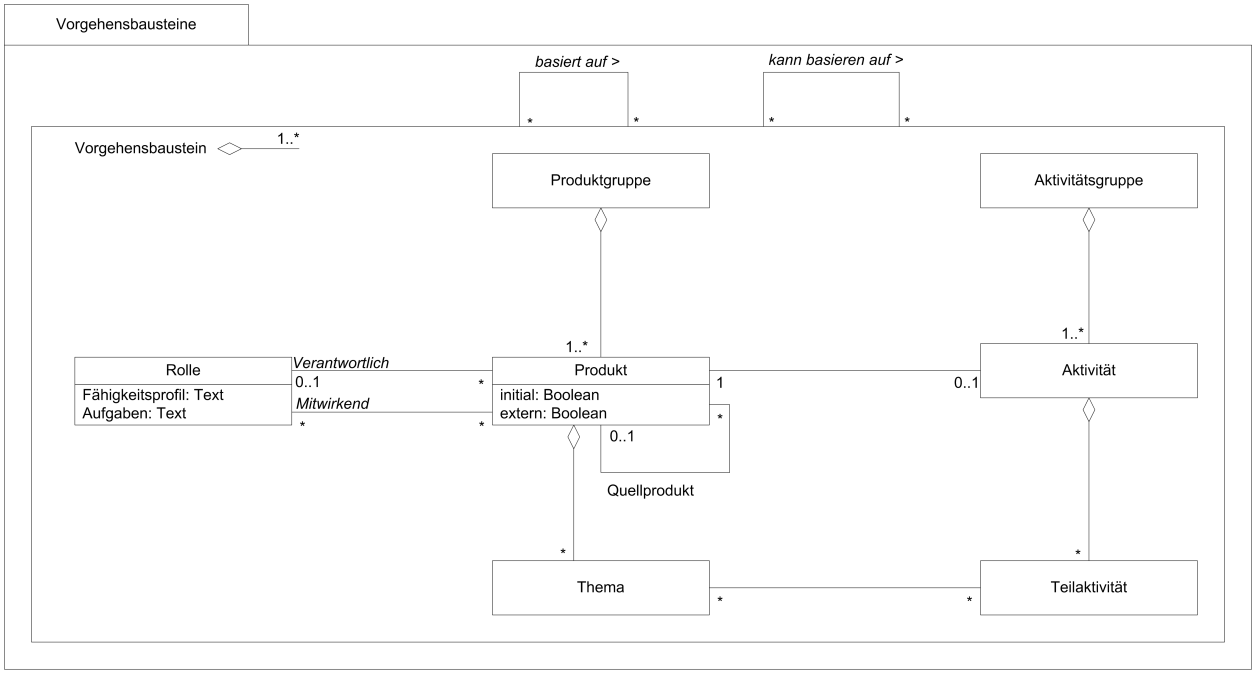

Abbildung B.9: Metamodell der Vorgehensbausteine (Quelle [132], eigene Darstellung)

Ein Produkt kann als initial, extern oder abhängig gekennzeichnet sein. Ein initiales Produkt wird genau einmal in einem Projekt instanziiert. Beispiele sind der Projektplan oder das Lastenheft. Diese Produkte braucht es immer und genau einmal in einem Projekt. Ein externes Produkt wird einem Projekt von außen zur Verfügung gestellt. Externe Produkte sind dadurch gekennzeichnet, dass ihnen keine verantwortliche Rolle und keine Aktivität zugeordnet wird. Ein Beispiel eines externen Produkts ist der Projektvorschlag. Ein Projektvorschlag initiiert ein V-Modell Projekt. Bei seiner Erstellung existiert das Projekt jedoch noch nicht. Abhängige Produkte sind Produkte, die als Ergebnis einer erzeugenden Abhängigkeit in einem Projekt erstellt werden. Erzeugende Abhängigkeiten werden im folgenden Abschnitt diskutiert. Zur Kennzeichnung eines Produktes als initial oder extern unterstützt das Metamodell für Produkte die entsprechend benannten Attribute.

Das Metamodell zum Aktivitätsmodell ist parallel zum Metamodell des Produktmodells aufgebaut. Elemente sind *Aktivitätsgruppen*, *Aktivitäten* und *Teilaktivitäten*. Parallel zur Produktgruppe erlauben Aktivitätsgruppen eine grobgranulare Strukturierung des Aktivitätsmodells im V-Modell. Bei der Entwicklung des V-Modells wurde auf eine Korrelation von Produktgruppen und Aktivitätsgruppen geachtet. Diese Einschränkung wird im Metamodell nicht explizit gefordert, hat sich jedoch

auf Grund der zu berücksichtigenden Abhängigkeitsstrukturen als sinnvoll erwiesen. Demnach gibt es im V-Modell zu jeder Produktgruppe genau eine Aktivitätsgruppe. Die Aktivitätsgruppe umfasst genau die Aktivitäten, die zur Erstellung der Produkte in der zugehörigen Produktgruppe erforderlich sind. Jede Aktivität beschreibt genau den Prozessschritt zur Erstellung ihres Produktes. Bei komplexen Aktivitäten kann eine verfeinerte Beschreibung über Teilaktivitäten sinnvoll sein. Eine Teilaktivität beschreibt einen Prozessschritt innerhalb einer Aktivität. Ergebnisse von Teilaktivitäten werden ausschließlich in den Themen zu dem von der Aktivität erstellten Produkt dokumentiert. Die Einhaltung dieser Einschränkung wird über eine entsprechende Konsistenzbedingung sichergestellt.

Verantwortlich für die Produkterstellung sind *Rollen*. Das Metamodell zum Rollenmodell umfasst genau ein Element, die Entität Rolle. Eine Rolle wird definiert durch einen Namen, ein Fähigkeitsprofil und eine Aufgabendefinition. Aufgabe einer Rolle ist die Erarbeitung der ihr zugeordneten Produkte. Das Metamodell unterscheidet bei der Zuordnung zwischen verantwortlichen Rollen und mitwirkenden Rollen. Für jedes Produkt im V-Modell (mit Ausnahme externer Produkte) gibt es eine für die Erstellung verantwortliche Rolle. An der Erstellung können jedoch weitere Rollen unterstützend mitwirken und dabei ihre Fähigkeiten bzw. ihr Wissen einbringen.

Beziehungen zwischen den Elementen können sich über Vorgehensbausteingrenzen hinweg ziehen. So kann ein Produkt einem anderen Vorgehensbaustein zugeordnet werden wie die Produktgruppe, zu der es gehört oder ein Thema, das ihm zugeordnet ist. Eine Rolle kann einem anderen Vorgehensbaustein zugeordnet werden, wie das Produkt, für das sie verantwortlich ist. Dies führt zu einer Abhängigkeitsstruktur zwischen Vorgehensbausteinen. Modelliert werden diese Abhängigkeiten explizit über die Beziehungen *basiert auf* und *kann basieren auf* zwischen Vorgehensbausteinen. Diese Beziehungen sind wie folgt zu verstehen:

- Ein Vorgehensbaustein A *basiert auf* einem Vorgehensbaustein B, wenn Elemente aus dem Vorgehensbaustein A Beziehung zu Elementen aus dem Vorgehensbaustein B haben. Diese Information ist essentiell für das Tailoring. Wurde Vorgehensbaustein A im Tailoring ausgewählt, muss auch Vorgehensbaustein B mit ausgewählt werden, um ein konsistentes Teilmodell als Tailoringergebnis zu gewährleisten. Eine *basiert auf* Beziehung entspricht einem mathematischen 'und' zwischen Vorgehensbausteinen.

- Die Beziehung *kann basieren auf* entspricht einer alternativen *basiert auf* Beziehung. Ein Vorgehensbaustein A kann auf Vorgehensbaustein B oder Vorgehensbaustein C basieren. Bei einer Auswahl von Vorgehensbaustein A muss demnach entweder Vorgehenbaustein B oder Vorgehensbaustein C mit ausgewählt werden um ein konsistentes Teilmodell als Tailoringergebnis zu erhalten. Eine *kann basieren auf* Beziehung entspricht einem mathematischen 'xor' zwischen Vorgehensbausteinen.

Metamodell der Produktabhängigkeiten

Produkte weisen eine Reihe von strukturellen und inhaltlichen Abhängigkeiten untereinander auf. Zur Modellierung der Abhängigkeiten unterstützt das Metamodell das Konzept der Produktabhängigkeit. Das V-Modell Metamodell unterscheidet vier Arten von Produktabhängigkeiten:

- **Erzeugende Produktabhängigkeiten:** Eine erzeugende Produktabhängigkeit beschreibt eine gerichtete Beziehung zwischen zwei Produkten. Sie definiert, welche Bedingungen im Ausgangsprodukt die Erzeugung eines oder mehrere Zielprodukte nach sich ziehen kann. Beispielsweise wird im Projekthandbuch festgelegt, zu welchen Terminen in einem Projekt ein Projektstatusbericht zu erstellen ist. Modelliert wird diese Beziehung durch eine erzeugende Produktabhängigkeit vom Projekthandbuch zum Projektstatusbericht. Interessant werden erzeugende Produktabhängigkeiten vor allem bei der Festlegung, dass ein bestimmtes Produkt im Projekt **nicht** zu erstellen ist. Beispielsweise kann im Projekthandbuch festgelegt werden, dass das Führen einer Risikoliste aus bestimmten Gründen in einem Projekt nicht erforderlich ist. Damit erlauben erzeugende Produktabhängigkeiten eine flexible, an die jeweilige Situation angepasste Projektplanung und -steuerung.

- **Inhaltliche Produktabhängigkeiten:** Eine inhaltliche Produktabhängigkeit beschreibt inhaltliche Abhängigkeiten zwischen verschiedenen Produkte. Diese müssen im Rahmen der Qualitätssicherung mit berücksichtigt werden. Diese Produktabhängigkeit nennt eine Menge von Produkten, deren Inhalte in einem Projekt immer konsistent gehalten werden müssen. Beispielsweise müssen Lasten- und Pflichtenheft bezüglich der Anforderungen an das zu entwickelnde System zueinander konsistent sein. Änderungen am Lastenheft müssen demnach im Pflichtenheft nachgezogen werden.

- **Strukturabhängigkeiten:** Strukturabhängigkeiten beschreiben hierarchische Strukturen von Produkten. Das V-Modell modelliert beispielsweise über Strukturabhängigkeiten die im Modell vorgegebene Systemstruktur. Strukturabhängigkeiten erlauben es dem V-Modell, eine mindestens dreistufige Hierarchisierung der Systemspezifikation zu fordern, ausgehend vom System selbst, über Einheiten bin hin zu Modulen. Diese Vorgaben erlauben im V-Modell die Festlegung eines qualitätsgesicherten Entwicklungsprozess bis auf Modulebene.

- **Tailoringabhängigkeiten:** Tailoringabhängigkeiten beschreiben die für das Tailoring relevanten Beziehungen im V-Modell. Tailoringabhängigkeiten bestehen ausschließlich zwischen Produkten, die bedingt durch ihre Inhalte das hinzutailoren eines neuen Vorgehensbausteins initiieren und dem Projekthandbuch, in dem die Wahl des neuen Vorgehensbausteins zu dokumentieren ist. Wird beispielsweise bei der Erstellung der Systemarchitektur festgelegt, dass ein Fertigprodukt eingesetzt werden soll, zieht diese Entscheidung das Hinzufügen des Vorgehensbausteins *Evaluierung von Fertigprodukten* nach sich. Die Tailoringabhängigkeit besteht in diesem Fall zwischen der Systemarchitektur, in der das Auftreten des Fertigprodukts

festgestellt wurde, und dem Projekthandbuch, in dem die Hinzunahme des neuen Vorgehensbausteins dokumentiert wird.

Abbildung B.10 zeigt das Metamodell zu den vier Arten von Produktabhängigkeiten. Jeder Produktabhängigkeit können über Produktabhängigkeitserweiterungen zusätzliche Produktreferenzen zugeordnet werden. Produktabhängigkeitserweiterungen wurden im V-Modell Metamodell eingeführt, um Produktabhängigkeiten modular erweiterbar und damit tailorbar zu gestalten. Mit der Auswahl eines neuen Vorgehensbausteins im Tailoringprozess können einer bereits vorhandenen Produktabhängigkeit über eine Produktabhängigkeitserweiterung weitere Produkte zugeordnet werden.

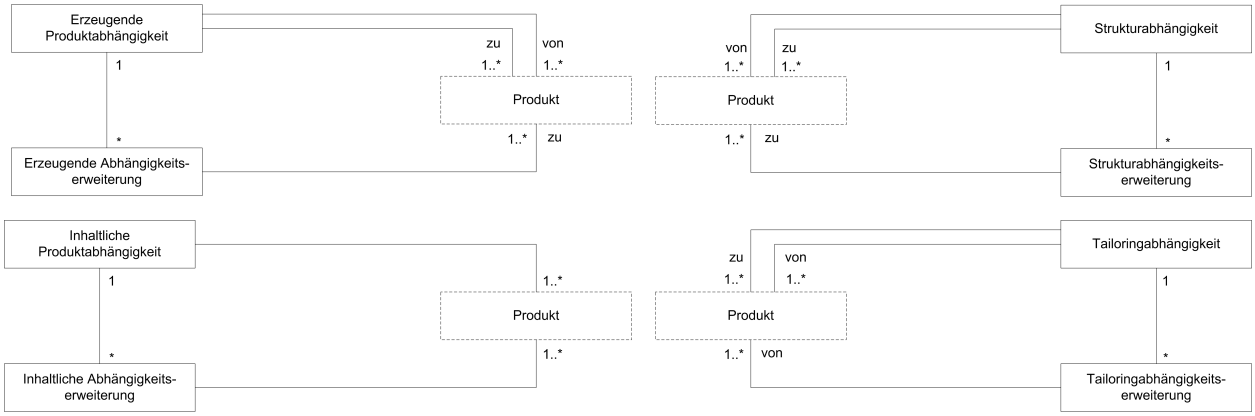

Abbildung B.10: Metamodell der Produktabhängigkeiten (Quelle [132], eigene Darstellung)

Metamodell der Projektdurchführungsstrategien

Das Ablaufmodell im V-Modell wird über die Konzepte *Entscheidungspunkt* und *Projektdurchführungsstrategie* modelliert. Ein Entscheidungspunkt ist ein Meilenstein und definiert einen Übergang zwischen zwei Projektfortschrittsstufen (Phasen). Einem Entscheidungspunkt ist eine Menge von Produkten zugeordnet. Qualität und Inhalte dieser Produkte liefern die Entscheidungskritierien für den Übergang in die folgende Projektfortschrittsstufe. Projektdurchführungsstrategien modellieren die Reihenfolge der Entscheidungspunkte.

Zur Modellierung von Projektdurchführungsstrategien stellt das Metamodell, wie in Abbildung B.11 dargestellt, Ablaufbausteine zur Verfügung. Ein Ablaufbaustein ordnet den Entscheidungspunkten der Projektdurchführungsstrategie Ablaufentscheidungspunkte zu und definiert für jeden Ablaufentscheidungspunkt eine Menge möglicher Nachfolge-Ablaufentscheidungspunkte. Ein Ablaufbaustein legt so explizit die Menge aller erlaubten Abläufe über den Entscheidungspunkten zur Projektdurchführungsstrategie fest.

Zur Modellierung paralleler Abläufe unterstützt das Metamodell das Konzept der Parallelabläufe. Ein Parallelablauf umfasst eine Menge von Parallelablaufteilen. Statt einer Menge einzelner Ablaufentscheidungspunkte kann einem Ablaufentscheidungspunkt alternativ ein Parallelablauf als

Nachfolger zugeordnet werden. Ein Parallelablauf stellt eine Auswahl möglicher Teilabläufe (Parallelablaufteil) als Nachfolger für diesen Ablaufentscheidungspunkt zur Verfügung. Die Modellierung eines jeden Parallelablaufteils erfolgt wieder über einen Ablaufbaustein. Einem Parallelablaufteil kann eine Vielfachheit zugeordnet werden. Die Vielfachheit besagt, wie oft ein Parallelablaufteil im Rahmen der Projektplanung durchlaufen werden darf. Erlaubte Werte sind

- *: beliebig oft,
- 1..*: mindestens einmal,
- 1: genau einmal,
- 0..1: höchstens einmal.

Das Konzept der Parallelabläufe wurde im V-Modell eingeführt, um ähnlich zur Modulstruktur der Vorgehensbausteine auch für Projektdurchführungsstrategien eine gewisse Modularisierung zu erreichen und so die Anwendung des V-Modells für Projekte flexibler zu gestalten.

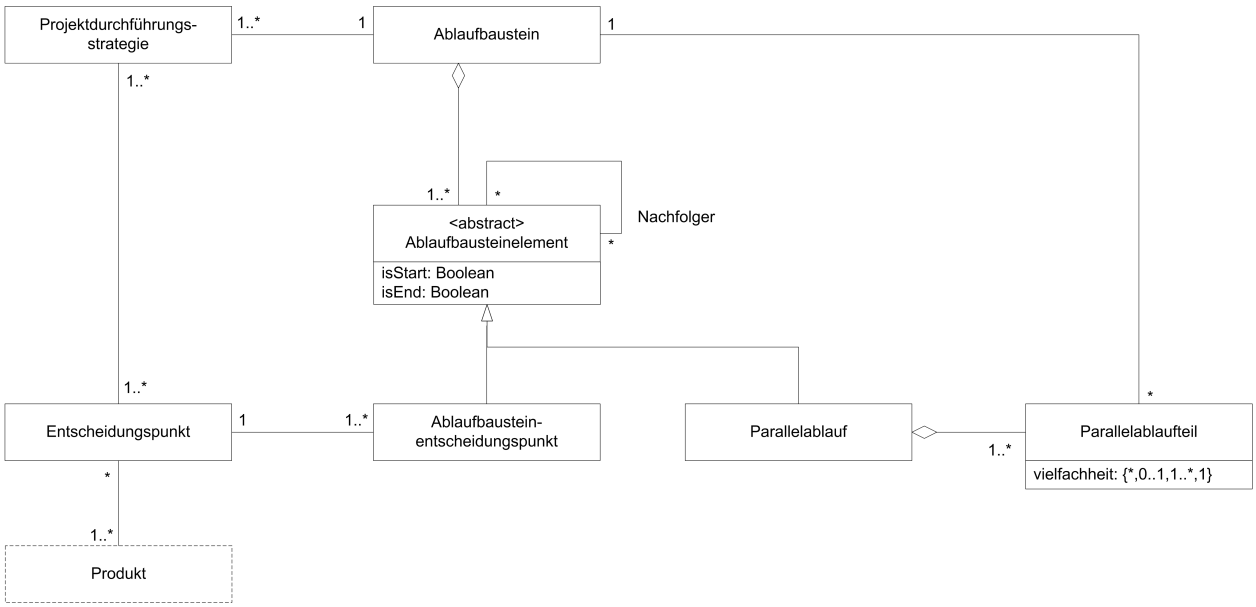

Abbildung B.11: Metamodell der Projektdurchführungsstrategien (Quelle [132], eigene Darstellung)

Metamodell zum Tailoringmodell

Ein kennzeichnendes Merkmal des V-Modells ist das explizit vorgegebenen Tailoringmodell zur Unterstützung der projektspezifischen Anpassung. Das Tailoringmodell bringt Vorgehensbausteine und Projektdurchführungsstrategien in einen konkreten Projektkontext, den Projekttyp, und erlaubt eine projektspezifische Auswahl über Projektmerkmale.

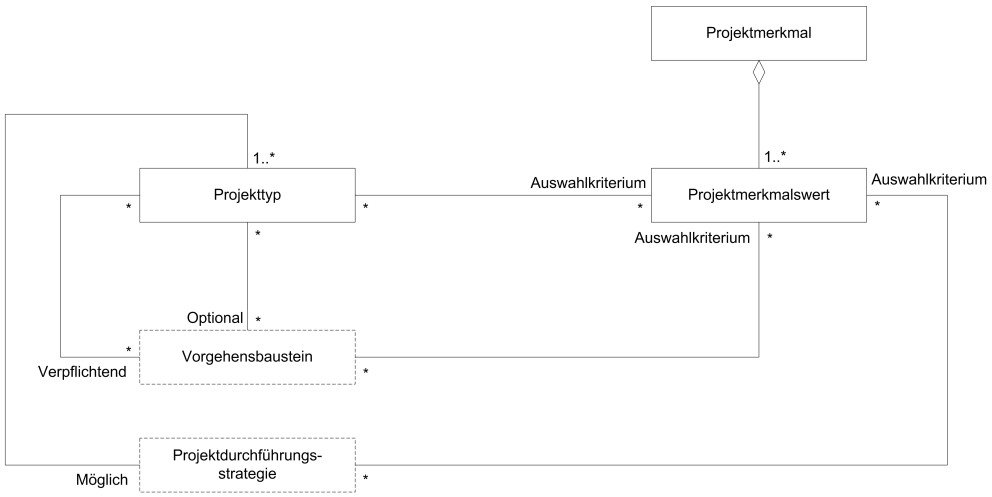

Abbildung B.12: Metamodell zum Tailoringmodell (Quelle [132], eigene Darstellung)

Die zentralen Tailoringkonzepte im Metamodell sind, wie in Abbildung B.12 dargestellt, der Projekttyp und das Projektmerkmal. Ein Projekttyp referenziert eine Menge von Vorgehensbausteinen und eine Menge von Projektdurchführungsstrategien. Dabei kennzeichnet der Projekttyp eine Menge von Vorgehensbausteinen als verpflichtend eine andere als optional. Optionale Vorgehensbausteine können im Rahmen der projektspezifischen Anpassung durch Wahl eines Projektmerkmalswert für ein spezifisches Projekt abgewählt werden. Die vom Projekttyp referenzierten Projektdurchführungsstrategien stehen bei der projektspezifischen Anpassung prinzipiell alle zur Auswahl. Projektmerkmalswerte können jedoch teilweise auch diese Auswahl beeinflussen und die Verwendung einer spezifischen Projektdurchführungsstrategie erzwingen.

Metamodell der unterstützenden Elemente

Das Metamodell bietet eine Reihe von Konzepten mit rein unterstützender Eigenschaft (vgl. Abbildung B.13). Diese erlauben beispielsweise die Modellierung von Zusatzinformationen im Modell wie Glossareinträge, Erläuterungen zu Abkürzungen und Quellenangaben.

Abbildung B.13: Metamodell der Unterstützenden Elemente (Quelle [132], eigene Darstellung)

Das V-Modell selbst ist, wie bereits angesprochen, methodenneutral. Es gibt keine explizite Methodik zur Anwendung in Projekten vor. Es kennt jedoch das Konzept einer Methodenreferenz bzw.

einer Werkzeugreferenz. Über diese Referenzen werden dem Anwender Hinweise und Vorschläge gegeben, welche Methodenklasse sich zur Durchführung einer Aktivität eignet. Methoden- und Werkzeugreferenzen sind lediglich zur Unterstützung der Anwender gedacht. Sie machen keine Vorgaben hinsichtlich einer verpflichtend anzuwendenden Methodik bzw. anzuwendender Werkzeuge.

Literaturverzeichnis

[1] *Meyers Neues Lexikon.* Meyers Lexikon Verlag, 1993.

[2] Ahonen, Jarmo J. und Hanna-Miina Sihvonen: *How Things Should Not Be Done: A Real-World Horror Story of Software Engineering Process Improvement.* In: *EuroSPI*, Seiten 59–70, 2005.

[3] Álvarez, José, Andy Evans und Paul Sammut: *MML and the Metamodel Architecture.* In: Whittle, Jon (Herausgeber): *WTUML: Workshop on Transformation in UML 2001*, April 2001.

[4] Ambler, Scott W.: *Using CRC Cards.* SIGS Books, 1995.

[5] Andresen, Andreas: *Komponentenbasierte Softwareentwicklug mit MDA UML2 und XML.* Hanser Verlag, 2004.

[6] Atkinson, Colin, Joachim Bayer, Christian Bunse, Eric Kamsties, Oliver Laitenberger, Roland Laqua, Dirk Muthig, Barbara Paech, Jürgen Wüst und Jörg Zettel: *Component based product line engineering with UML.* Addison-Wesley, 2002.

[7] Atkinson, Colin und Thomas Kühne: *The Essence of Multilevel Metamodeling.* In: *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, Seiten 19–33, London, UK, 2001. Springer-Verlag.

[8] Balzert, Helmug: *Lehrbuch der Software-Technik 2 - Software-Management, Software-Qualitaetssicherung, Unternehmensmodellierung.* Spektrum Akademischer Verlag, 1998.

[9] Balzert, Helmut: *Lehrbuch der Software-Technik 1 - Software-Entwicklung.* Spektrum Akadademischer Verlag, 2001.

[10] Basili, Victor R., Gianluigi Caldiera und H. Dieter Rombach: *The Goal Question Metrics Approach.* Encyclopedia of Software Engineering, 1:528–532, 1994.

[11] Basili, Victor R. und H. Dieter Rombach: *Tailoring the software process to project goals and environments.* In: *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, Seiten 345–357, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[12] Beck, Kent: *eXtreme Programming Explained (2.Auflage).* Addison-Wesley, Reading Massachusetts, 2004.

[13] Bender, Helmut: *Software-Engineering in der Praxis : das Bertelsmann-Modell.* CW-Publ, München, 1983.

[14] Benington, H. D.: *Production of Large Computer Programs.* In: *ICSE*, Seiten 299–310, 1987.

[15] Bergner, Klaus, Andreas Rausch, Marc Sihling und Alexander Vilbig: *A Componentware Development Methodology based on Process Patterns.* In: *Proceedings of the 5th annual Conference on the Pattern Languages of Programs*, 1998.

[16] Bischofberger, W. und G. Pomberger: *Prototyping-Oriented Software Development.* Springer Verlag, 1992.

[17] Blechinger, Sabine: *Entwurf eines Vorgehens zur Prozessanalyse und zur Prozessverbesserung einer Software-Entwicklungsorganisation.* Diplomarbeit, TU-München, 2004.

[18] Boehm, Barry W.: *A Spiral Model of Software Development and Enhancement.* Computer, 21(5):61–72, 1988.

[19] Boehm, Barry W., Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald Reifer und Bert Steece: *Software Cost Estimation with CoCoMo II.* Prentice-Hall, 2000.

[20] Booch, Grady: *Object Oriented Analysis and Design.* The Benjamin / Cummings Publishing Company, Inc., Second Edition Auflage, 1994.

[21] Broy, Manfred und Ketil Stolen: *Specification and Development of Interactive Systems.* Springer, 2001.

[22] Buhl, Axel: *Grundkurs Software-Projektmanagement.* Hanser, 2004.

[23] Buschermöhle, Ralf, Heike Eekhoff und Apl. Prof. Dr. Berndard Josko: *SUCCESS: Erfolgsfaktoren aktueller IT-Projekte in Deutschland.* Objektspektrum, (1):42–47, 2007.

[24] Buschermöhle, Ralf, Heike Eekhoff und Apl. Prof. Dr. Bernhard Josko: *SUCCESS - Erfolgs- und Misserfolgsfaktoren bei der Durchführung von Hardeware- und Softwareentwicklungsprojekten in Deutschland.* BIS-Verlag, CvO Universität Oldenburg, 2006.

[25] Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad und Michael Stal: *Pattern-Oriented Software Architecture: A System of Patterns.* Wiley, New York, 1996.

[26] Cheesman, John und John Daniels: *UML Components, A Simple Process for Specifying Component-Based Systems.* Addison-Wesley, 2001.

[27] Chen, Peter P.: *The Entity-Relationship Model - A basis for the Enterprise View of Data.* In: *AFIPS National Computer Conference*, Seiten 77–84, 1977.

[28] Chroust, Gerhard: *Modelle der Software-Entwicklung.* Oldenbourg, 1992.

[29] Coad, Peter und Edward Yourdon: *Object-oriented analysis.* Yourdon Press, Upper Saddle River, NJ, USA, 1990.

[30] Coad, Peter und Edward Yourdon: *Object-oriented design.* Yourdon Press, Upper Saddle River, NJ, USA, 1991.

[31] Cockburn, Alistair: *Agile Software-Entwicklung.* Mitp-Verlag, 2003.

[32] Dahl, Ole-Johan, Edsger W. Dijkstra und C.A.R. Hoare: *Structured Programming*, Band 8 der Reihe *A.P.I.C. Studies in Data Processing.* Academic Press, 1972.

[33] DeMarco, Tom: *Structured Analysis and System Specification.* Yourdon Press Prentice Hall Building, 1979.

[34] Deming, William E.: *Out of the crisis.* Cambridge, MA: Massachusetts Institute of Technology, Center for Advanced Engineering Study, 1986.

[35] Deursen, Arie van, Paul Klint und Joost Visser: *Domain-specific languages: an annotated bibliography.* SIGPLAN Not., 35(6):26–36, 2000.

[36] Dijkstra, Edsger W.: *Go to statement considered harmful.* jCACM, 11(3):147–148, März 1968.

[37] Dijkstra, Edsger W.: *The humble programmer.* Communication of the ACM, 15(10):859–866, 1972. Turing Award lecture.

[38] Dilg, Peter: *Praktisches Qualitätsmanagement in der Informationstechnologie.* Carl Hanser Verlag München Wien, 1995.

[39] Dröschel, Wolfgang und Manuela Wiemers: *Das V-Modell 97.* Oldenbourg Verlag, 1999.

[40] D'Souza, Desmond Francis und Alan Cameron Wills: *Objects, Components and Frameworks with UML: The Catalysis Approach.* Addison-Wesley Publishing Company, 1998.

[41] End, Wolfgang: *Softwareentwicklung, Leitfaden für Planung, Realisierung und Einführung von DV-Verfahren.* Publicis Corporate Publishing, Auflage 6, 1987.

[42] Feeley, Bob Mc: *IDEAL: A User's Guide for Software Process Improvement.* Technischer Bericht, 1996.

[43] Fernstroem, Christer und Lennart Ohlsson: *ESF: an approach to industrial software production.* In: *Proceedings of the fourth conference on Software engineering environments: research and practice*, 1989.

[44] Firesmith, Donald G. und Brian Henderson-Sellers: *The OPEN Process Framework.* Addison-Wesley Longman, 2001.

[45] Fowler, Martin: *Refactoring - Improving the Design of Existing Code.* Addison-Wesley, Reading/Massachusetts, 1999.

[46] Frick, Andreas: *Der Softwareentwicklungsprozeß - Ganzheitliche Sicht.* Carl Hanser Verlag München Wien, 1995.

[47] Gamma, Erich, Richard Helm, Ralph Johnson und John Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA, 1995.

[48] Gane, Chris und Trish Sarson: *Structured Systems Analysis.* Prentice Hall, 1979.

[49] Gnatz, Michael: *Vorgehensmodelle und Projektplanung.* Doktorarbeit, Institut für Informatik, Technische Universität München, 2005.

[50] Gnatz, Michael, Frank Marschall, Gerhard Popp, Andreas Rausch und Wolfgang Schwerin: *Enabling a Living Software Development Process with Process Patterns.* Technischer Bericht, TU-München, Juli 2003.

[51] Gödel, Kurt: *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.* Monatshefte für Mathematik und Physik, 38:173–198, 1931.

[52] Government Commerce (OGC), Office of: *IT Infrastructure Library - Version 3.* Technischer Bericht. [Zugriff 17.11.2007].

[53] Graham, Ian, Brian Henderson-Sellers und Houman Younessi: *The OPEN Process Specification.* Addison-Wesley, 1997.

[54] Harry, Mikel und Richard Schroeder: *Six Sigma.* Campus Fachbuch, 3. Auflage 2000.

[55] Henderson-Sellers, Brian und Bhuvan Unhelkar: *OPEN modeling with UML.* Addison-Wesley, 2000.

[56] (Hrsg.), IBM: *IBM Global Services Method 3.0, Familie von Vorgehensweisen für Serviceprojekte.* Technischer Bericht ZZ81-0045-00, IBM Internes Dokument, 2000.

[57] IEEE: *Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990.* Technischer Bericht, IEEE Computer Society Press, 1990.

[58] IFPUG: *Function Point Counting Practices: Manual Release 4.0.* International Function Point Users Group, Blendonview Office Park, 5008-28 Pine Creek Drive, Westerviell, OHZ 43001-4899, 1994.

[59] International Organization for Standardization: *ISO/IEC 9000:2000 Quality Management Systems – Fundamentals and vocabulary.* Technischer Bericht, International Organization for Standardization, 2000.

[60] International Organization for Standardization: *ISO/IEC 9001:2000 Quality Management Systems – Guidelines for performance improvements.* Technischer Bericht, International Organization for Standardization, 2000.

[61] International Organization for Standardization: *ISO/IEC 9001:2000 Quality Management Systems – Requirements.* Technischer Bericht, International Organization for Standardization, 2000.

[62] International Organization for Standardization: *ISO/IEC 12207:1995/Amd 2:2002 Information technology – Software life cycle processes, institution = International Organization for Standardization.* Technischer Bericht, 2002.

[63] International Organization for Standardization: *ISO/IEC 15288:2002 Systems engineering – System life cycle processes, institution = International Organization for Standardization.* Technischer Bericht, 2002.

[64] International Organization for Standardization: *ISO/IEC 15504:2004 Information technology – Process assessment – Part 4: Guidance on use for process improvement and process capability determination.* Technischer Bericht, International Organization for Standardization, 2003.

[65] International Organization for Standardization: *ISO/IEC 15504:2004 Information technology – Process assessment – Part 1: Concepts and vocabulary.* Technischer Bericht, International Organization for Standardization, 2004.

[66] International Organization for Standardization: *ISO/IEC 15504:2004 Information technology – Process assessment – Part 3: Guidance on performing an assessment.* Technischer Bericht, International Organization for Standardization, 2004.

[67] International Organization for Standardization: *ISO/IEC 15504:2006 Information technology – Process Assessment – Part 5: An exemplar Process Assessment Model.* Technischer Bericht, International Organization for Standardization, 2006.

[68] Jacobson, Ivar: *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison-Wesley Publishing Company, 1992.

[69] Jacobson, Ivar, Grady Booch und James Rumbaugh: *The Unified Software Development Process.* Addison-Wesley Longman, 1999.

[70] Jones, Cliff B.: *Systematic Software Development using VDM.* Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1990.

[71] Jr., Stanley M. Sutton, Dennis Heimbigner und Leon J. Osterweil: *APPL/A: A Language for Software Process Programming.* ACM Transactions on Software Engineering and Methodology, 4(3):221–286, 1995.

[72] Jurow, Susan und Susan B. Barnard: *Introduction: TQM fundamentals and overview of contents.* Journal of Library Administration, 18(1/2), 1-13. (EJ 469 099), 1993.

[73] Kastens, Uwe und Hans Kleine Büning: *Modellierung.* Hanser Verlag, 2005.

[74] KBST: *UfAB IV - Unterlagen für die Ausschreibung und Bewertung von IT-Leistungen, Version 1.0.* Technischer Bericht, (KBST) Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung, November 2006.

[75] Kelly, John: *Logik.* Pearson Studium, 2003.

[76] Kneuper, Ralf: *CMMI Verbesserung von Softwareprozessen mit Capability Maturity Model Integration.* dpunkt.verlag GmbH, 2003.

[77] Kruchten, Philippe: *The Rational Unified Process: An Introduction.* Addison-Wesley, Boston, 3 Auflage, 2003.

[78] Kuhn, Thomas: *Die Struktur wissenschaftlicher Revolutionen.* Suhrkamp-Taschenbuch Wissenschaft. Verlag Suhrkamp, Frankfurt am Main, 12 Auflage, 1993.

[79] Manfred Broy, Andreas Rausch und: *Das V-Modell XT. Grundlagen, Erfahrungen und Werkzeuge.* noch nicht erschienen, April 2008.

[80] McMenamin, Stephen M. und John F. Palmer: *Essential systems analysis.* Yourdon Press, Upper Saddle River, NJ, USA, 1984.

[81] Members of the Assessment Method Integrated Team: *Standard CMMI Appraisal Method for Process Improvement (SCAMP), Version 1.1: Method Definition Document.* Technischer Bericht, 2001.

[82] Meyer, Bertrand: *Object-Oriented Software Construction.* Prentice Hall, Second Edition Auflage, 1997.

[83] Montangero, Carlo, Jean-Claude Derniame, Badara Ali Kaba und Brian Warboys: *The Software Process: Modelling and Technology.* In: *Software Process: Principles, Methodology, Technology*, Seiten 1–14, 1999.

[84] Naur, Peter und Brian Randell (Herausgeber): *Proceedings, NATO Conference on Software Engineering*, Garmisch, Germany, 1968.

[85] Object Management Group (OMG): *UML 2.0 Infrastructure Specification.* Technischer Bericht.

[86] Object Management Group (OMG): *Unified Modeling Language, Version 2.0.* Technischer Bericht.

[87] Object Management Group (OMG): *OCL 2.0 Specification Version 2.0.* Technischer Bericht, 2005.

[88] Object Management Group (OMG): *Software Process Engineering Metamodel, Version 1.1.* Technischer Bericht, 2005.

[89] Object Management Group (OMG): *Software Process Engineering Metamodell SPEM 2.0 OMG Draft Adopted Specification.* Technischer Bericht, OMG, 2006.

[90] Object Management Group (OMG): *Software Process Engineering Metamodell SPEM 2.0 Revised Submission.* Technischer Bericht, OMG, 2006.

[91] Oestereich, Bernd: *Analyse und Design mit UML 2.0 - Objektorientierte Softwareentwicklung.* Oldenburg, 2005.

[92] Offices of Government Commerce: *Managing Successful Projects with PRINCE2.* Offices of Government Commerce, 2005.

[93] Osterweil, Leon J.: *Software Processes Are Software Too.* In: *ICSE*, Seiten 2–13, 1987.

[94] Pikkarainen, Minna, Outi Salo und Jari Still: *Deploying Agile Practices in Organizations: A Case Study.* In: Richardson, Ita, Pekka Abrahamsson und Richard Messnarz (Herausgeber): *EuroSPI*, Lecture Notes in Computer Science, Seiten 16–27. Springer, 2005.

[95] Poensgen, Benjamin und Bertram Bock: *Function-Point-Analyse - Ein Praxishandbuch.* dpunkt.verlag, 2005.

[96] Raasch, Jörg: *Systementwicklung mit Strukturierten Methoden.* Carl Hanser Verlag München, 1993.

[97] Rechtin, Eberhardt und Mark W. Maier: *The Art of Systems Architecting.* CRC Press, Inc., 2000, Second Edition.

[98] Richardson, Ita, Pekka Abrahamsson und Richard Messnarz (Herausgeber): *Software Process Improvement, 12th European Conference, EuroSPI 2005, Budapest, Hungary, November 9-11, 2005, Proceedings*, Band 3792 der Reihe *Lecture Notes in Computer Science.* Springer, 2005.

[99] Rook, Paul: *Controlling software projects.* Software Engineering Journal, 1(1), 1986.

[100] Royce, Winston W.: *Managing the Development of Large Software Systems: Concepts and Techniques.* In: *Technical Papers of Western Electronic Show and Convention (WesCon)*, 1970.

[101] Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy und William Lorensen: *Object-Oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[102] Scheerer, Immanuel: *Architekturentwurf mit dem V-Modell XT und der UML.* Diplomarbeit, TU-München, 2006.

[103] Schumann, Jörg und Manfred Gerisch: *Software-Entwurf: Prinzipien, Methoden, Arbeitsschritte, Rechnerunterstützung.* Verlag Technik Berlin, 1984.

[104] Schwaber, Ken: *Agile Project Management with Scrum.* Prentice Hall, 2004.

[105] Schwarze, Jochen: *Projektmanagement mit Netzplantechnik.* Verlag Neue Wirtschafts-Briefe, 2001.

[106] Smith, Einar: *Elementare Berechenbarkeitstheorie.* Springer Verlag, Berlin Heidelberg, 1996.

[107] Software Engineering, Center of: *CoCoMo II Model Definition Manual.* [Zugriff 17.11.2007].

[108] Stachowiak, Herbert: *Allgemeine Modelltheorie.* Springer-Verlag, Wien New York, 1973.

[109] Strahringer, Susanne: *Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips.* In: *Modellierung*, 1998.

[110] Taylor, Richard N., Frank C. Belz, Lori A. Clarke, Leon Osterweil, Richard W. Selby, Jack C. Wileden, Alexander L. Wolf und Michael Young: *Foundations for the Arcadia environment architecture.* SIGSOFT Software Engineering Notes, 13(5):1–13, 1988.

[111] Team, CMMI Product: *CMMI for Development, Version 1.2 - Improving processes for better products.* Technischer Bericht, Software Engineering Institute (SEI), Carnegie Mellon University, August 2006. [Zugriff 17.11.2007].

[112] Wallmüller, Ernest: *Ganzheitliches Qualitätsmanagement in der Informationsverarbeitung.* Hanser Verlag, 1998.

[113] Wastell, David Graham, Selma Arbaoui, Jacques Lonchamp und Carlo Montangero: *The Human Dimension of the Software Process.* In: *Software Process: Principles, Methodology, Technology*, Seiten 165–200, 1999.

[114] Weinberg, Victor: *Structured Analysis.* Yourdon Press, Englewood Cliffs, 1978.

[115] Yourdon, Edward: *Modern Structured Analysis.* Prentice Hall, 1989.

[116] Yourgrau, Palle: *A World Without time, The Forgotten Legacy of Gödel and Einstein.* Penguin Group, 2005.

Internet-Links

[117] *Baldridge National Quality Award.* http://baldrige.nist.gov/. [Zugriff 17.11.2007].

[118] *Das V-Modell XT - Entwicklungsstandard des Bundes.* http://www.v-modell-xt.de. [Zugriff 17.11.2007].

[119] *Eclipse Process Framework.* http://www.eclipse.org/epf/. [Zugriff 17.11.2007].

[120] *European Foundation for Quality Management.* http://www.efqm.org/. [Zugriff 22.02.2007].

[121] *Extensible Markup Language (XML) 1.0 (Fourth Edition) W3C Recommendation 16 August 2006, edited in place 29 September 2006.* http://www.w3.org/TR/REC-xml/. [Zugriff 17.11.2007].

[122] *Industrial XP.* http://www.industrialxp.com/. [Zugriff 17.11.2007].

[123] *Manifesto for Agile Software Development.* http://www.agilemanifesto.org/. [Zugriff 17.11.2007].

[124] *Microsoft Solutions Framework.* http://msdn2.microsoft.com/en-us/teamsystem/aa718801.aspx. [Zugriff 17.11.2007].

[125] *Modell Driven Architecture.* http://www.omg.org/mda/. [Zugriff 17.11.2007].

[126] *Object Engineering Process.* http://www.oose.de/oep/. [Zugriff 17.11.2007].

[127] *Object Management Group (OMG).* http://www.omg.org/. [Zugriff 22.02.2007].

[128] *SCRUM.* http://www.controlchaos.com/. [Zugriff 17.11.2007].

[129] *The OPEN Homepage.* http://www.open.org.au/. [Zugriff 17.11.2007].

[130] *The Standish Group.* http://www.standishgroup.com. [Zugriff 17.11.2007].

[131] *Unified Method Architecture - Model Download.* http://dev.eclipse.org/viewcvs/index.cgi/ *checkout*/org.eclipse.epf/design/ UMA-model.zip? rev=HEAD&cvsroot=Technology_Project. [Zugriff 17.11.2007].

[132] *V-Modell XT: Metamodell und Konsistenzbedingungen.* http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Infomaterial/Meta-Modell-Dokumentation.pdf. [Zugriff 17.11.2007].

[133] *XML Schema 1.0 W3C Recommendations.* http://www.w3.org/XML/Schema. [Zugriff 17.11.2007].

[134] Object Management Group (OMG): *Catalog of UML Profile Specifications.* http://www.omg.org/technology/documents/profile_catalog.htm. [Zugriff 01.07.2007].

[135] Sauer, Chris und Christine Cuthbertson: *The State of IT-Projektmanagement in the UK 2002-2003.* http://www.computerweekly.com/Articles/2003/11/03/198355/more-it-projects-are-hitting-their-targets.htm. [Zugriff 17.11.2007].

[136] Software Engineering, Center of: *CoCoMo II - Homepage.* http://sunset.usc.edu/research/COCOMOII/cocomo_main.html. [Zugriff 17.11.2007].

[137] Wikipedia: *Meilensteintrendanalyse — Wikipedia, Die freie Enzyklopädie*, 2007. [Online; Stand 19. November 2007].