5. Integrierter Modellierungsansatz für Vorgehensmodelle

In diesem Kapitel greifen wir die Ergebnisse der formalen Modellierung aus Kapitel 4 auf und überführen sie auf einen integrierten, UML 2-basierten Modellierungsansatz für Vorgehensmodelle (IMV). Dieser beschreibt die beiden wesentlichen Ergebnisse dieser Arbeit: das *Metamodell* und das *Lebenszyklusmodell* (vgl. Abbildung 1.2, Kapitel 1.2). Im ersten Teil betrachten wir die Architektursichten und führen die Basis- und Strukturmodelle ein. Auf der nächsten Modellierungsebene beschreiben wir einen Ansatz zur Komponentenbildung – insbesondere mit Berücksichtigung der Abhängigkeitsstrukturen. Diese betrachten wir weiter gehend im Abschnitt *Konfigurationen*. Konfigurationen sind unser Vorschlag eines Konstruktionsmodells und stellen das zentrale Element für die Komposition eines Vorgehensmodells sowie die Bildung von Varianten dar. Im Anschluss stellen wir auf der letzten Modellierungsebene die physische Repräsentation eines Vorgehensmodells vor. Dies umfasst Verzeichnis- und Schemahierarchien. Im zweiten Teil dieses Kapitels definieren wir den Lebenszyklus eines Vorgehensmodells. Wir beschreiben einen phasenorientierten Ansatz inform von UML 2 Verhaltensdiagrammen. Basierend auf den durch diese Arbeit abgedeckten Anwendungsfällen detaillieren wir die einzelnen Phasen des Lebenszyklus. Wir schließen dieses Kapitel mit einer Beschreibung des begleitenden Werkzeugkonzepts.

Am Ende dieses Kapitels wurde auf Basis eines formalen Modells ein integrierter Modellierungsansatz für Vorgehensmodelle (IMV) eingeführt. Neben einer Strukturmodellierung ist insbesondere der Lebenszyklus eingeführt. Die dort definierten Prozesse dienen nicht nur der Erstellung eines Vorgehensmodells, sondern auch seiner Pflege und Weiterentwicklung.

5.1. Modellierungssicht: Vorgehensmodell

Wir fokussieren zuerst die Strukturmodellierung. Diese haben wir bislang im Kapitel 4 weitgehend abstrahiert. Wir gehen in diesem Abschnitt auch auf Details ein, die erforderlich sind, um eine Referenzimplementierung zu ermöglichen. Im Anhang C stellen wir ein entsprechendes Werkzeug vor.

5.1.1. Basis- und Strukturmodelle

Wir nähern uns der Modellierung des Vorgehensmetamodells schrittweise an. Zuerst geben wir die Basisstruktur inform eines UML 2 Paketdiagramms an (Abbildung 5.1). Anders als Gnatz [Gna05] fokussieren wir nicht planungsorientierte Vorgehensmodelle, sondern stellen allgemeine Strukturen ins Zentrum. Unser Ansatz in Abbildung 5.1 zeigt eine Aufteilung[1] in drei Pakete:

Basis: Das Basispaket definiert dem Gedanken eines Frameworks [HR02] folgend alle notwendigen Basiselement- und Beziehungstypen. Prozesselemente oder Prozesselementabhängigkeiten (Kapitel 4.2) aber auch die Konfigurationen sind hier zu finden.

Prozesskomponenten: Prozesskomponenten enthalten die strukturellen Verfeinerungen des Basispakets, mit denen die inhaltlichen Bestandteile eines Vorgehensmodells modelliert und erstellt werden.

1 Die Architektur ist modular ausgelegt und daher bereits an dieser Stelle erweiterbar. Auf Erweiterungsoptionen gehen wir in Anhang B noch einmal ein.

Planung: Das Paket Planung enthält eine Modellbeschreibung für Planungstechniken. Ohne entsprechende Planungseinheiten, also nur mit Elementen der Pakte *Basis* und *Prozesskomponenten*, ist ein Vorgehensmodell lediglich ein druckbarer Leitfaden. Zur Operationalisierung sind Planungseinheiten erforderlich, die beispielsweise Gnatz [Gna05] detailliert modelliert hat. Bis auf weiteres betrachten wir die Planung als Blackbox, greifen dieses Teilsystem aber später noch einmal auf.

Vorgehensmodell: Das Paket Vorgehensmodell fasst die Elemente zusammen, die wir für die Ableitung von Vorgehensmodellen benötigen. Im Wesentlichen geht es hier um die Öffnung und gegenseitige Bekanntmachung der Namensräume und Elemente, die zum Erstellen eines Vorgehensmodells benötigt werden.

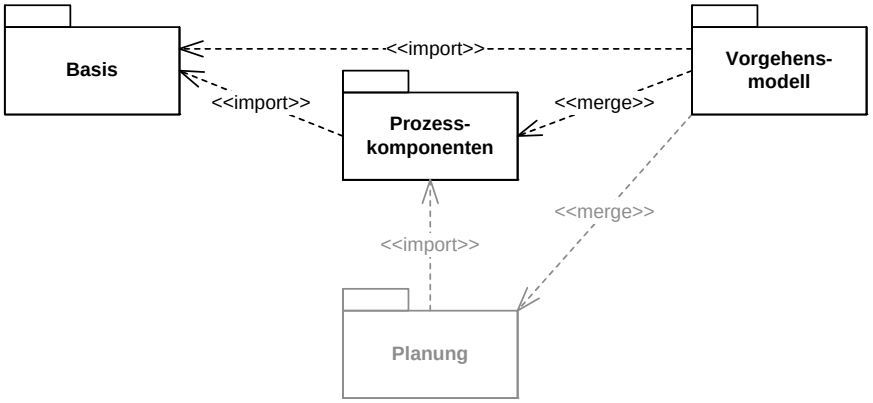

Abbildung 5.1.: Paketstruktur des Vorgehensmetamodells

Im Anschluss stellen wir erst das *Basis*-Paket detaillierter vor und geben dann einen ersten Eindruck des Gesamtsystems. Wir betrachten dabei allgemeine Eigenschaften wie die Variabilität aber auch Dokumentation, Ressourcen und Konfiguration.

Basiselemente und -modell. Vorgehensmodelle sind nach unserer Theorie *komplexe komposite Strukturen*. Als kleinste Elemente definieren wir *Prozesselemente*, die mithilfe einer *Konfiguration* in Prozesskomponenten zusammengefasst werden. *Prozesskomponenten* wiederum fassen wir ebenfalls konfigurativ zu Vorgehensmodellen zusammen. Wir erhalten somit ein hierarchisches Konstruktionsmodell, das wir in Kapitel 4.1 motiviert haben. Neben der reinen Betrachtung von Strukturen (Ergebnisstrukturen) betrachten wir weiterhin Abläufe, die ebenfalls Teil eines Vorgehensmodells sind. Das von uns hier modellierte Konzept ist ein einfaches Basiskonzept, das hier noch hinter dem Umfang etablierter Vorgehensmetamodelle wie zum Beispiel dem V-Modell XT zurücksteht. Wir setzen jedoch auf konsequente Erweiterbarkeit und können mit diesem Basismodell alle benötigen Aspekte modellieren. Im Anhang B geben wir dafür ein Beispiel an.

Nachdem das *Prozesselement* unsere grundlegende Einheit ist, verfeinern wir zunächst diesen Teil des Metamodells (Abbildung 5.2). Gemäß Definition 4.3 muss ein Prozesselement im Rahmen einer Modellierung konkretisiert werden. Wir modellieren das Prozesselement somit als abstrakte Klasse. Zusätzlich modellieren wir auch den grundlegenden Abhängigkeitstyp auf der Basis der erweiterten Prozesselementabhängigkeit. Im weiteren Verlauf der Arbeit definieren wir entsprechende Spezialisierungen für konkrete Prozesselemente und Prozesselementabhängigkeiten. Die wesentlichen Elemente des Modells in Abbildung 5.2 sind die Klassen `ProzessElement`, `ProzessElementAbhaengigkeit` und `Konfiguration`. Sie bilden die Grundlage aller weiteren Diskussionen. Nicht vordergründig aber dennoch vorhanden ist die Information, dass wir generell versionierbare Elemente betrachten (Klasse `Version`). Auch die Dokumentation der einzelnen Elemente wir nicht vernachlässigt.

Das grundlegende Modell orientiert sich am Konzept aus Kapitel 4.1. Ausgangspunkt hierfür ist eine Konfiguration, die eine Menge von Prozesselementen und eine Menge

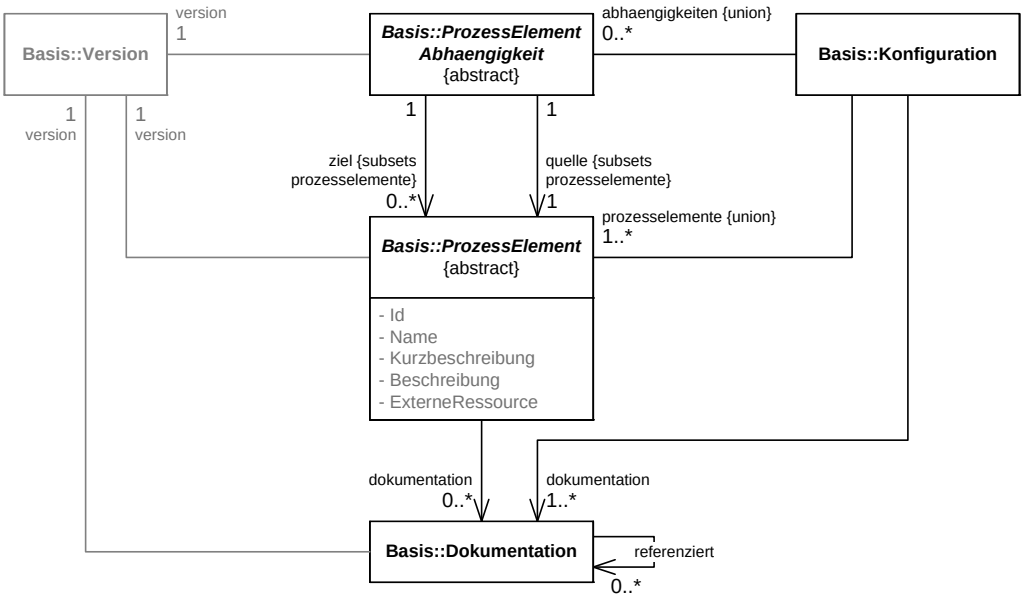

Abbildung 5.2.: Verfeinerung des Pakets Basis des Vorgehensmetamodells

von Relationen enthält. Analog zu Kapitel 4.1.3 betrachten wir hier wieder einen Abhängigkeitsgraph G, in dem nun Spezialisierungen der Klasse `ProzessElement` die Knoten bilden und Spezialisierungen der Klasse `ProzessElementAbhaengigkeit` die Beziehungen zwischen den Prozesselementen herstellen (vgl. Abbildung 4.13, unten). Ein weiterer Aspekt, der unmittelbar durch das Pakte *Basis* abgedeckt wird, ist die *Variationsfähigkeit* von Vorgehensmodellen. Um den Konzepten der Produktlinienentwicklung (vgl. Kapitel 2.3.2) zu folgen, müssen wir *Variationspunkte* im Metamodell vorsehen. Einmal haben wir Variationspunkte durch definierte Soll-Bruchstellen im Gesamtkonzept definiert. Prozesskomponenten zusammen mit Konfigurationen auf der Vorgehensmodellebene liefern hier einen einfach Mechanismus. Dieser ist jedoch zu grobgranular, sodass wir auch auf der Ebene einzelner Prozesselemente einen entsprechenden Mechanismus vorsehen müssen. In Abbildung 5.3 ist der hier konzipierte Mechanismus gezeigt, der eine erste Spezialisierung der Prozesselementabhängigkeit darstellt (vgl. hierzu auch Kapitel 4.3).

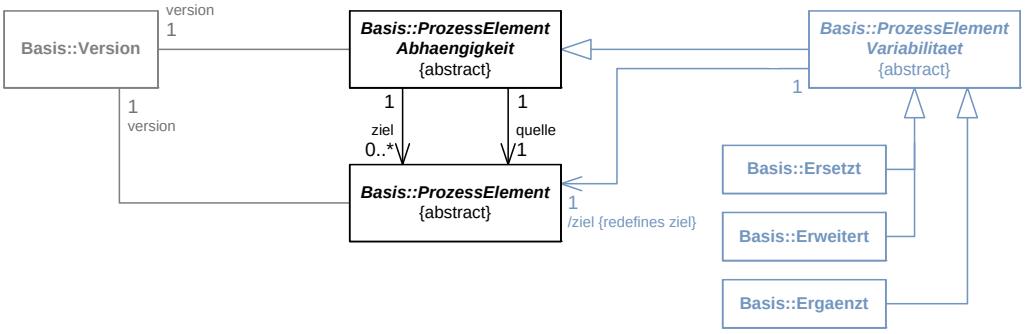

Abbildung 5.3.: Sicht des Pakets Basis: Variabilitätsanteile

Vorgehensmetamodell (vereinfacht). In Abbildung 5.4 geben wir eine vereinfachte Sicht des Vorgehensmetamodells[2] an. Diese zeigt die grundlegenden Klassen des Pa-

2 Ergänzend zu der hier gezeigten Modellierung verweisen wir auch auf Anhang C, wo wir eine Interpretation des Metamodells als DSL-Modell im Rahmen des Werkzeugkonzepts anbieten.

kets *Basis* und deren Einbettung in das Konzept Prozesskomponente. Jede Prozesskomponente enthält somit eine Menge von Prozesselementen und Abhängigkeiten, die in einer Konfiguration zusammengefasst und in Beziehung gestellt werden. Die Konfiguration einer Prozesskomponente ist privat[3].

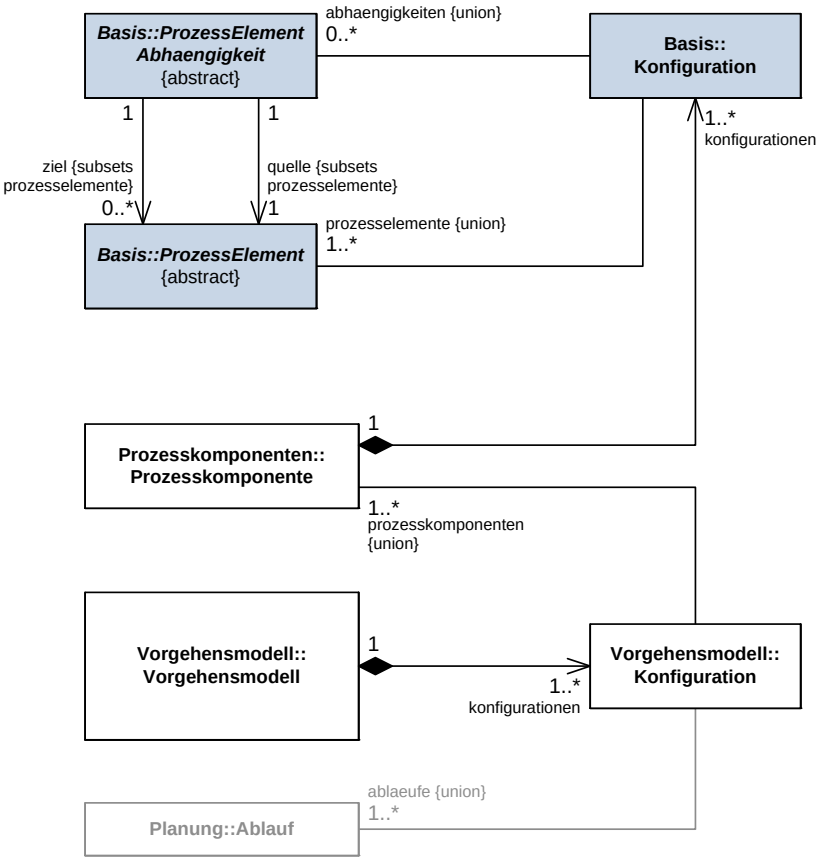

Abbildung 5.4.: Allgemeines Vorgehensmetamodell mit Basisklassen und -beziehungen

Analog zur Bildung der Prozesskomponenten wird auch bei der Erstellung von Vorgehensmodellen beziehungsweise Vorgehensmodellvarianten (die wir hier nicht unterscheiden) verfahren. Die Diskussion der Konfiguration von Prozesskomponenten führen wir jedoch später. Zunächst betrachten wir noch den Aspekt der Dokumentation der einzelnen Prozesselemente.

Dokumentation. Ein weiteres Strukturierungsmerkmal findet sich ebenfalls in diesem Teil des Modells. Bereits Gnatz [Gna05] befasste sich mit der Modellierung von (strukturierten) Texten. Bartelt und Herold [BH06] griffen diese Problematik wieder auf, diskutierten sie jedoch im Kontext eines allgemeinen Variantenmanagements. Sie stellten heraus, dass hier der Mensch eingebunden werden müsse, wenn im Kontext von zum Beispiel informalen und unstrukturierten Texten auf Varianten von Modellen gearbeitet werden muss. Sotó et al. [SM06c] adressieren das Problem in ähnlicher Weise. In dieser Arbeit abstrahieren wir jedoch von der Problematik von Texten, Beschreibungen etc. und wählen einen pragmatischen Ansatz, indem wir eine *allgemeine Dokumentation* auf der Ebene von Prozesselementen mit modellieren. Jedes Prozesselement beschreibt sich weitgehend selbst (durch entsprechende Attribute). Als Option sehen wir die Referenzierung weiterer Dokumentationseinheiten vor; ebenso wie die Möglichkeit, dass auch andere Elemente als Prozesselemente Dokumentationseinheiten referenzieren können.

3 Wir sehen jedoch prinzipiell mehrere Konfigurationen pro Prozesskomponente vor, um ggf. erweiterten Ansprüchen hinsichtlich der Variationsfähigkeiten nachkommen zu können, vgl. Kapitel 4.3.1

Hier ergibt sich die Möglichkeit, die Dokumentation[4] trotz der engen Bindung zum dokumentierten Element separat zu erstellen, zu verwalten und zu organisieren. Ein (entfernt) ähnliches Konzept finden wir bislang nur im V-Modell XT mit den so genannten Textbausteinen als abstrakten Modellelementen.

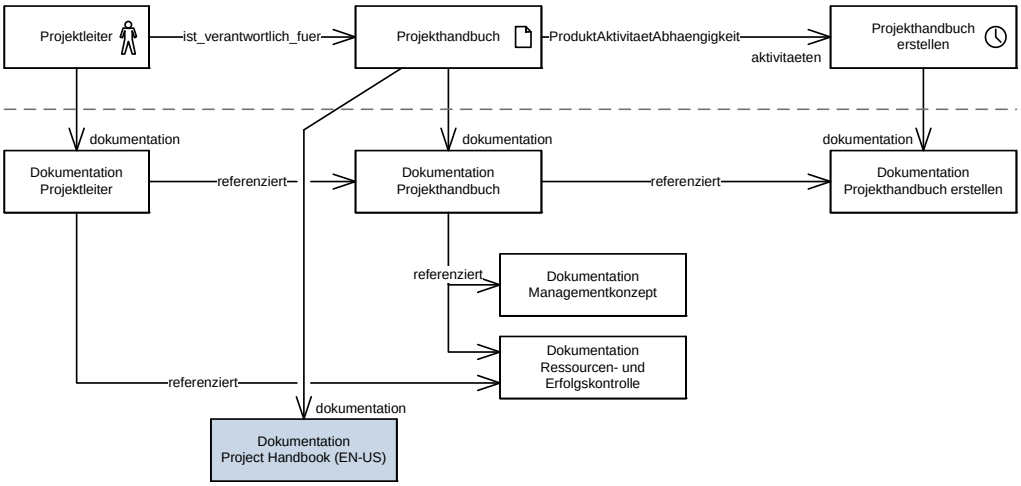

Abbildung 5.5.: Beispiel für die Trennung von Elementstruktur und Dokumentation eines Vorgehensmodells

Die Abbildung 5.5 zeigt als Vorgriff inform einer Instanziierung des Metamodells (inkl. der Submodelle) ein Beispiel für die Separation von Elementstrukturen und der Dokumentation. Die Vorgehensmodellstruktur, also die Struktur der Prozesselemente, ist im oberen Teil der Abbildung zu sehen; der untere Teil zeigt eine separate Dokumentationsstruktur. So ist es beispielsweise möglich, eine minimale (abstrakte, generische) Dokumentation für ein Vorgehensmodell zu erstellen und diese dann entweder anzupassen oder problemspezifisch zu ergänzen, ohne dass die Elementstrukturen geändert werden müssen.

Vorteile einer separierten Dokumentation

Ist es beispielsweise erforderlich, dass ein Vorgehensmodell multilingual zur Verfügung gestellt werden muss, kann durch die Separation der Vorgehensmodell-Elementstruktur und der Dokumentation ein Mehr an Flexibilität erzielt werden. Die Elementstrukturen des Vorgehensmodells müssen hier nur einmal erstellt werden, während die Dokumentation mehrsprachig zur Verfügung gestellt werden kann.

Ergänzend dazu kann die Dokumentation im Rahmen eines Anpassungsprozesses einfach ergänzt werden. Je nach Formalisierungsgrad des betrachteten Vorgehensmodells können über den Beziehungstypen zwischen der Dokumentation und den dokumentierten Elementen *Konsistenzbedingungen* definiert werden, um beispielsweise eine gewisse „Mindestdokumentation" zu erzwingen.

5.1.2. Submodelle und Komponentenbildung

In diesem Abschnitt befassen wir uns mit der grundlegenden Strukturmodellierung von Prozesskomponenten, die als Bausteine unseres Vorgehensmodellkonzepts fungieren. Wir stützen uns dabei auf den Begriffsbildungen aus Kapitel 2.4 ab, um einen inhaltlichen Rahmen zu schaffen. Nach der Modellierung der Strukturen mit allen relevanten Entitätstypen konzentrieren wir uns auf die Abhängigkeitsstrukturen. Wir greifen dabei

4 Zu beachten ist beim Terminus *Dokumentation*, dass es sich nicht zwangsläufig um ein Dokument wie im Produktmodell definiert (Abschnitt 5.1.2) handeln muss. Eine Dokumentation kann beispielsweise eine Checkliste oder eine allgemeine Verfahrensbeschreibung zum Beispiel für eine Aktivität sein.

die Diskussionen aus Kapitel 4 wieder auf und konzentrieren uns neben internen Abhängigkeiten schwerpunktmäßig auf Abhängigkeiten zwischen Prozesskomponenten. Die Inhalte dieses Abschnitts liefern die Grundlagen für den Rest dieses Kapitels. Neben der Modellierung auf Basis der UML 2 geben wir in weiten Teilen eine entsprechende Formalisierung mit an. Wo für das Verständnis erforderlich geben wir auch Beispiele an, in denen wir auf Inhalte zurückgreifen, die wir im Kontext von Anhang B erarbeitet haben.

Prozesskomponenten

Abbildung 5.6 zeigt eine vereinfachte Darstellung der Prozesskomponente[5]. Im Folgenden gehen wir detailliert auf Prozesskomponenten ein und nehmen die notwendigen Detaillierungen vor. Prozesskomponenten definieren wir als eigenständige Einheiten, die autonom verwendbar und (weiter-)entwickelbar sind [HR02, Sie04, VAC+05]. Gemäß unserer Modellierung in Kapitel 4 definieren wir Prozesskomponenten zunächst als Container für Prozesselemente. Wir konkretisieren hier jedoch und geben eine Prozesskomponente in einer Form an, wie sie für Vorgehensmodelle üblich ist (Abbildung 5.6).

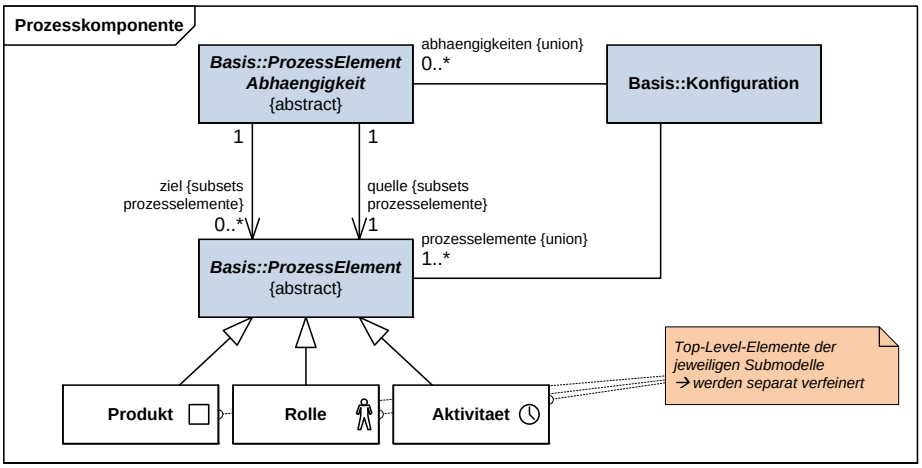

Abbildung 5.6.: Metamodell einer Prozesskomponente (ohne Verfeinerungen)

Wir orientieren uns bei der Definition von Prozesskomponenten am Konzept *Vorgehensbaustein* (Kapitel 3) des V-Modell XT. Das Method Plug-In von EPF/UMA stellt ebenfalls eine Option dar, jedoch sind wir hier bestrebt, auch generische Vorgehensmodelle erstellen zu können. Nicht desto trotz finden sich Bestandteile beider Konzepte hier wieder: die Integrationsdichte des V-Modells und die physische Modularität von EPF/UMA. Das Modell in Abbildung 5.6 zeigt insbesondere im oberen Teil die Nutzung der im letzten Abschnitt beschriebenen Basiskonzepte. Die interne Struktur einer Prozesskomponente ist durch die Vielzahl von *explizit* modellierten Beziehungen für die Konfiguration sehr komplex. Wir verfeinern daher im Folgenden schrittweise.

In Abbildung 5.6 ist als Verfeinerungsaspekt für die Prozesselemente die *Ontologie für Vorgehensmodelle* (vgl. Kapitel 2.4) enthalten. Wir verwenden den Begriff wie Gnatz [Gna05] und orientieren uns auch strukturell an seiner Modellierung. Abbildung 5.7 zeigt die drei wesentlichen Submodelle für Produkte, Rollen und Aktivitäten – jeweils durch eine Klasse repräsentiert. Zwischen den Submodellen bestehen Beziehungen, die beispielsweise durch das Rollenmodell hergestellt werden können.

5 Die hier modellierte Prozesskomponente ist nur eine mögliche Struktur, die wir jedoch im Rahmen dieser Arbeit bereits vordefinieren. Nach dem hier vorgestellten Konzept lassen sich weitere Spezialisierungen beziehungsweise Verknüpfungen von Prozesselementen vornehmen und in ein Vorgehensmodell integrieren. Im Anhang B zeigen wir hierfür ein Beispiel.

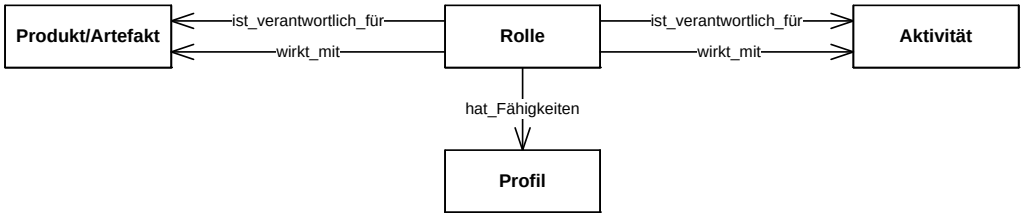

Abbildung 5.7.: Ontologie für Vorgehensmodelle im Kontext einer Prozesskomponente

Im Folgenden gehen wir auf die drei Submodelle vertiefend ein und verfeinern dadurch die Modellierung der Prozesskomponente. Nicht weiter verfeinern werden wir das Rollenmodelle, das bereits in einfacher Form der Abbildung 5.7 entnommen werden kann. Wichtig sind uns an dieser Stelle nur die Beziehungstypen, die wir im Kontext der Modellierung der Submodelle ebenfalls für die Rollen betrachten wollen.

Produktmodell

Wir führen zunächst die beiden komplexen Submodelle für Produkte und Aktivitäten ein. Die Abbildung 5.8 zeigt die Verfeinerung des Metamodells aus Abbildung 5.4 für den Kontext der Produkte. Das Produktmodell wird aufbauend auf der zentralen

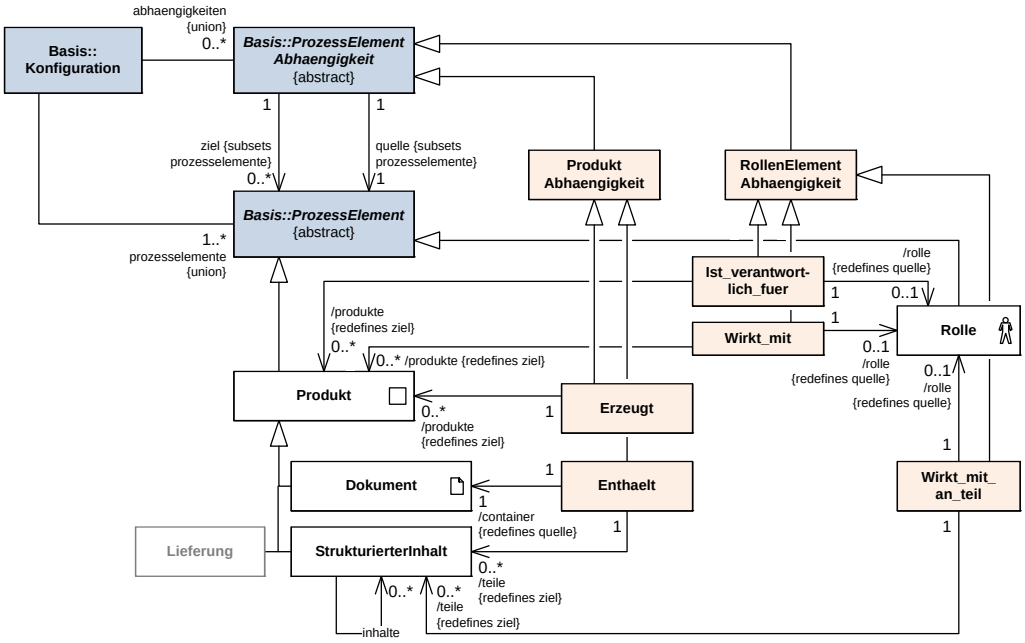

Abbildung 5.8.: Verfeinerung des Metamodells für den Kontext Produktmodell

Klasse `Produkt` (Kind von `ProzessElement`) modelliert. Über Produkten lassen wir Beziehungstypen der Klasse `ProduktAbhaengigkeit`[6] (Kind von `ProzessElementAbhängigkeit`) zu. Unser Produktmodell sieht im Wesentlichen nur allgemeine Produkte (Klasse `Produkt`), Dokumente (Klasse `Dokument`) und Lieferungen (Klasse `Lieferung`) vor. Wir orientieren uns für die Modellierung des Produktmodells an Gnatz [Gna05] und EPF/UMA (Kapitel 3.2.3). Für Dokumente modellieren wir bereits einfache Strukturen (Klasse `StrukturierterInhalt` als Kind von `Produkt`). Eine ähnlich verfeinerte Modellierung wäre zum Beispiel für Lieferung ebenfalls möglich, jedoch führen wir sie

6 Diesem Beziehungstyp hat das Konzept der Produktabhängigkeiten des V-Modell XT Pate gestanden. Wir verwenden dieses Konzept hier und überführen es nach konsequenter Weiterentwicklung gesamtheitlich auf das ganze Vorgehensmetamodell.

hier nicht durch. Die Verknüpfungen zwischen Elementen des Produktmodells werden über die Produktabhängigkeiten modelliert. Weiterhin hat das Produkt Beziehungen zum Rollenmodell. Über einen eigenen Beziehungstyp `RollenElementAbhaengigkeit` modellieren wir hier analoge Verknüpfungsmechanismen. Über die innere Struktur der einzelnen Klassen treffen wir an dieser Stelle keine weitere Aussage, verfeinern jedoch noch die Betrachtung der Beziehungstypen. Im Anschluss daran geben wir ein kurzes Beispiel.

Beziehungstypen im Produktmodell. Wir gehen nun detaillierter auf die Beziehungstypen ein, die aus den Prozesselementabhängigkeiten abgeleitet werden und konkretisieren diese zunächst für das Produktmodell. Wir geben auch hier nur ein Minimalmodell an, das unseren unmittelbaren Anforderungen genügt[7]. Wir modellieren das Produktmodell derart, dass es abgeschlossen ist, d. h. dass jeder durch das Produktmodell gegebene Beziehungstyp ausschließlich über Elementen des Produktmodells definiert ist.

Muster für Elemente und Abhängigkeiten

Wir verwenden für die Modellierung grundsätzlich folgendes Muster: Zu allen betrachteten Elementtypen modellieren analoge Beziehungstypen (vgl. Descriptoren aus EPF/UMA, Kapitel 3.2.3). Derart modellierte Beziehungen sind abgeschlossen, d. h. sie referenzieren nur Elemente des jeweiligen Submodells. Für die Kopplung von verschiedenen Submodellen modellieren nach demselben Muster spezialisierte Beziehungstypen.

Produktabhängigkeiten. In unserem Modell sind Produktabhängigkeiten Spezialisierungen von Relationen zwischen Prozesselementen vom Typ Produkt. Produktabhängigkeiten lassen sich vielfältig modellieren, wie zum Beispiel im V-Modell XT [RH06] oder in [Gna05]. Um die mögliche Breite der Begriffsdefinition zu verdeutlichen, verweisen wir auf den folgenden Auszug aus dem V-Modell XT-Glossar:

Produktabhängigkeiten im V-Modell XT

Eine Produktabhängigkeit beschreibt eine Konsistenzbedingung zwischen zwei oder mehreren Produkten. Dabei kann eine Produktabhängigkeit sowohl innerhalb eines Vorgehensbausteins als auch zwischen Produkten verschiedener Vorgehensbausteine bestehen.

Man unterscheidet Tailoring-Produktabhängigkeiten, erzeugende [...], strukturelle [...] und inhaltliche Produktabhängigkeiten. Alle diese Arten von Produktabhängigkeiten können relevante Produktabhängigkeiten sein. [RH06]

Das V-Modell XT als produktorientiertes Vorgehensmodell misst den Produktabhängigkeiten einen großen Stellenwert bei. Wir abstrahieren und reduzieren die Menge der relevanten Abhängigkeitstypen für unser Modell derart, dass wir nur die Erzeugung und strukturelle Bezüge modellieren (vgl. Abbildung **??**). Im folgenden spezifizieren wir die beiden Beziehungstypen, die wir aus der allgemeinen Produktabhängigkeit (Klasse: `ProduktAbhaengigkeit`) ableiten.

Für den Beziehungstyp `Erzeugt`, der als Klasse modelliert von `ProduktAbhaengigkeit` erbt, legen wir ein Verhalten fest, das im wesentlichen der einfachen Prozesselementabhängigkeit genügt. Abbildung 5.8 illustriert dies analog auch für den Beziehungstyp `Enthaelt`[8]. Für den Beziehungstyp `Erzeugt` und $x, y \in$ *Produkte* definieren wir:

$$\text{erzeugt}\,(x) = y, \text{bzw. erzeugt}\,(x, y) \tag{5.1}$$

7 Bei einer Erweiterung eines der Submodelle eines Vorgehensmodells sollte auch eine Anpassung der Assoziationstypen in Betracht gezogen werden. Dies entspricht dann einer *strukturellen Anpassung* im Sinne von Kapitel 2.2.1.

8 In den Abbildungen der Metamodell, beziehungsweise deren Ausschnitte wie zum Beispiel Abbildung 5.8, sind die Klassen, die Beziehungstypen modellieren, immer farblich hervorgehoben

Das Produkt x heißt der Terminologie des V-Modell XT folgend *erzeugendes Produkt*, y heißt *erzeugtes Produkt*. Eine „Selbsterzeugung", also *erzeugt* (x, x), verbieten wir. Analog verfahren wir für den Beziehungstyp `Enthaelt`. Sei $x \in$ *Dokumente* und seien $y_1, \ldots, y_n \in$ *StrukturierterInhalt*, dann gilt:

$$\text{enthaelt}(x) = \{y_1, \ldots, y_n\}, \text{bzw. enthaelt}(x, \{y_1, \ldots, y_n\}) \tag{5.2}$$

Das Dokument x nennen wir *Container*. Auf der Grundlage von Formel 5.2 definieren wir dabei strukturierte Dokumente.

Innerhalb einer Prozesskomponente sind die Produktabhängigkeiten in der Regel vollständig. Den Aspekt der Abhängigkeiten über die Grenzen einer Prozesskomponente hinaus vertiefen wir im weiteren Verlauf dieses Kapitels noch.

Rollen und deren Einbindung im Produktmodell. Rollen sind die Elemente eines Vorgehensmodells, die Personen einbringen. Die Erfahrung hat gezeigt, dass es einerseits erforderlich ist, Personen, die eine Rolle besetzen, über ihr Befugnisse, andererseits aber auch über ihre Aufgaben zu informieren. Üblicherweise werden Rollen enger mit Aktivitäten in Beziehung gesetzt. Lediglich das V-Modell XT setzt konsequent auf eine Kopplung von Rollen und Produkten. Für Prozesskomponenten nach der Modellierung aus Abbildung 5.6 setzt sich die Menge der in einer Prozesskomponente zu betrachtenden Prozesselemente zunächst wie folgt zusammen:

$$Pe = \text{Rollen} \cup \text{Produkte} \cup \text{Aktivitaeten} \tag{5.3}$$

Die einzelnen Teilmengen sind dabei *disjunkt*. Da wir in dieser Arbeit sowohl Rollen, Produkte und Aktivitäten auf der selben Ebene konkretisieren und dabei insbesondere Produkte und Aktivitäten gleich behandeln, modellieren wir die Beziehungstypen für Rollen (`Ist_verantwortlich_fuer`, `Wirkt_mit` und `Wirkt_mit_an_teil`) analog, siehe Formeln 5.4 und 5.5. Die Beziehungstypen sind gerichtet von Elementen aus *Rollen* $\subseteq$ *Pe* zu Elementen entweder aus *Produkte* oder *Aktiviaeten*. Wir definieren daher:

$$\text{ist\_verantwortlich\_fuer} \subseteq \text{Rollen} \times (\text{Produkte} \cup \text{Aktivitaeten}) \tag{5.4}$$

$$\text{wirkt\_mit} \subseteq \text{Rollen} \times (\text{Produkte} \cup \text{Aktivitaeten}) \tag{5.5}$$

$$\text{wirkt\_mit\_an\_teil} \subseteq \text{Rollen} \times \text{StrukturierterInhalt} \tag{5.6}$$

Diese allgemeinen Definitionen genügen jedoch noch nicht vollständig, da eine Rolle nun mit Produkten und Aktivitäten in Beziehung stehen kann. Dies kann in einigen Szenarios durchaus sinnvoll sein, jedoch sind auch Vorgehensmodelle wie das V-Modell XT zu berücksichtigen, die eine Gewichtung von Rollen und deren Beziehungen zu Ergebnistypen vornehmen. Wir verfeinern daher die Assoziationen für typechte Teilmengen[9].

Unabhängig von der Gewichtung der referenzierten Prozesselemente ist es jedoch sinnvoll, die beiden Assoziationen `Ist_verantwortlich_fuer` und `Wirkt_mit` näher zu beschreiben. Sei r eine Rolle, für die die Assoziationen `Ist_verantwortlich_fuer` und `Wirkt_mit` über Teilmengen von Prozesselementen definiert sind:

$$\text{ist\_verantwortlich\_fuer} \subseteq \text{Rollen} \times (\text{Produkte} \cup \text{Aktivitaeten}) \text{ und es gilt:}$$

$$\text{ist\_verantwortlich\_fuer}|_{\text{Produkte}} \subseteq \text{Rollen} \times \text{Produkte} \tag{5.7}$$

$$\text{ist\_verantwortlich\_fuer}|_{\text{Aktivitaeten}} \subseteq \text{Rollen} \times \text{Aktivitaeten} \tag{5.8}$$

und analog für die Assoziation `Wirkt_mit`:

$$\text{wirkt\_mit} \subseteq \text{Rollen} \times (\text{Produkte} \cup \text{Aktivitaeten}) \text{ und es gilt:}$$

$$\text{wirkt\_mit}|_{\text{Produkte}} \subseteq \text{Rollen} \times \text{Produkte} \tag{5.9}$$

$$\text{wirkt\_mit}|_{\text{Aktivitaeten}} \subseteq \text{Rollen} \times \text{Aktivitaeten} \tag{5.10}$$

9 Unter einer typechten Teilmenge verstehen eine Teilmenge von Prozesselementen, in der nur Elemente eines Typs enthalten sind, zum Beispiel ausschließlich Produkte.

dann gilt für ein $n+1$-stelliges Prädikat *ist_verantwortlich_fuer* $(r, \{p_1, \ldots, p_n\})$, dass die Elemente $p_1, \ldots, p_n$ nur von einem Typ sind. Also:

$$\neg\exists p_i \in \text{ist\_verantwortlich\_fuer}\,|_{\text{Produkte}}\,(r, \{p_1, \ldots, p_n\}) : p_i \in \text{Aktivitaeten} \quad (5.11)$$

$$\neg\exists p_i \in \text{ist\_verantwortlich\_fuer}\,|_{\text{Aktivitaeten}}\,(r, \{p_1, \ldots, p_n\}) : p_i \in \text{Produkte} \quad (5.12)$$

Beispiel: Instanziierung des Produktmodells. Wir geben ein kleines Beispiel an, in dem wir eine Instanziierung des Produktmodells mit Inhalten des V-Modell XT zeigen (Abbildung 5.9). Im oberen Teil der Abbildung ist der Ausschnitt des V-Modell XT gezeigt, auf den wir uns beziehen. Im unteren Teil die Instanziierung des Produktmodells mit einigen weiter gehenden Elementen.

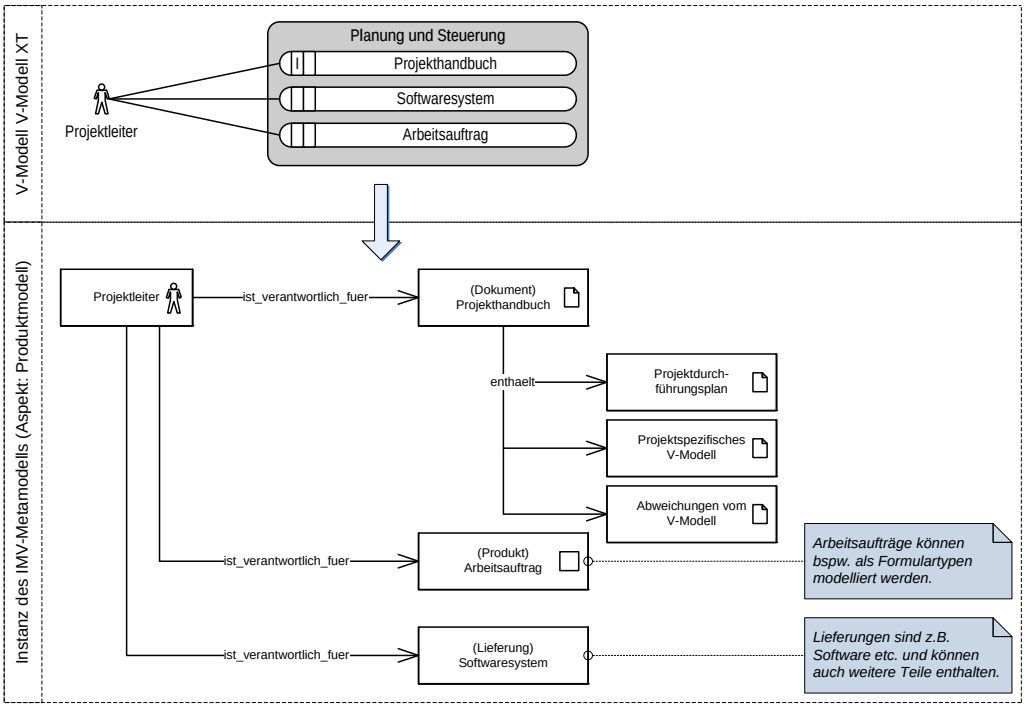

Abbildung 5.9.: Beispielhafte Anwendung des Produktmodells für Auszüge aus dem V-Modell XT

Im Beispiel ist mithilfe unseres Produktmodells ein Auszug aus dem V-Modell XT remodelliert. Zuerst ist das *Projekthandbuch* zu nennen, das wir als Dokument (der Anschaulichkeit halber entsprechend gekennzeichnet) modellieren. Das Projekthandbuch kann laut V-Modell Themen enthalten, wie zum Beispiel *Projektdurchführungsplan*, *Projektspezifisches V-Modell* oder *Abweichungen vom V-Modell*. Diese Themen modellieren wir als strukturierte Inhalte und verknüpfen sie mithilfe der Assoziation „*Enthaelt*". Ebenfalls betrachten wir das V-Modell-Produkt *Arbeitsauftrag*, welches wir nicht zwangsläufig als Dokument modellieren , sondern beispielsweise als Formular (siehe [KK07]). Analog wählen wir als drittes Beispiel das V-Modell-Produkt *Lieferung*. Auch dieses modellieren wir nicht als Dokument, sondern führen einen eigenen Produkttyp ein.

Aktivitätsmodell

In unserem Vorgehensmetamodell definieren wir ebenfalls ein minimales Aktivitätsmodell (Abbildung 5.10). Das Aktivitätsmodell haben wir sehr einfach gehalten, jedoch kann es analog zum Produktmodell erweitert werden. Zentrale Klasse ist die `Aktivitaet` als Kind von `ProzessElement` (vgl. Abbildung 5.6). Für `Aktivitaet` definieren wir nur zwei Kinder `Aufgabe` und `Arbeitspaket`. Wir verweisen hier auf das Metamodell des V-Modell XT, das hier eine ähnliche jedoch explizite Abstufung in Aktivitäten

und Teilaktivitäten vorsieht. Eine derartige Beschränkung wollen wir hier nicht einführen; auch betrachten wir alle Aufgaben als gleichberechtigt an. Arbeitspakete definieren wir als Container für mehrere Aufgaben (Kardinalität: $2..*$).

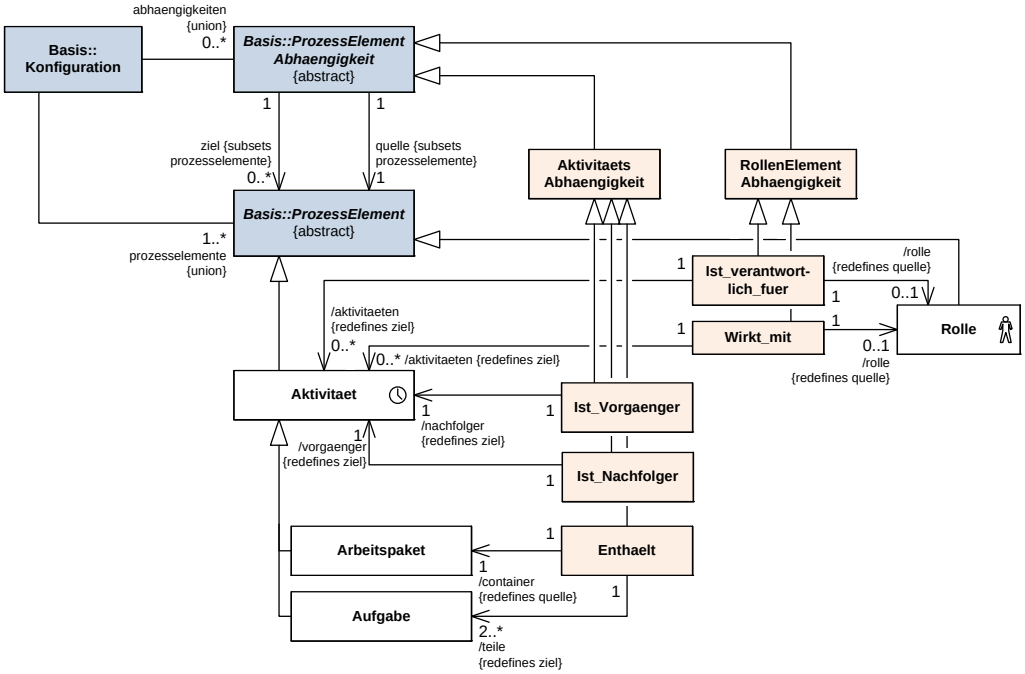

Abbildung 5.10.: Verfeinerung des Metamodells für den Kontext Aktivitätsmodell

Analog zum Produktmodell ist auch das Aktivitätsmodell abgeschlossen. Verknüpfungen zu Prozesselementen, die nicht vom Typ `Aktivitaet` oder spezieller sind, werden nur durch externe Konfiguration gestattet (wie zum Beispiel auch bei den Produkten gezeigt – Abbildung 5.10).

Beziehungstypen im Aktivitätsmodell. Auf ähnlicher Ebene wie Produktabhängigkeiten modellieren wir auch *Aktivitätsabhängigkeiten*. Vorgehensmodelle wie das V-Modell XT messen Aktivitäten vergleichsweise wenig Bedeutung zu, da sie Produkte favorisieren. Andere Vorgehensmodelle, wie beispielsweise OpenUP oder MSF (vgl. für beide Kapitel 2.1) favorisieren hingegen Aktivitäten und liefern hier eine entsprechend ausführliche Modellierung. Da wir in dieser Arbeit sowohl Aktivitäten als auch Produkte vom selben Basiskonzept ableiten, behandeln wir sie auch gleichberechtigt und definieren Aktivitätsabhängigkeiten (vgl. Abbildung 5.10). Für Aktivitäten modellieren wir folgende grundlegende Beziehungstypen als Spezialisierung der Klasse `AktivitaetsAbhaengigkeit` (Kind von `ProzessElementAbhaengigkeit`):

- `Enthaelt`
- `Ist_Nachfolger`
- `Ist_Vorgaenger`

Den Beziehungstyp `Enthaelt` modellieren wir analog zum Beziehungstyp `Enthaelt` über Produkten[10]. Als Elemente für diese Beziehung stehen uns gemäß Abbildung 5.10 Aufgaben und Arbeitspakete zur Verfügung, wobei die Aufgaben den strukturierten Inhalten und die Arbeitspakete den Dokumenten entsprechen. Wir definieren die Beziehung für $x \in$ *Arbeitspakete* und $y_1, \ldots, y_n \in$ *Aufgaben* also wie folgt:

$$\text{enthaelt}\,(x) = \{y_1, \ldots, y_n\}\,, \text{bzw. enthaelt}\,(x, \{y_1, \ldots, y_n\})\,, n \geq 2 \tag{5.13}$$

10 Im Rahmen einer Implementierung muss hier natürlich eine Eindeutigkeit hergestellt werden. Im Rahmen des Werkzeugkonzepts in Anhang C gehen wir darauf näher ein.

Für die Beziehungstypen `Ist_Vorgaenger` und `Ist_Nachfolger` verlangen wir wie bei der Produkterzeugung, dass es sich um gerichtete, einfache Assoziationen handelt. Für $x, y \in$ *Aktivitaeten* gilt also:

$$\text{ist\_vorgaenger}(x) = y, \text{bzw. ist\_vorgaenger}(x, y) \tag{5.14}$$

$$\text{ist\_nachfolger}(x) = y, \text{bzw. ist\_nachfolger}(x, y) \tag{5.15}$$

Wir können diese beiden Beziehungstypen so einfach gestalten, da wir Vorgänger-/Nachfolgerrelationen direkt über Aktivitäten definieren. Da wir Aktivitäten zum Beispiel auch als Arbeitspakete spezialisieren, führen wir damit implizit einen Kompositionsoperator für Aktivitäten ein. Betrachten wir folgendes Beispiel:

$$\begin{aligned} &p_1, p_2 \in \text{Arbeitspakete}, a_1, a_2, a_3, a_4, a_5 \in \text{Aufgaben} \wedge \\ &\text{enthaelt}(p_1, \{a_1, a_5\}) \wedge \text{ enthaelt}(p_2, \{a_2, a_3, a_4\}) \wedge \\ &\text{ist\_nachfolger}(p_1, p_2) \Rightarrow \\ &\forall a_i \in \text{enthaelt}(p_2) . \forall x_j \in \text{enthaelt}(p_1) : \text{ist\_nachfolger}(x_j, a_i) \end{aligned} \tag{5.16}$$

Für den Beziehungstyp `Ist_Vorgaenger` können wir eine analoge Modellierung angeben. Relationen der Containerelemente gelten auch für die enthaltenen Kindelemente. Analog zu den Produktabhängigkeiten sind auch die Aktivitätsabhängigkeiten innerhalb einer Prozesskomponente in der Regel vollständig.

Beispiel: Instanziierung des Aktivitätsmodells. Auch für die Instanziierung des Aktivitätsmodells wollen wir ein kleines Beispiel angeben. Der Rückgriff auf das V-Modell XT ist hier jedoch nur eingeschränkt möglich, da die Aktivitiäten des V-Modells erst bei der Plangenerierung mit dem Projektassistenten relevant werden.

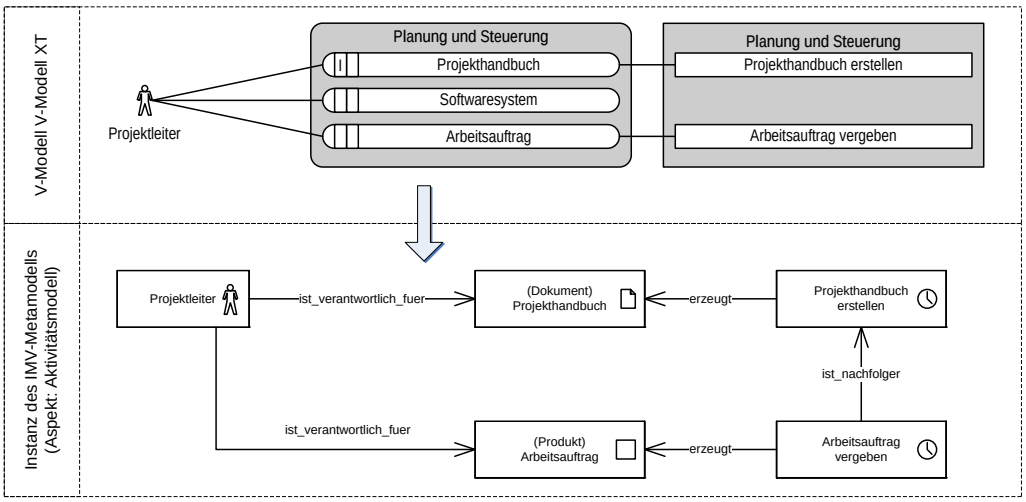

Abbildung 5.11.: Beispielhafte Anwendung des Aktivitätsmodells für Auszüge aus dem V-Modell XT

Das Beispiel greift die den beiden Produkten *Projekthandbuch* und *Arbeitsauftrag* zugeordneten Aktivitäten *Projekthandbuch erstellen* und *Arbeitsauftrag vergeben* auf. Ein direkte Verknüpfung der Aktivitäten sieht das V-Modell nicht vor. Abhängigkeiten und Nachfolgerrelationen werden durch die Ausgestaltung der Entscheidungspunkte und der durch diese definierten Produktvoluminar ermittelt. Direkte Verknüpfungen zwischen den Aktivitäten gibt es aber auch an dieser Stelle nicht. Die Überführung auf das IMV-Metamodell läßt beispielsweise eine Verknüpfung der beiden gezeigten Aktivitäten zu, sodass durch

$$\text{ist\_nachfolger}(\text{Arbeitsauftrag vergeben}, \text{Projekthandbuch erstellen})$$

ausgedrückt werden kann, dass zuerst das Projekthandbuch zu erstellen ist, bevor Arbeitsaufträge vergeben werden können.

Verknüpfung von Produkten und Aktivitäten

Die Beziehungstypen zwischen Prozesselementen erzeugen jeweils die Strukturen innerhalb der einzelnen Submodelle. Jedoch müssen die Submodelle auch untereinander in Beziehung gesetzt werden, um eine sinnvolle Ausgestaltung und letztendlich auch Anwendung eines Vorgehensmodells zu ermöglichen. Hierfür spezialisieren wir den Beziehungstyp `ProzessElementAbhaengigkeit` einmal für die Verknüpfung von Produkten und Aktivitäten (`ProduktAktivitaetAbhaengigkeit`) und weiterhin für die Verknüpfung eben dieser Elemente mit Rollen (`RollenElementAbhaengigkeit`, siehe weiter oben). Wir spezialisieren `ProduktAktivitaetAbhaengigkeit` in drei konkreten Beziehungstypen:

- Beziehung: `Erzeugt`
- Beziehung: `Erfordert`
- Beziehung: `Wird_bearbeitet`

Diese drei in dieser Arbeit definierten Beziehungstypen stellen einen minimalen Umfang dar, um Produkterstellung (`Erzeugt`), Produktfluss (`Erfordert`) und Produktbearbeitung (`Wird_bearbeitet`) zu ermöglichen. Von den grundlegenden Fähigkeiten orientieren wir uns wieder schwerpunktmäßig am V-Modell XT, schließen jedoch anders lautende Spezialisierungen auf der Grundlage anderer Vorgehensmodellphilosophien nicht aus. In Abbildung 5.12 haben wir die Modellierung der zu betrachtenden Beziehungstypen vorgenommen.

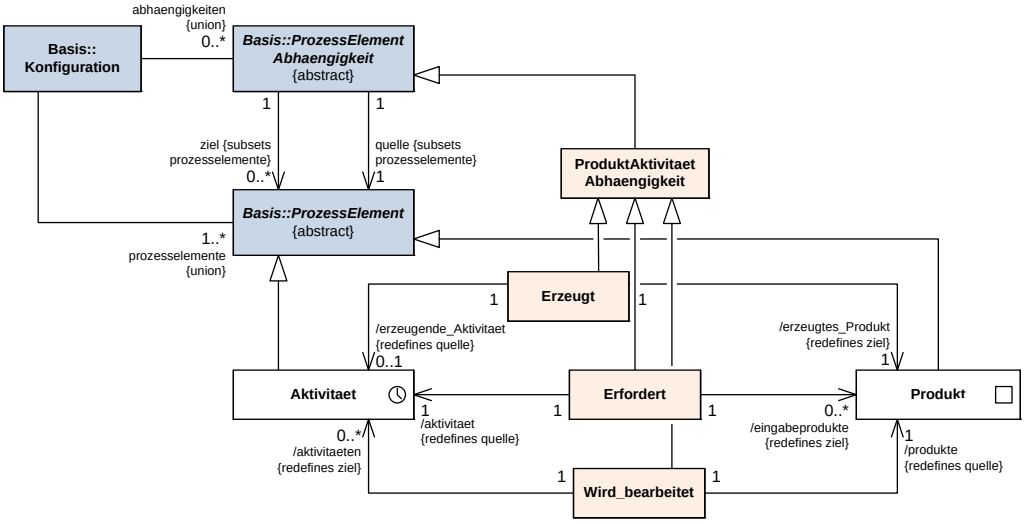

Abbildung 5.12.: Verfeinerung des Metamodells für den Kontext Produkt- und Aktivitätsabhängigkeiten

Beziehung: `Erzeugt` – Der Beziehungstyp `Erzeugt` legt eine „Konstruktionsoperation" für ein Produkt fest. Anders als die Produktabhängigkeit `Erzeugt` wird hier konkret festgelegt, welche (genau eine) Aktivität zur Erstellung eines Produkts führt. Die Produkterstellung wird somit beispielsweise im Kontext einer Projektplangenerierung zur Planungsgröße (vgl. [Mün01, Gna05]). Es gilt also für $a \in \mathit{Aktivitaeten}, p_1, p_2 \in \mathit{Produkte} \wedge p_1 \neq p_2$:

$$\text{erzeugt}\,(a, p_1) \Rightarrow \neg\exists p_2 : \text{erzeugt}\,(a, p_2) \tag{5.17}$$

Anders herum muss ein Produkt nicht unbedingt von einer Aktivität erzeugt werden. Analog zum V-Modell XT sprechen wir dann von einem *externen* Produkt[11].

11 Für die Modellierung von Produkten hat das natürlich zur Folge, dass ein entsprechendes Attribut vorzusehen ist, das eine entsprechende Unterscheidung zwischen externen und nicht externen Produkten erlaubt.

Ein Beispiel für ein derartiges Konstrukt findet sich beispielsweise dort, wo organisationsübergreifend Standardformulare zu erstellen sind, der jeweilige Erstellungsprozess jedoch spezifisch durch eine Anpassung geregelt werden muss. Ein weiterer Fall der zu berücksichtigen ist, ist wenn die Beziehung zwischen einem *Arbeitspaket* und einem Produkt instanziiert wird. Analog zur Vorgänger- und Nachfolgerrelation gilt auch hier, dass Beziehungen des Containers transitiv für die Containerelemente gelten. Für ein Arbeitspaket sind dabei $n + 1$ Aktivitätstypen zu betrachten, wobei einer ausgezeichnet ist. Für die Erzeugung gilt daher:

$$\begin{aligned} &\text{erzeugt}\,(A, p)\ \wedge A \in \text{Arbeitspakete} \Rightarrow \\ &\text{abgeschlossen}\,(A) := \bigwedge_{\forall i} \text{abgeschlossen}\,(a_i) \end{aligned} \tag{5.18}$$

Alle Aktivitäten (Aufgaben) a_i werden somit im Kontext eines Projektplans unter einem Sammelvorgang zusammengefasst. Das Prädikat *abgeschlossen* bildet eine abschließende Bewertung von a_i auf einen boolschen Wert ab. Eine Konjuktion verlangt, dass alle Aufgaben des Arbeitspakets erfolgreich abgeschlossen sind, um die Produkterzeugung ebenfalls abzuschließen. Gnatz [Gna05] (Seite 135 ff.) geht hierauf aus Sicht der Plangenerierung noch detaillierter ein.

Beziehung: `Erfordert` – Der Beziehungstyp `Erfordert` dient dazu, Eingaben für Aktivitäten zu modellieren. In Kombination mit dem Beziehungstyp `Erzeugt` können wir damit auch *Produktflüsse* modellieren. Ein Produktfluss unterscheidet sich von einer erzeugenden Produktabhängigkeit in sofern, als dass hier die Erstellung eines Produkts durch eine Aktivität erfolgt, die explizit andere Produkte als Eingabe erfordert. Ein Anwendungsfall findet sich beispielsweise im V-Modell XT beim Verfahren am Entscheidungspunkt. Dieses Verfahren zur Projektfortschrittsbestimmtung ist im V-Modell nur semiformal und über viele Elemente verstreut definiert. Wir modellieren `Erfordert` wieder als $n + 1$-stellige Assoziation mit $n \geq 1$ für $a \in$ *Aktivitaeten*, $p_1, \ldots, p_n \in$ *Produkte*:

$$\text{erfordert}\,(a, \{p_1, \ldots, p_n\}) \tag{5.19}$$

Über das Resultat von a sagen wir an dieser Stelle nichts aus. Durch Verknüpfung erzeugen wir dann einen Produktfluss:

$$\text{erzeugt}\,(a, p_0)\ \wedge \text{erfordert}\,(a, \{p_1, \ldots, p_n\}) \tag{5.20}$$

Als konkretes Beispiel beziehen wir uns auf den V-Modell Entscheidungspunkt *Projekt definiert*, zu dem mindestens ein Projekthandbuch (PHB), ein QS-Handbuch (QSHB) und ein Projektplan (PL) vorgelegt werden müssen. Aufgrund dieser Eingaben wird eine Projektfortschrittsentscheidung herbeigeführt, die als Ergebnis das Produkt Projektfortschrittsentscheidung (PFE) hat. Wir modellieren das wie folgt:

$$\begin{aligned} &\text{erfordert}\,(\text{PFE herbeiführen}, \{\text{PHB}, \text{QSHB}, \text{PL}\})\ \wedge \\ &\text{erzeugt}\,(\text{PFE herbeiführen}, \text{PFE}) \end{aligned}$$

Beziehung: `Wird_bearbeitet` – Der Beziehungstyp `Wird_bearbeitet` dient uns dazu, planbare Operationen auf Produkten zu modellieren. Im V-Modell zum Beispiel dienen Aktivitäten ausschließlich der Fertigstellung von Produkten, d. h. Details zur Produktbearbeitung sind transparent. Aktivitätsorientierte Vorgehensmodelle wie MSF treffen andererseits sehr detaillierte Aussagen darüber, welche Aktivitäten ein Produkt im Laufe seines Lebenszyklus bearbeiten können. Beispielhaft sei hier ein Modell, zum Beispiel ein Entwurfsmodell genannt, dass erstellt, geprüft, aktualisiert und ggf. sogar bewiesen wird. Um dies auf der Metamodellebene bereits vorzusehen, definieren wir den Beziehungstyp `Wird_bearbeitet` für Aktivitäten $a_1, \ldots, a_n$ und ein Produkt p wie folgt:

$$\text{wird\_bearbeitet}\,(p, \{a_1, \ldots, a_n\}) \tag{5.21}$$

An dieser Stelle ist es für uns dann auch unerheblich, zu welchem Aktivitätstyp $a_1, \ldots, a_n$ gehören.

Viele weitere Beziehungstypen zwischen Produkten und Aktivitäten sind denkbar, ähnlich wie diverse Einschränkungen, die für Spezialisierungen dieser beiden Konzepte denkbar sind. Um jedoch eine so hohe Komplexität im möglichen Abhängigkeitsgeflecht zu vermeiden, wie sie zum Beispiel bei EPF/UMA (Kapitel 3.2.3 und 3.4) zu finden ist, begnügen wir uns mit dieser kleinen Menge Beziehungstypen[12]. Die möglichen Kombinationsmöglichkeiten zwischen Einzelelementen sind somit zwar eingeschränkt, können jedoch durch die Kombinierbarkeit der Einzelprädikate sehr schnell eine hohe Komplexität und Integration erreichen.

Zwischenstand. Alle Beziehungstypen, die wir bisher definiert haben dienen uns zur internen Konfiguration für Prozesskomponenten. Einige dieser Beziehungstypen sind jedoch im Rahmen der Komposition von Vorgehensmodellen aus Prozesskomponenten ebenfalls erforderlich. Hierfür fassen wir zunächst noch einmal die definierten Beziehungstypen zusammen (Tabelle 5.1) und widmen uns anschließend dem Themenkomplex der internen und externen Abhängigkeiten zur Bildung von Prozesskomponenten.

Name	Quelltyp	Zieltyp	Typ und Bemerkungen
`Erzeugt`	Produkt	Produkt	$1:1$ Beziehung
`Enthaelt`	Produkt	Produkt	$1:n$ über Spezialisierungen; Container und Hierarchien sind erforderlich, hier: Dokument enthält strukturierte Inhalte.
`Enthaelt`	Aktivität	Aktivität	$1:n$ über Spezialisierungen; Container und Hierarchien sind erforderlich, hier: Arbeitspakete enthalten Aufgaben (min. 2).
`Ist_Vorgaenger`	Aktivität	Aktivität	$1:1$ Beziehung; transitiv für Container und enthaltene Elemente
`Ist_Nachfolger`	Aktivität	Aktivität	$1:1$ Beziehung; transitiv für Container und enthaltene Elemente
`Ist_verantwortlich_fuer`	Rolle	Aktivität	$1:n$ Beziehung
`Ist_verantwortlich_fuer`	Rolle	Produkt	$1:n$ Beziehung
`Wirkt_mit`	Rolle	Aktivität	$1:n$ Beziehung
`Wirkt_mit`	Rolle	Produkt	$1:n$ Beziehung
`Wirkt_mit_an_teil`	Rolle	Produkt	$1:n$ Beziehung; für strukturierte Inhalte
`Erzeugt`	Aktivität	Produkt	$1:1$ Beziehung mit Ausnahme der externen Produkte
`Erfordert`	Aktivität	Produkt	$1:n$ Beziehung für die Modellierung von Produktflüssen
`Wird_bearbeitet`	Produkt	Aktivität	$1:n$ Beziehung

Tabelle 5.1.: Beziehungstypen des Vorgehensmetamodells (Zwischenstand und Übersicht)

Wie in Kapitel 4 definiert, stehen uns mit der *einfachen* und der *erweiterten Prozessele-*

12 Im Rahmen von Erweiterungen und Anpassungen besteht hier ggf. sowieso die Notwendigkeit der Anpassung oder Ergänzung.

mentabhängigkeit ausreichend mächtige Konstrukte für die Modellierung der Abhängigkeitsstrukturen zur Verfügung. Die in Tabelle 5.1 zusammengestellten Beziehungstypen genügen, um sämtliche relevanten Beziehungen zwischen einzelnen Prozesselementen in einer Prozesskomponente auszudrücken. Gemäß unseren Überlegungen aus Kapitel 4.2.1 erhalten wir so einen *gerichteten Abhängigkeitsmultigraphen*. Wir vertiefen im Folgenden die Frage nach der Grenze zwischen der Konfiguration einer Prozesskomponente, ihren Schnittstellen sowie externen Abhängigkeiten und korrespondierenden Konfigurationen von Vorgehensmodellen.

5.1.3. Prozesskomponenten und Vorgehensmodelle

Wir widmen uns nun der Zusammenstellung von Vorgehensmodellen auf Basis der gerade modellierten Prozesskomponenten. Gemäß unseren Vereinbarungen aus Kapitel 4 fixieren wir die Hierarchieebenen der Vorgehensmetamodellarchitektur und sehen *keine* Komposition von (elementaren) Prozesskomponenten zu Prozesskomponenten höherer Integration vor. In Abbildung 5.13 verfeinern wir das Modell aus Abbildung 5.4 und befassen uns nun mit einigen Fragestellungen der Konfiguration von Prozesskomponenten im Kontext eines Vorgehensmodells. Dabei betrachten wir in diesem Abschnitt zunächst nur die Beziehungstypen zwischen Prozesskomponenten und widmen uns dabei insbesondere den Schnittstellen, die externe Abhängigkeiten verursachen oder befriedigen.

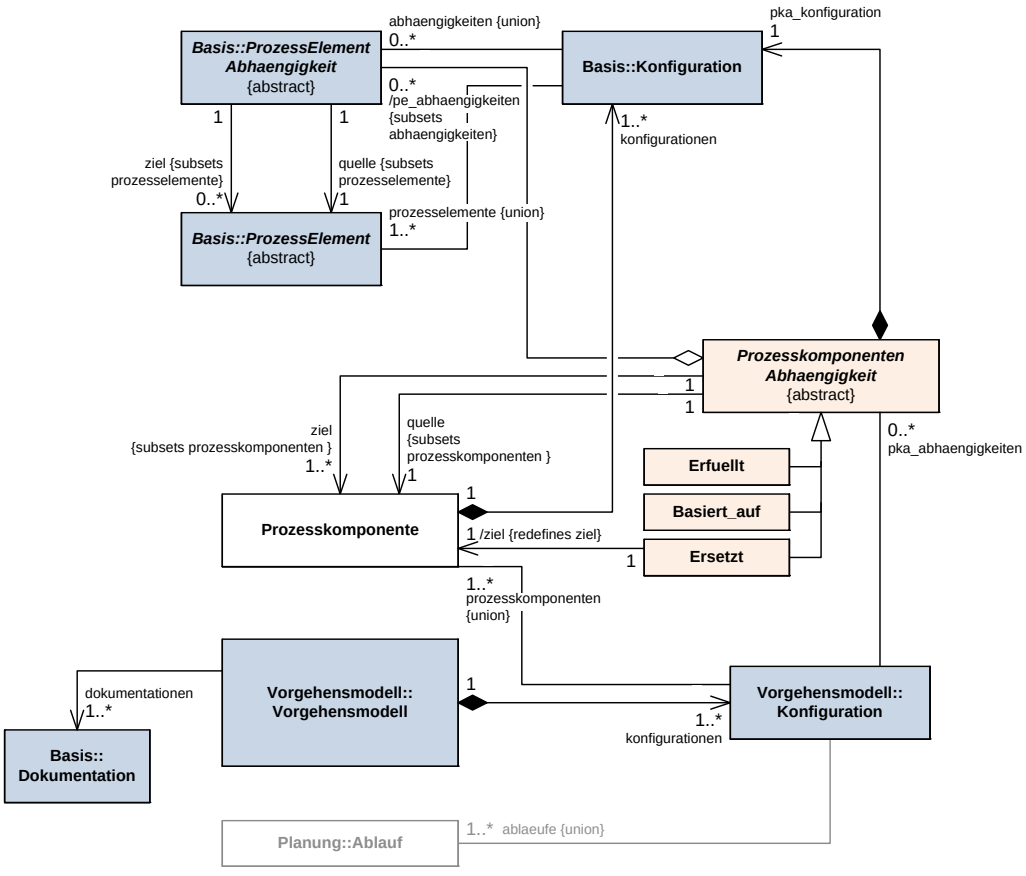

Abbildung 5.13.: Verfeinerung des Metamodells für Vorgehensmodelle mit Beziehungstypen

Beziehungstypen. Für Prozesskomponenten sehen wir analog zu Prozesselementen Beziehungstypen vor, die Relationen über der Menge der Prozesskomponenten definieren. In Kapitel 4.2.2 haben wir dafür entsprechende Relationen eingeführt. Die Be-

ziehungstypen zwischen Prozesskomponenten werden dabei wieder auf Beziehungstypen zwischen Prozesselementen zurückgeführt. Die Beziehungstypen legen dabei Gültigkeiten und Semantik fest. In diesem Abschnitt beschreiben wir zunächst die Beziehungstypen für Prozesskomponenten näher und bauen im folgenden Abschnitt 5.1.5 darauf auf und betrachten dann das Konfigurationsgesamtkonzept. Für Prozesskomponenten definieren wir drei Beziehungstypen:

- Beziehung: `Erfuellt`
- Beziehung: `Ersetzt`
- Beziehung: `Basiert_auf`

Mit diesen drei Beziehungstypen können wir alle relevanten Verknüpfungen zwischen Prozesskomponenten modellieren. Prozesskomponenten liegen uns als (weitgehend) abgeschlossene und vollständige Einheiten vor. Ihre interne Struktur ist definiert.

Schnittstellen von Prozesskomponenten. Durch die Konfiguration von Prozesselementen definieren Prozesskomponenten eine sehr detaillierte Schnittstelle, die alle Elemente und alle Beziehungen beschreibt. Dem Komponentenbegriff der Software Architektur (und hier insbesondere Siedersleben [Sie04]) folgend, können Schnittstellen für Komponenten *angeboten* oder *angefordert* sein. Bei der Verknüpfung von Prozesskomponenten müssen wir dies ebenfalls berücksichtigen. Wir nehmen folgende Fallunterscheidung vor:

1. Wir betrachten ausschließlich *konkrete* Prozesskomponenten. Dann existieren nur *angebotene*, jedoch keine *angeforderten* Schnittstellen und ein Vorgehensmodell kann durch einfache Konfiguration der Einzelkomponenten gebildet werden. Vorbedingungen existieren in diesem Fall nicht. Die Beziehungstypen `Basiert_auf` und `Ersetzt` sind anwendbar.
2. Wir betrachten auch *abstrakte* Prozesskomponenten, sodass es *angeforderte* Schnittstellen gibt. Als Vorbedingungen sind hier zunächst die angeforderten Schnittstellen eventuell genutzter Prozesskomponenten zu erfüllen. Dann sind insbesondere die Beziehungstypen `Erfuellt` und `Basiert_auf` zu berücksichtigen und zu realisieren.

Abbildung 5.13 zeigt den Ausschnitt des Metamodells, der die Beziehungstypen für Prozesskomponenten verfeinert. Für den allgemeinen Beziehungstyp (Klasse `ProzesskomponentenAbhaengigkeit`) gilt: Eine Konfiguration auf der Ebene des Vorgehensmodells referenziert alle für das jeweilige Vorgehensmodell relevanten Prozesskomponenten[13]. Zusätzlich enthält eine Konfiguration für Vorgehensmodelle eine Menge von Prozesskomponentenabhängigkeiten. Diese sind über der Menge der Prozesskomponenten definiert, die für ein Vorgehensmodell konfiguriert (also in der Konfiguration referenziert sind) werden. Eine Prozesskomponentenabhängigkeit referenziert alle beteiligten Prozesskomponenten und stellt zwischen diesen Beziehungen her. Da es sich hierbei um Beziehungen auf der Ebene von Prozesselementen handelt, werden hier neue Beziehungen zwischen den Prozesselementen der referenzierten Prozesskomponenten hergestellt.

Beispiel: Nehmen wir als Beispiel ein Auftraggeberprojekt (AG) des V-Modell XT: Das Produkt *Projekthandbuch* ist dort im Vorgehensbaustein Projektmanagement definiert. Das Produkt *Anforderungen (Lastenheft)* ist im Vorgehensbaustein Anforderungsfestlegung definiert. Es gibt eine Beziehung zwischen diesen beiden Vorgehensbausteinen, nämlich Anforderungsfestlegung basiert auf Projektmanagement. Weiterhin ist im *Projekthandbuch* ein Thema *Organisation und Vorgaben zum Anforderungsmanagement* enthalten, welches aber im Vorgehensbaustein Anforderungsfestlegung definiert ist, dem *Projekthandbuch* jedoch hinzugeführt wird.

Für unseren Kontext betrachten wir zwei Prozesskomponenten Projektmanagement und Anforderungsmanagement, die in einer *basiert_auf*-Beziehung stehen. In der betreffenden Konfiguration ist diese Beziehung hinterlegt und beide Prozesskomponen-

13 An dieser Stelle wir auch die Unterscheidung zwischen Vorgehensmodell und Vorgehensmodellvariante hinfällig, da die Unterscheidung ausschließlich über unterschiedliche Konfigurationen getroffen wird. Wir sprechen auf dieser Ebene ausschließlich von Vorgehensmodellvarianten.

ten (und nur diese beiden) können nun miteinander verknüpft werden, indem Beziehungen zwischen in ihnen enthaltenen Prozesselementen hergestellt werden. In der Konfiguration der Prozesskomponentenabhängigkeit zwischen Projektmanagement und Anforderungsmanagement ist also das Produkt (hier modelliert als Dokument) *Projekthandbuch* zu referenzieren und um den strukturierten Inhalt *Organisation und Vorgaben zum Anforderungsmanagement* zu erweitern.

Eine nach diesem Muster hergestellte Beziehung zwischen Elementen zweier oder mehrerer Prozesskomponenten ist auch nur für den Kontext dieser Prozesskomponentenabhängigkeit gültig. Wird die Prozesskomponentenabhängigkeit aus einer Konfiguration entfernt, werden auch sämtliche Verknüpfungen entfernt. Im Folgenden gehen wir detailliert auf die oben aufgeführten, speziellen Beziehungstypen ein:

Beziehung: `Basiert_auf` – Dieser Beziehungstyp dient im Wesentlichen der Nachnutzung und Wiederverwendung von Prozesskomponenten. Wir orientieren uns hier am gleichnamigen Konzept des V-Modell XT (Kapitel 3). Eine Prozesskomponente pk_0 kann auf n weiteren Prozesskomponenten basieren. Wir notieren sofort in vereinfachter Form:

$$\begin{aligned}&\text{basiert\_auf}\,(pk_0) = \{pk_1, \ldots, pk_n\}\,, \text{bzw.}\\ &\text{basiert\_auf}\,(pk_0, \{pk_1, \ldots, pk_n\})\end{aligned} \tag{5.22}$$

Dies hat zur Folge, das Inhalte aus pk_0 und Inhalte aus $pk_1, \ldots, pk_n$ sich referenzieren können. Zur Verfügung stehen hier sämtliche Prozesselemente und Prozesselementabhängigkeiten die wir in den vorangegangenen Abschnitten beschrieben haben. Dabei ist bei der Anwendung dieser Version darauf zu achten, dass gelten muss:

$$\forall pk \in \{pk_1, \ldots, pk_n\} : pk\,|_{G,abs} = \emptyset \tag{5.23}$$

Keine durch `Basiert_auf` referenzierte Prozesskomponente darf *abstrakt* sein, d. h. keine referenzierte Prozesskomponente darf eine Schnittstelle anfordern. Wird durch eine Prozesskomponente eine Schnittstelle angefordert, ist anstelle einer Basierungs- eine *Erfüllungs*beziehung anzugeben.

Beziehung: `Erfuellt` – Eine Prozesskomponente darf explizit über *angeforderte Schnittstellen* verfügen, d. h. konkrete Funktionalität von einer spezialisierenden beziehungsweise ergänzenden Prozesskomponente anfordern. Für Prozesskomponenten $pk_0, \ldots, pk_n$ schreiben wir wieder:

$$\text{erfuellt}\,(pk_0) = \{pk_1, \ldots, pk_n\}\,, \text{bzw. erfuellt}\,(pk_0, \{pk_1, \ldots, pk_n\}) \tag{5.24}$$

wenn die Prozesskomponente pk_0 die angeforderten Schnittstellen der Prozesskomponenten $pk_1, \ldots, pk_n$ befriedigt. In den Prozesskomponenten $pk_1, \ldots, pk_n$ gibt es also Beziehungen, deren Ende nicht gültig ist (wir schreiben dafür einfach $\perp$, also zum Beispiel *enthaelt* $(a, \perp)$). Betrachten wir alle abstrakten Prozesskomponenten pk_i in einer solchen Verknüpfung, dann gilt:

$$G_{VgM,abs} = \bigcup_{1 \leq i \leq n} pk_i\,|_{G,abs} = \left(Pe_{abs}, (PEA \cup EPEA)_{abs}\right)$$

ist der Abhängigkeitsgraph aller referenzierten Prozesskomponenten $pk_1, \ldots, pk_n$. Gemäß unserem Beispiel aus Kapitel 4.2.2 ist dieser Graph nicht (zwangsweise) zusammenhängend. Der Zusammenhang wird durch den Beziehungstyp `Erfuellt` in der Prozesskomponente pk_0 hergestellt. Anders als bei der einfachen Verknüpfung von Graphen durch neue Kanten zwischen den Teilgraphen wird die Verknüpfung hier durch Substitution der „ungültigen" Knoten $\perp$ durch entsprechende Knoten aus pk_0 hergestellt. Die offenen Knoten sind bestimmt durch $\rho \subseteq (PEA \cup EPEA)_{abs}$, sodass für alle ρ gilt: $\rho = (x, \perp) \vee \rho = (x, Y) \wedge \perp \in Y$. Seien ferner die für die Substitution verfügbaren Prozesselemente aus pk_0 gegeben durch Pe_{pk_0}. Es gilt für eine konfigurierte `Erfuellt`-Beziehung:

$$\forall \rho . \exists \alpha \in Pe_{pk_0} : \text{subst}\,(\perp, \alpha) \tag{5.25}$$

Für alle existierenden ungültigen Knoten muss bei der Konfiguration ein gültiger Knoten zur Substitution angegeben werden. Eine Operation *subst* übernimmt dies. Es entstehen hierbei *keine* neuen Kanten im Graph; vielmehr werden existierende Kanten mit gültigen Zielen vervollständigt. Die Beziehung `Erfuellt` zwischen n-Prozesskomponenten muss also sicher stellen, dass keine illegalen Knoten mehr im resultierenden Gesamtabhängikeitsgraph vorhanden sind.

Aufgrund der Typisierung der Prozesselementabhängigkeiten, können hier sehr feingranulare Anforderungen beschrieben werden. Beispielhaft verweisen wir auf den Beziehungstyp `Erfordert` zwischen Aktivitäten und Produkten. Eine Aktivität kann beispielsweise bestimmte Eingangsprodukte erfordern, die konkrete Belegung jedoch verlagern. Eine beispielhafte Formulierung für eine Aktivität aus einer Prozesskomponente, die ein Eingangsprodukte aus einer anderen Prozesskomponenten benötigt, hat folgendes Aussehen:

$$\begin{aligned}&\text{erfordert}\,(akt_0, \{p_1, \ldots, p_i, \bot_{akt_0}, p_j, \ldots, p_n\})\\&\ldots\\&\text{subst}\,(\bot_{akt_0}, p_\alpha)\end{aligned}$$

Ein konkretes Anwendungsfeld finden wir beispielsweise in Organisationen, in denen generische Verfahren beschrieben und durch Abteilungen konkret umzusetzen sind (zum Beispiel Berichte, Meldungen, Aufträge etc.).

Beziehung: `Ersetzt` – Der letzte zu betrachtende Beziehungstyp ist der Typ `Ersetzt`. Dieser Beziehungstyp dient dazu, einzelne Prozesskomponenten gegen Prozesskomponenten mit kompatiblen Schnittstellen (in der Regel Weiterentwicklungen) zu ersetzen. Der Beziehungstyp `Ersetzt` ist der einzige der hier besprochenen, der ausschließlich 2-stellig ist. Eine kumulative Ersetzung, d. h. eine Prozesskomponente ersetzt n andere, lassen wir *nicht* zu. Wir definieren also:

$$\text{ersetzt}\,(pk_0, pk_1) \tag{5.26}$$

für zwei Prozesskomponenten pk_0 und pk_1 genau dann wenn sichergestellt ist, dass:

$$pk_1\,|_G \subseteq pk_0\,|_G \tag{5.27}$$

Dies schließt natürlich nicht aus, dass die Schnittstelle beziehungsweise das allgemeine Leistungsangebot von pk_0 umfassender sein kann, als das von pk_1. Wir ziehen uns hier auf komponentenorientierte Techniken zurück, die für Software durch so genannte *Schnittstellenvererbung* (zum Beispiel [HR02]) realisiert werden.

Analog zu Tabelle 5.1 fassen wir in Tabelle 5.2 noch einmal die vorgestellten Beziehungstypen für Prozesskomponenten zusammen.

Name	Kardinalität	Bemerkungen
`Basiert_auf`	$1:n$	Anwendbar auf alle Prozesskomponenten, die nicht abstrakt sind. Die Nutzung der aggregierenden Prozesskomponente hat automatisch die Mitnutzung der aggregierten Prozesskomponenten zur Folge.
`Erfuellt`	$1:n$	Nachbedingung: Alle referenzierten Prozesskomponenten sind befriedigt. Die erfüllende Prozesskomponente ist nicht abstrakt.
`Ersetzt`	$1:1$	Kummulative Ersetzung ist verboten. Transitives Ersetzen ist jedoch möglich.

Tabelle 5.2.: Beziehungstypen des Vorgehensmetamodells für Prozesskomponenten

Beispiel: Instanziierung des Metamodells für Prozesskomponenten. Nachdem wir die einzelnen Beziehungstypen für Prozesskomponenten eingeführt haben, wollen wir noch ein kurzes Beispiel geben. Betrachten wir das System von Prozesskomponenten $pk_0, \ldots, pk_5$ in Abbildung 5.14 (oben). pk_4 ist eine abstrakte Prozesskomponente, die eine angeforderte Schnittstelle definiert. Diese Prozesskomponente kann beispielsweise einen allgemeinen Qualitätssicherungsprozess abteilungsübergreifend definieren, der durch Abteilungen konkretisiert werden muss, zum Beispiel: allgemeine QS vs. QS für Software oder QS für Hardware oder QS für integrierte Soft- und Hardware. Diese Spezialisierung nimmt in unserem Beispiel pk_2 vor. Im mittleren Teil der Abbildung haben wir dies inform der UML 2-Notation für Komponenten und deren Schnittstellen noch einmal gesondert hervorgehoben. Es existiert hier also eine Beziehung *erfuellt* (pk_2, pk_4) zwischen diesen beiden Prozesskomponenten. Sie wird durch ein Paar Schnittstellen realisiert, die wie oben diskutiert die Ersetzung ungültiger Knoten vornehmen. In der angeforderten Schnittstelle von pk_4 sind dabei die zu erfüllenden Abhängigkeiten ρ enthalten, während in der erfüllenden Schnittstelle diejenigen Prozesselmente enthalten sind, die für die Substitution der $\perp$-Knoten verwendet werden.

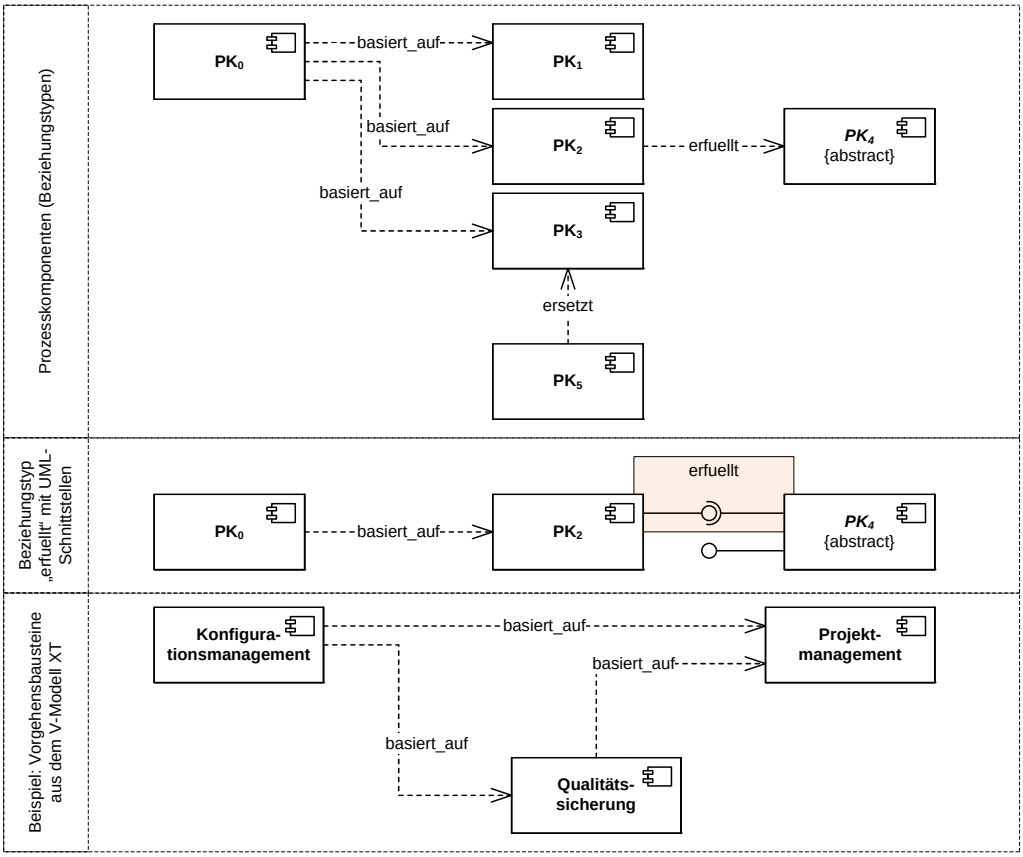

Abbildung 5.14.: Beispiel die Anwendung von Prozesskomponentenabhängigkeiten

Die Prozesskomponente pk_0 basiert auf drei weiteren Prozesskomponenten, sodass hier eine Beziehung: *basiert_auf* $(pk_0, \{pk_1, pk_2, pk_3\})$ existiert. Für pk_3 ist darüber hinaus noch zu beachten, dass mit pk_5 eine äquivalente Ersetzungsmöglichkeit existiert, also eine Beziehung *ersetzt* (pk_5, pk_3). Beispielhaft könnten wir für pk_3 annehmen, dass es die Prozesselemente enthält, die OO-Programmierung mit Java beschreiben, während pk_5 OO-Porgrammierung mit .NET beschreibt.

Im unteren Teil der Abbildung 5.14 finden wir eine beispielhafte Anwendung des V-Modell XT auf das Konzept der Prozesskomponente. Basiskomponente hier ist der Vorgehensbaustein *Projektmanagement* (PM), auf dem sowohl *Qualitätssicherung* (QS) als auch *Konfigurationsmanagement* (KM) aufsetzen. Konfigurationsmanagement basiert

weiterhin auf Qualitätssicherung. In unserer Modellierung würde sich das wie folgt darstellen:

$$\text{Prozesskomponenten} = \{\text{PM}, \text{QS}, \text{KM}\}$$
$$\text{basiert\_auf}\,(\text{QS}, \text{PM})$$
$$\text{basiert\_auf}\,(\text{KM}, \{\text{QS}, \text{PM}\})$$

Bereits auf Grundlage von Abbildung 5.14 (oben) können schon zwei verschiedene Konfigurationen angeben, die wir im Kontext eines Vorgehensmodells bilden können. Die verschiedenen Beziehungstypen zwischen den einzelnen Prozesskomponenten sind dabei eine grobgranulare Sicht. In Abhängigkeit vom Beziehungstyp sind die Konfigurationen frei oder nur unter bestimmten Vor- und Nachbedingungen möglich (vgl. Tabelle 5.2). Wenden wir zum Beispiel die Option *ersetzt* $\left(pk_5, pk_3\right)$ nicht an, enthält unsere Vorgehensmodellvariante die Prozesskomponenten pk_0, pk_1, pk_2, pk_3 und pk_4 sowie alle enthaltenen Prozesselemente. Wenden wir hingegen *ersetzt* $\left(pk_5, pk_3\right)$ an, entfallen die Prozesselemente aus pk_3 vollständig und werden durch diejenigen aus pk_5 ersetzt, wobei sicherzustellen ist, dass die Beziehung *basiert_auf* $\left(pk_0, pk_3\right)$ geprüft werden muss, da:

$$\text{basiert\_auf}\left(pk_0, pk_3\right) \rightarrow$$
$$\text{basiert\_auf}\left(pk_0, \text{ersetzt}\left(pk_5, pk_3\right)\right) \rightarrow$$
$$\text{basiert\_auf}\left(pk_0, pk_5\right)$$

Die (inhaltliche) Konsistenz dieser Ersetzung muss im Nachhinein sicher gestellt werden.

5.1.4. Variabilität in Vorgehensmodellen

Nachdem wir bereits die Beziehungstypen für Prozesselemente und Prozesskomponenten vorgestellt haben, greifen wir nun wieder die Beziehungstypen auf, die die Operation für die Variabilität realisieren. Diese gewinnen nun an Bedeutung, da wir auf der Ebenen der Prozesskomponenten nun Beziehungen zwischen den enthaltenen Elementen herstellen können. Die Anwendung von Variabilitätsoperationen ist hier vorzunehmen – hier werden im Rahmen der Anwendung und Ausgestaltung eines Vorgehensmodelles auch Prozessingenieure Variabilitäten einfügen.

[[Vervollständigen...]]

5.1.5. Konfigurationen – Ein Beispiel

Mit diesem Beispiel sind wir im Problembereich der *Konfigurationen* angelangt, den wir uns in diesem Abschnitt widmen. Wir haben bislang die verschiedenen Assoziationstypen für Prozesselemente (Tabelle 5.1) und Prozesskomponenten (Tabelle 5.2) ausführlich diskutiert. Mit diesen Assoziationstypen können wir nun bereits:

- beliebige Strukturen auf der Basis von Prozesselementen und Prozesselementabhängigkeiten modellieren.
- Prozesskomponenten eines Vorgehensmodells inklusive der relevanten Prozesselemente und deren Abhängigkeitsstrukturen modellieren.
- Konfiguration von Prozesskomponenten modellieren.

Für die allgemeinen Aussagen zu Prozesselementen und Prozesselementabhängigkeiten verweisen wir auf Kapitel 4, in dem wir unser mathematisches Basismodell entwickelt haben. Eine Konkretisierung dieses Basismodells haben wir für Vorgehensmodelle in den vorangegangenen Abschnitten erstellt. Mit diesem Abschnitt widmen wir uns der integrierten Betrachtung der verschiedenen *Konfigurationsebenen* für Prozesselemente und Prozesskomponenten. Wir greifen dabei auf einen kleinen Auszug des V-Modell XT als Beispiel zurück, dass wir im Anhang B verwenden, um V-Modell-Elemente mit unserem Modell zu remodellieren. Wir gehen dabei schrittweise vor und betrachten:

- Prozesskomponenten: Enthaltene Prozesselemente und deren Abhängigkeiten
- Vorgehensmodell: Enthaltene Prozesskomponenten und deren Abhängigkeiten

Modellierung einer Prozesskomponente auf Basis eines Vorgehensbausteins

Wir wollen zunächst einen Vorgehensbaustein aus dem V-Modell herausgreifen und exemplarisch eine Prozesskomponente modellieren. Da hier ein Element realer Komplexität vorliegt abstrahieren wir auch in diesem Beispiel, um es überschaubar zu halten. So verzichten wir auf eine vollständige Modellierung und greifen nur einige wenige, repräsentative Elemente heraus.

Als Beispiel wählen wir den Vorgehensbaustein *Projektmanagement*, der in grober Struktur in Abbildung 5.15 zu sehen ist. Wir greifen die Inhalte dieses Vorgehensbausteins auf und modellieren ihn als Prozesskomponente.

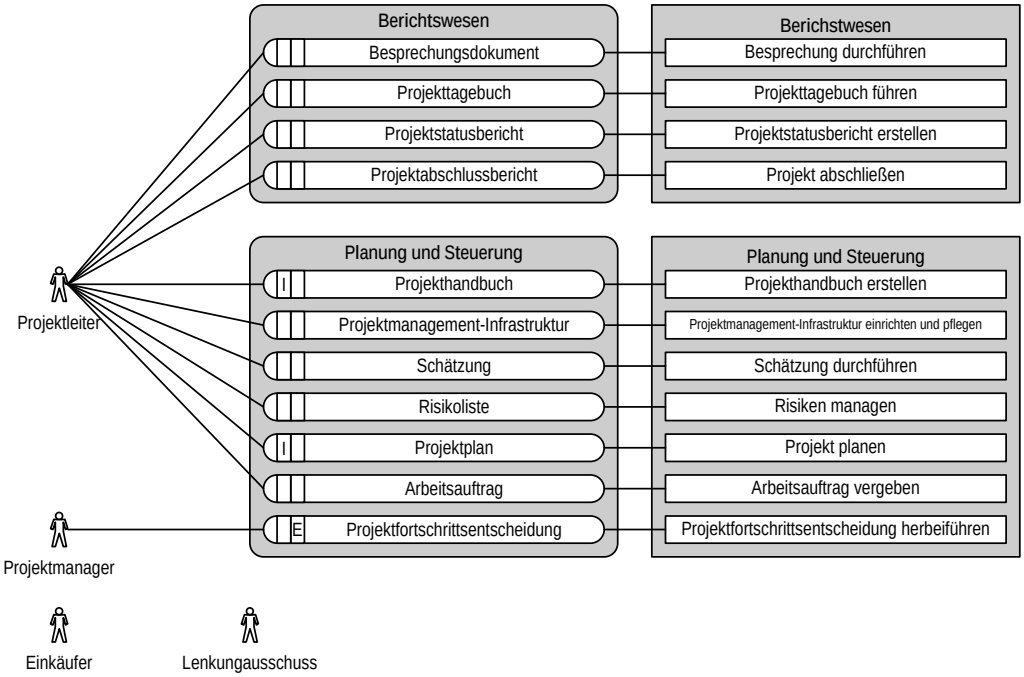

Abbildung 5.15.: Vorgehensbaustein Projektmanagement des V-Modell XT

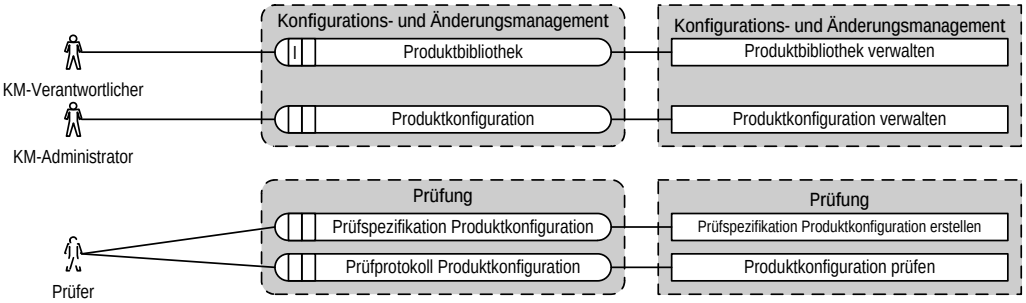

Abbildung 5.16.: Vorgehensbaustein Konfigurationsmanagement des V-Modell XT

Wir ermitteln zuerst auf der Basis der Produkte, Themen, Aktivitäten und Teilaktivitäten sowie der Rollen die zu modellierenden Prozesselemente und deren Abhängigkeiten. Diese resultieren dann in einer Konfiguration der Prozesskomponente.

[[vervollständigen]]

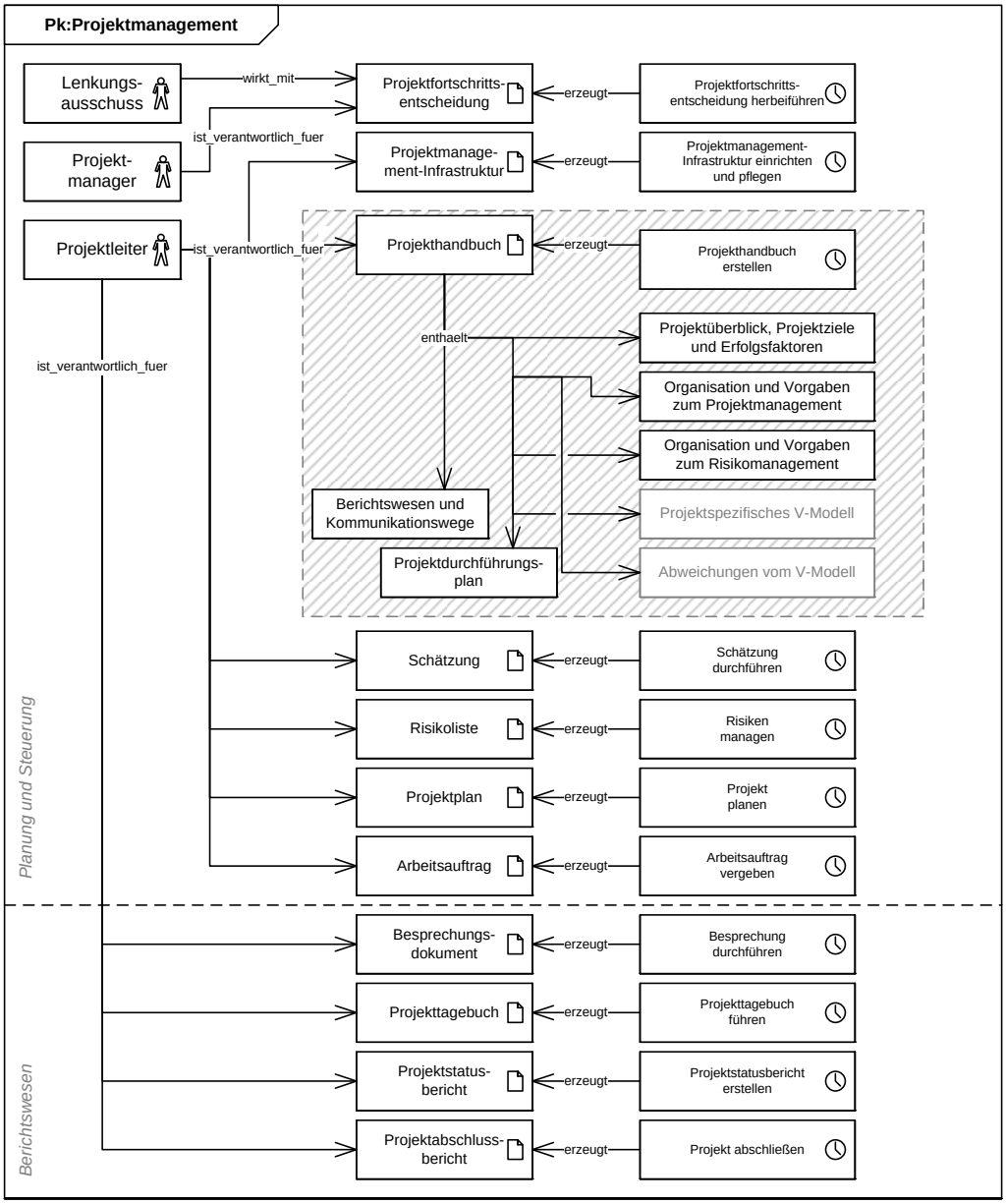

Abbildung 5.17.: Vorgehensbaustein Projektmanagement des V-Modell XT als Prozesskomponente modelliert

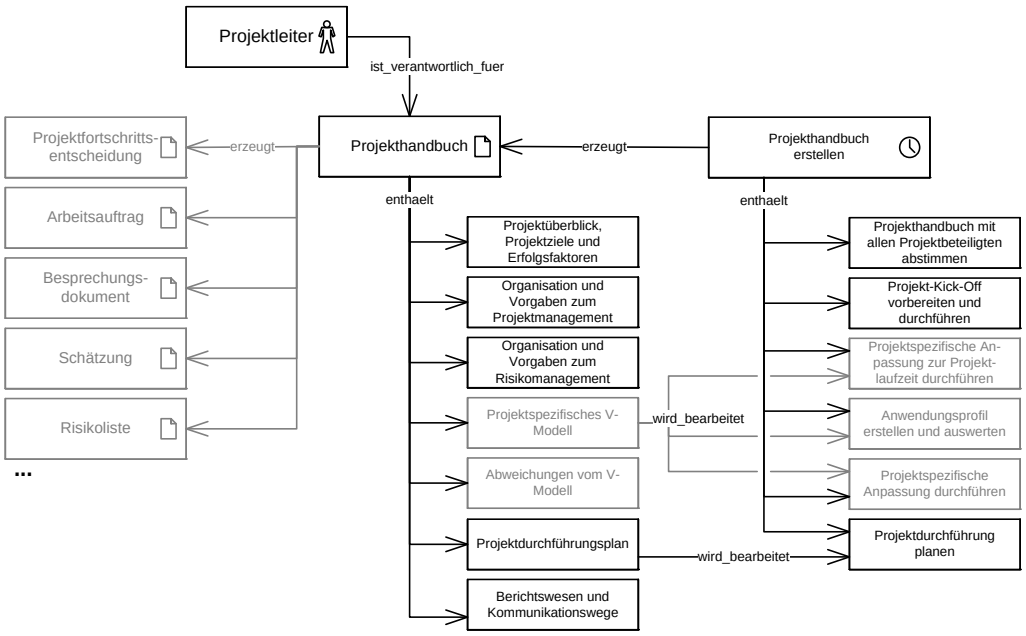

Abbildung 5.18.: Ausschnitt der Prozesskomponente Projektmanagement mit Detaillierung für das Projekthandbuch

Integration einer Prozesskomponente in ein Vorgehensmodell

[[vervollständigen]]

5.2. Modellierungssicht: Lebenszyklusmodell

In diesem Abschnitt setzen wir auf dem erarbeiteten Architekturmodell auf und definieren ein *Lebenszyklusmodell* über der Architektur. Wir orientieren uns dabei an den Entwicklungs- und Anpassungsprozessen, wie wir sie in Kapitel 2.2 analysiert haben. Die grundlegende Gestaltung des Modells führen ebenfalls auf den Solution Delivery Process (SDP) und die (Software-)Produktlinienentwicklung (PLE) aus Kapitel 2.3 zurück. Abbildung 5.21 zeigt das *Lebenszyklusmodell für Vorgehensmodelle und Vorgehensmodelllinien*. Es berücksichtigt die

- Entwicklung,
- Pflege und Bereitstellung sowie die
- Anwendung

von Vorgehensmodellen nach dem hierarchisch strukturierten Muster aus Kapitel 4. Dabei steht hier nicht mehr nur ein einzelnes Vorgehensmodell im Fokus, sondern eine Menge von Vorgehensmodellen. Dem PLE-Referenzmodell folgend, betrachten wir die vier Bereiche:

- Linienentwicklung,
- Management,
- Exemplarentwicklung und
- Anwendung.

Das Modell ist bewusst einfach gehalten und besteht im wesentlichen nur aus den fünf gezeigten Prozessen. Diese sind *komposit* und können verschiedenartig verfeinert werden. Wir geben im Kontext dieser Arbeit eine mögliche Verfeinerung an. Gleichzeitig beschreiben wir auch die Schnittstellen zwischen den einzelnen Prozessen, um die verschiedenartige Detaillierung zu unterstützen und ein hohes Maß an Flexibilität zu erhalten.

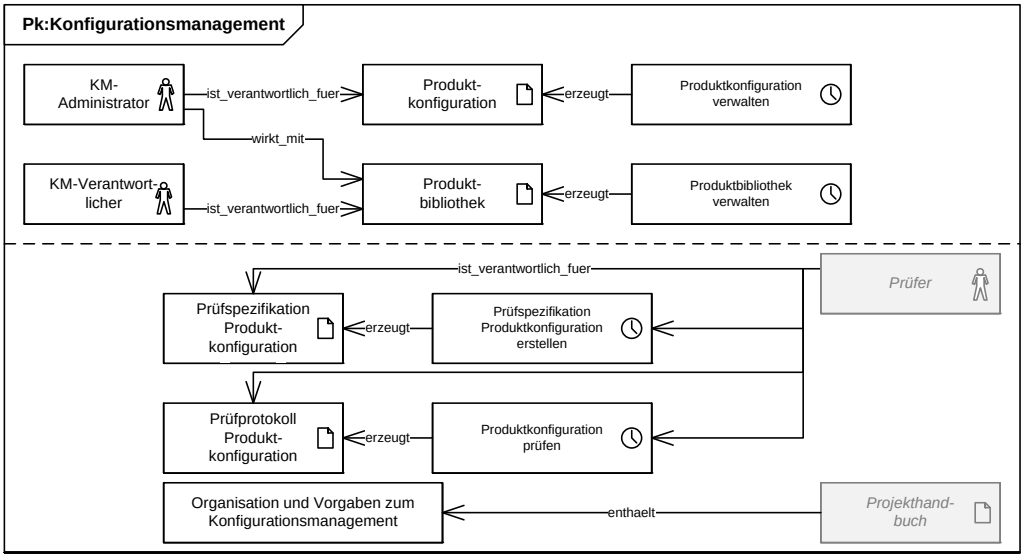

Abbildung 5.19.: Vorgehensbaustein Konfigurationsmanagement des V-Modell XT als Prozesskomponente modelliert

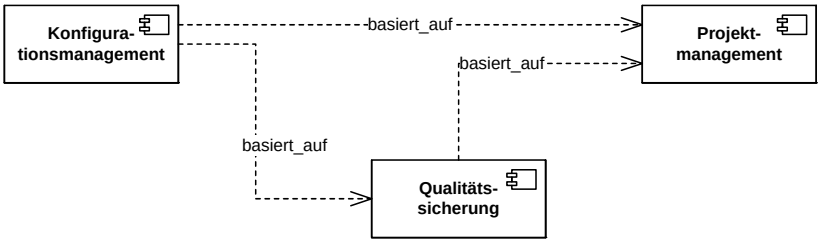

Abbildung 5.20.: Prozesskomponenten mit modellierten Abhängigkeiten

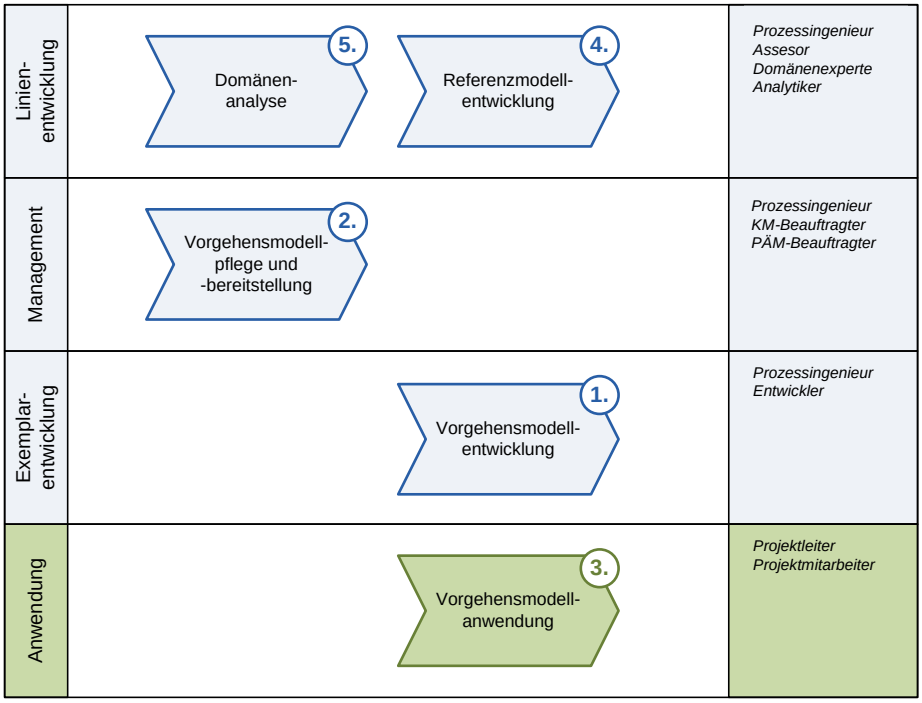

Abbildung 5.21.: Lebenszyklusmodell für Vorgehensmodelle und -Linien als Menge von Prozessen nach dem Schema der (Software-)Produktlinienentwicklung (PLE)

Wir erfassen mit dem Lebenszyklusmodell drei wesentliche Bereiche, die wir zu Böckle et al. [BKPS04] (vgl. Abbildung 2.13) positionieren (Tabelle 5.3) und im Anschluss kurz beschreiben.

PLE	Lebenszyklusmodell	Bemerkung
Domain Engineering	Linienentwicklung	–
Application Engineering	Exemplarentwicklung	–
Repository Management	Management	Umfasst die Artefakte und die Prozesse zu deren Verwaltung.
–	Anwendung	Enthält die Anwendung des Vorgehensmodells zur Etablierung einer Feedbackschleife.

Tabelle 5.3.: Positionierung des Lebenszyklusmodells zum PLE-Referenzprozesses

Bereits in Kapitel 2.3.2 wird auf die Notwendigkeit verschiedener Prozesse für den Kontext der *Linienentwicklung* hingewiesen. Wir definieren dies in unserem Ansatz mithilfe eines Entwicklungsprozesses für die Exemplarentwicklung, den wir in einen PLE-orientierten Ansatz integrieren. Als Schnittstelle hierfür dient uns ein expliziter *Management- und Bereitstellungsprozess*.

5.2.1. Verwaltung als zentraler Prozess

Im Zentrum des Lebenszyklusmodells in Abbildung 5.21 steht der Managementprozess (*2. Vorgehensmodellpflege und -bereitstellung*). Dieser ist die Anlaufstelle für die Neu- und Weiterentwicklung sowie die Anwendung von Vorgehensmodellen. Der zentrale Managementprozess arbeitet auf dem Metamodell, das wir im Rahmen dieser Arbeit vorgestellt haben. Konkret verwaltet er:

- Vorgehensmodellkonfigurationen und
- Prozesskomponenten sowie
- alle sonstigen benötigten Elemente

als Entitäten und stellt ebenfalls die assoziierten Operationen auf den Entitäten bereit (zum Beispiel Konstruktions-/Konfigurations- oder Exportfunktionen). Der Managementprozess (*2. Vorgehensmodellpflege und -bereitstellung*, Abbildung 5.21) bekommt als Eingaben nicht nur ausschließlich Informationen aus *1. Vorgehensmodellentwicklung*, sondern auch aus den Prozessen der Linienentwicklung (*4.* und *5.*). Der Managementprozess stellt eine Menge möglicher Konfigurationen bereit, von denen der Hauptstrang eine ist.

5.2.2. Prozessschnittstellen

In Abbildung 5.21 haben wir bislang die notwendigen Prozesse gesammelt und im PLE-Referenzmodell positioniert. Die benannten Prozesse sind jedoch untereinander auch gekoppelt. Wir betrachten nun die Schnittstellen zwischen diesen Prozessen und gehen auch hier wieder vom zentralen Prozess *2. Vorgehensmodellpflege und -bereitstellung* aus. Abbildung 5.22 zeigt die Schnittstellen zwischen dem Management, der Exemplarentwicklung und der Anwendung von Vorgehensmodellen.

Die zentrale Rolle des Prozesses *2. Vorgehensmodellpflege und -bereitstellung* wird zum Beispiel beim Prozess *3. Vorgehensmodellanwendung* deutlich, der nicht direkt zur Ausführung gelangt, sondern Eingaben vom Prozess *2. Vorgehensmodellpflege und -bereitstellung* erfordert. Dies weist auf ein zentrales Konfigurationsmanagement hin, in dem Vorgehensmodelle beziehungsweise Teile davon verwaltet werden. Analog dazu verläuft die Entwicklung eines Vorgehensmodells. Der Prozess *1. Vorgehensmodellentwicklung* benötigt Eingaben aus dem Prozess *2. Vorgehensmodellpflege und -bereitstellung*, stellt ihm

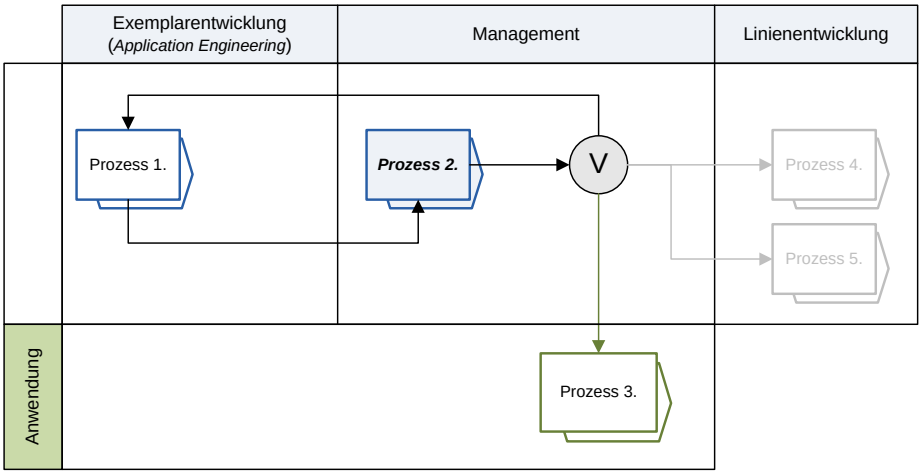

Abbildung 5.22.: Prozessschnittstellen zwischen Management, Exemplarentwicklung (Application Engineering) und Anwendung von Vorgehensmodellen

aber in jedem Fall seine Ergebnisse wieder als Eingabe zur Verfügung. Entwicklung und Weiterentwicklung[14] eines Vorgehensmodells finden also nur in den beiden Prozessen *1.* und *2.* statt.

Abbildung 5.23 zeigt die zweite Gruppe von Schnittstellen, die zu betrachten sind. Auch hier fungiert der Prozess *2. Vorgehensmodellpflege und -bereitstellung* als Anlauf- und Verteilerpunkt für die Prozesse *4. Referenzmodellentwicklung* und *5. Domänenanalyse*.

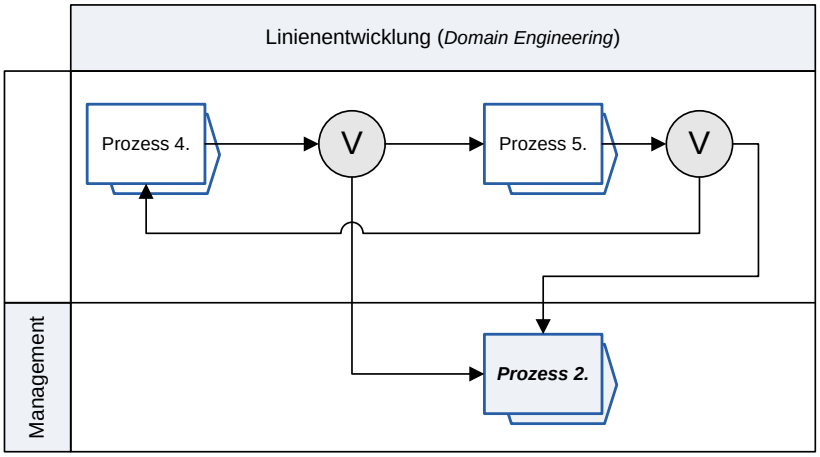

Abbildung 5.23.: Prozessschnittstelle zwischen Management und Linienentwicklung (Domain Engineering) von Vorgehensmodellen

Beide Prozesse erzeugen Eingaben für *2. Vorgehensmodellpflege und -bereitstellung* und herhalten über ihn selbst wieder Eingaben. Wir modellieren dadurch bereits auf dieser Ebene *Feedbackschleifen*, die kontinuierliche Informationsflüsse zwischen den einzelnen Prozessen des Lebenszyklusmodells ermöglichen.

Beispiel: Wird die Anwendung eines Vorgehensmodells abgeschlossen (*3. Vorgehensmodellanwendung*), d. h. wird ein Projekt abgeschlossen, stehen Erfahrungen mit dem eingesetzten Vorgehensmodell zur Verfügung. Diese werden in die Weiterentwicklung und Pflege zurückführt. Erfahrungen können dabei zum Beispiel auch aktualisierte Dokumentvorlagen oder aus Pragmatismus heraus optimierte Teilprozesse sein.

14 Dies betrifft auch eventuelle Evaluierungen zum Beispiel mithilfe von Pilotprojekten, die wir im Prozess *1. Vorgehensmodellentwicklung* positionieren.

Feedback ist im Kontext einer Prozesslinie umständlicher zu handhaben, als bei nur einem einzelnen Vorgehensmodell. Das Feedback muss der gesamten Plattform zugeführt werden und wirkt sich somit auf alle aus der Plattform instanziierten Modelle aus. Hier ist dann zu unterscheiden, welchen Einfluss das Feedback hat. Mögliche Einflussgrößen können wir wie folgt klassifizieren:

- *Domänenbezogenes Feedback* – Hier sind grundlegende Änderungen an der Plattform zu erwarten. Die Konsequenzen sind hier am weit reichensten.
- *Strukturbezogenes Feedback* – Hier sind Änderungen an Metamodellelementen also Strukturanpassungen zu erwarten.
- *Inhaltliches Feedback* – Hier sind Änderungen an konkreten Prozesskomponenten zu erwarten, beispielsweise durch Bugfixing oder sonstige Aktualisierungen.

Durch die so formulierten Feedback-Mechanismen sehen wir prinzipiell eine *proaktive Evolution* ([BKPS04], Seite 177ff.) der Gesamtplattform vor.

5.2.3. Prozess 1. Vorgehensmodellentwicklung

Die Entwicklung eines Vorgehensmodells ist ein äußerst komplexe Aufgabe, die neben den technischen Fähigkeiten und Fertigkeiten noch weitere *Soft Skills* von den Prozessingenieuren verlangt. Aus diesem Grund gibt es vielfältige Verfahren, eine Vorgehensmodellentwicklung durchzuführen. Dennoch lässt sie sich in der Regel immer auf die in Abbildung 5.24 gezeigten Schritte abstrahieren (vgl. Kapitel 2.3.3).

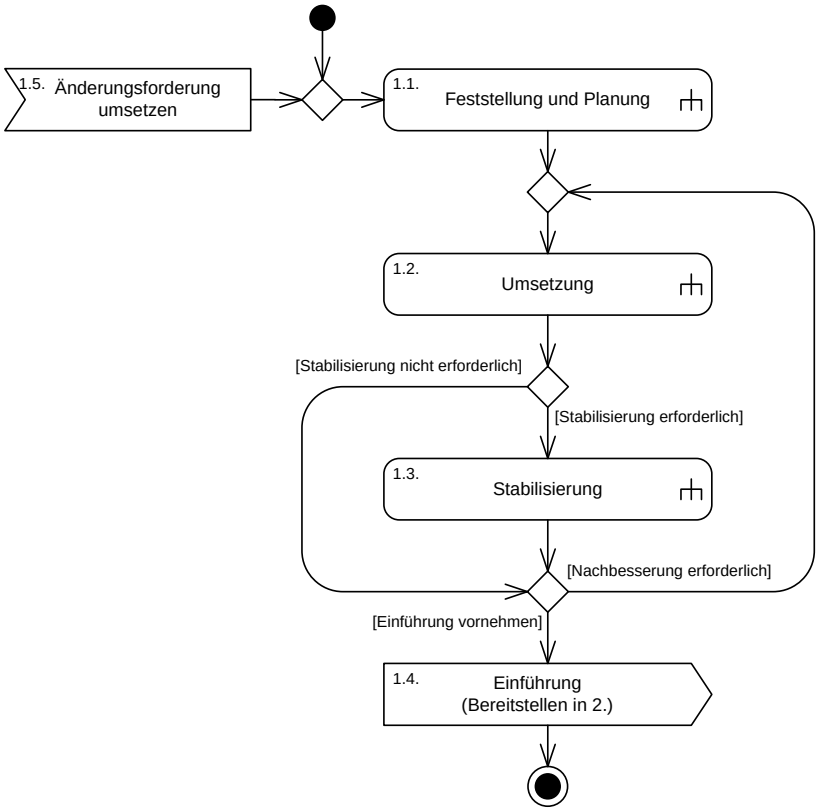

Abbildung 5.24.: Verfeinerung des Prozesses 1. Vorgehensmodellentwicklung

Der Prozess *1. Vorgehensmodellentwicklung* nimmt weiterhin einen besonderen Status ein, da er auch losgelöst vom Lebenszyklusmodell angewendet werden kann. Der Projekttyp *Einführung und Pflege eines organisationsspezifischen Vorgehensmodells* des V-Modell XT lässt sich zum Beispiel sehr gut durch diesen Prozess nachbilden.

Bevor wir die Teilprozesse verfeinern (Abschnitte 5.2.6 bis 5.2.8), betrachten wir die Schnittstelle des Prozesses (Tabelle 5.4).

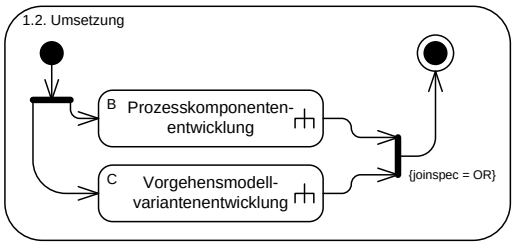

Abbildung 5.25.: Verfeinerung der Aktivität 1.2

Von	An	Beschreibung
2.		– Metamodell – Vorgehensmodellkonfigurationen (inkl. des Referenzmodells) – Prozesskomponenten – (sonst. durch das Metamodell beschriebene Elemente)
(extern)		– Konzeption für Vorgehensmodell – Anforderungen an Vorgehensmodellanpassung
	2.	– Vorgehensmodellkonfigurationen – Prozesskomponenten (neu, aktualisiert) – (sonst. durch das Metamodell beschriebene Elemente)

Tabelle 5.4.: Schnittstelle von Prozess 1. im Kontext Exemplarentwicklung und Linienentwicklung

5.2.4. Prozess 2. Vorgehensmodellpflege und -bereitstellung

Die Pflege und Bereitstellung von Vorgehensmodellen im Kontext einer Prozesslinie ist kompliziert, da hier neben verschiedenen Vorgehensmodellkonfigurationen auch noch verschiedene Versionen und ein Referenzmodell zu beachten sind. Abbildung 5.26 zeigt die Verfeinerung des entsprechenden Prozesses.

Kurzbeschreibung. Neu…

Beispiel: Um dies zu verdeutlichen, betrachten wir folgendes Beispiel. Aufgrund einer geänderten Anforderung in der Domäne (zum Beispiel ein neues Gesetz) ist es notwendig, eine der durch die Plattform bereit gestellten Vorgehensmodellkonfiguration anzupassen. In dieser Konfiguration werden einige Prozesskomponenten geändert, die gleichzeitig Teil des Referenzmodells sind. Die Anpassung der Konfiguration löst hier gleichzeitig eine Änderungsprozedur auf der Ebene des Referenzmodells aus, das die aktualisierten Prozesskomponenten einbindet. Eine Konsequenz ist mindestens eine Konsistenzprüfung des Referenzmodells.

Schnittstelle. Die Schnittstelle (Tabelle 5.5) dieses Prozesses ist sehr umfangreich. Sie muss hier anders als bei der Verwaltung von einzelnen Vorgehensmodellen nicht nur einen Pflegeprozess für ein Vorgehensmodell etablieren, sondern gleichzeitig mehrere Konfigurationen unterstützen und dabei Vorgaben einer übergeordneten Plattform berücksichtigen.

Die Schnittstelle zwischen diesem und dem Prozess *1. Vorgehensmodellentwicklung* besteht im Wesentlichen aus den bereits modellierten Größen (Abschnitt 5.1). Wie bereits zu Beginn eingeführt, fungiert dieser Prozess auch als zentraler Anlaufpunkt für die Bereitstellung eines organisationsspezifischen Vorgehensmodells. Dieses wird vom Prozess *3. Vorgehensmodellanwendung* als Eingabe benötigt und kann dort zum Beispiel analog zum V-Modell XT weiter angepasst werden (vgl. Kapitel 2.1.1 und 2.2.2). Nach Abschluss eines Projekts soll in unserem Modell das erworbene Wissen wieder zurück geführt werden. Hierzu verwenden ein *Feedback*-Konstrukt.

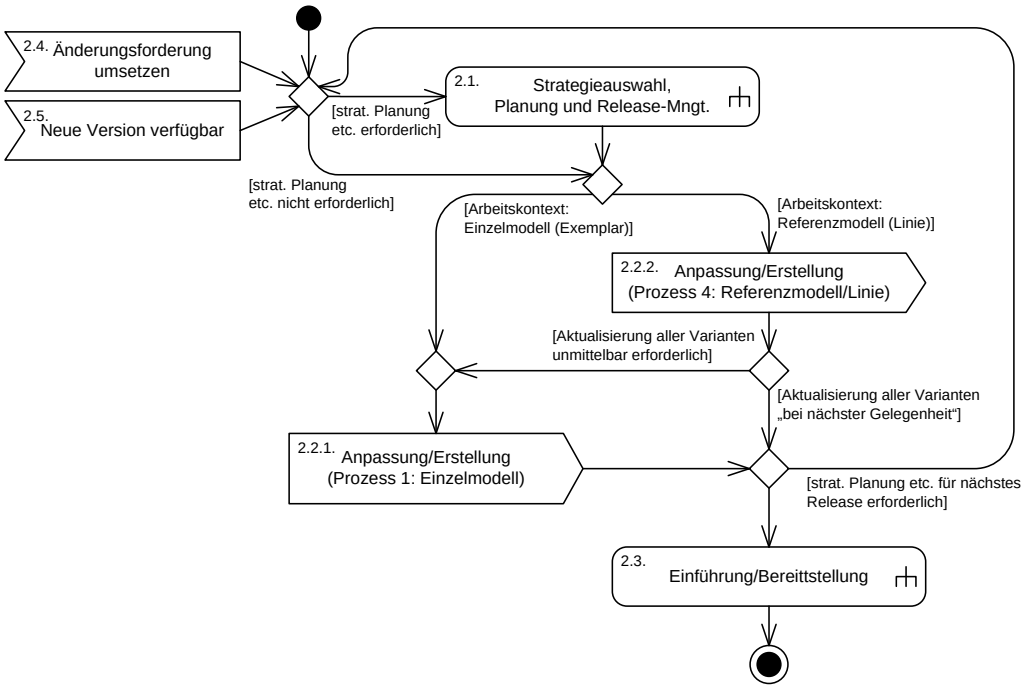

Abbildung 5.26.: Verfeinerung des Prozesses 2. Vorgehensmodellpflege und -bereitstellung

Von	An	Beschreibung
1.		– Vorgehensmodellkonfigurationen – Prozesskomponenten (neu, aktualisiert) – (sonst. durch das Metamodell beschriebene Elemente)
4.		– Metamodell – Referenzmodell (Konfiguration etc.) – Prozesskomponenten (des Referenzmodells und allg.)
6.		– Beschreibung der Domäne, Normen, Vorgaben etc. – Prozessbeschreibungen – (Dokument-)Vorlagen
	1.	– Metamodell – Vorgehensmodellkonfigurationen (inkl. des Referenzmodells) – Prozesskomponenten – (sonst. durch das Metamodell beschriebene Elemente)
	3.	– Vorgehensmodellkonfigruation

Tabelle 5.5.: Schnittstelle von Prozess 2. Vorgehensmodellpflege und -bereitstellung

Feedback im Lebenszyklusmodell. Da Feedback in verschiedener Form vorliegen kann, widmen wir uns dem im Folgenden intensiver. Feedback kann in verschiedenen Größenordnungen vorliegen und orientiert sich in der Regel immer an konkreten Anwendungsfällen. Mögliche Formen von Feedback sind:

- Problemmeldungen und Änderungsforderungen (Change Requests)
- Feature Requests
- Angepasste Dokumentvorlagen
- Neu erstellte oder angepasste Prozessbeschreibungen
- Beispielunterlagen
- Erfahrungsberichte
- Projektspezifische Vorgehensmodelle
- Neue Prozesskomponenten
- ...

Der Prozess *2. Vorgehensmodellpflege und -bereitstellung* muss diese Eingaben sinnvoll verarbeiten. Für neue Elemente muss eine entsprechende Einpflegung vorgenommen werden.

5.2.5. Prozess 4. Referenzmodellentwicklung

Die Entwicklung eines *Referenzmodells* is einer der zentralen Bestandteile einer Prozesslinie. Das Referenzmodell definiert gleichzeitig die Basisstruktur und die (minimal) gemeinsamen Inhalte aller Exemplare, die auf der Grundlage der Prozesslinie erstellt werden. Abbildung 5.27 zeigt den Prozess für die Modellierung eines Referenzmodells.

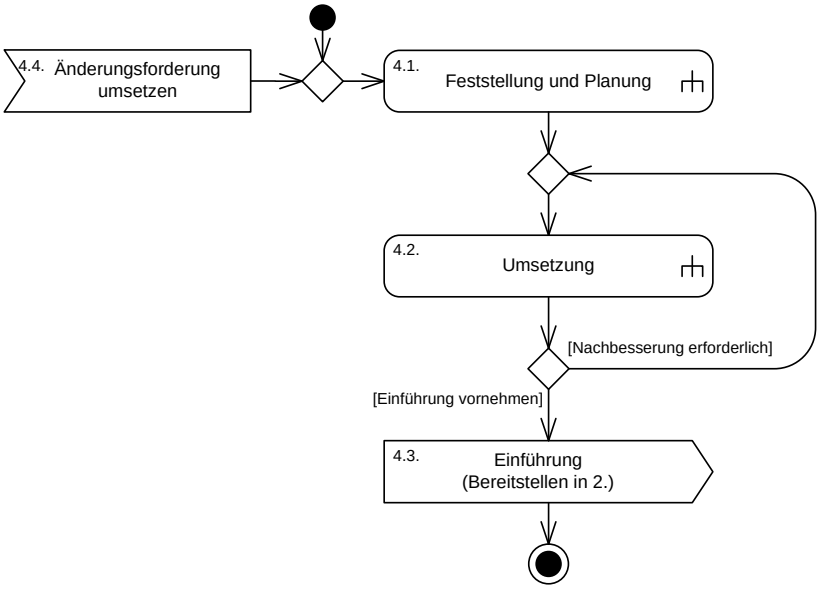

Abbildung 5.27.: Verfeinerung des Prozesses 4. Referenzmodellentwicklung

Kurzbeschreibung. Neu...

Schnittstelle. Die Schnittstelle dieses Prozesses ist in Tabellen 5.6 zu finden. Als Ausgaben liefert dieser Prozess das Metamodell und das Referenzmodell sowie dessen Bestandteile. Als Eingaben erhält dieser Prozess diejenigen Prozesskomponenten von Prozess *4.*, die Bestandteil des Referenzmodells sind. Weiterhin bekommt dieser Prozess alle Informationen aus der Domäne, die zur Erstellung und (inhaltichen) Pflege des Referenzmodells erforderlich sind.

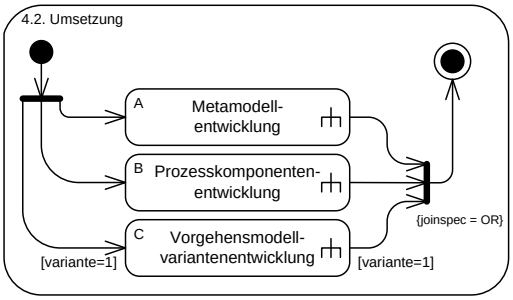

Abbildung 5.28.: Verfeinerung der Aktivität 4.2

Von	An	Beschreibung
2.		– Prozesskomponenten
5.		– Beschreibung der Domäne, Normen, Vorgaben etc. – Prozessbeschreibungen – (Dokument-)Vorlagen
	2.	– Metamodell – Referenzmodell (Konfiguration etc.) – Prozesskomponenten (des Referenzmodells)
	5.	– Metamodell

Tabelle 5.6.: Schnittstelle von Prozess 4. im Kontext Linienentwicklung

5.2.6. Subprozess A. Metamodellentwicklung

5.2.7. Subprozess B. Prozesskomponentenentwicklung

Der letzte für uns relevante Teilprozess ist der in Abbildung 5.30 gezeigte Prozess für die Entwicklung von Prozesskomponenten. Die Entwicklung von Prozesskomponenten findet hier anders als im Prozess *1. Vorgehensmodellentwicklung* nicht im Kontext eines konkreten Vorgehensmodells statt, sondern übergeordnet auf der Ebene der Plattform. Prozesskomponenten, die hier entwickelt werden, beziehen sich also in der Regel auf verallgemeinerbare Problemstellungen, die ein Wiederverwendung ermöglichen.

Kurzbeschreibung. Die Entwicklung von Prozesskomponenten ist sowohl fachlich als auch technisch sehr eng an das hier vorgestellte Metamodell (Abschnitt 5.1.2) gekoppelt. So werden nach einem Spezifikationsprozess die ermittelten Prozesselemente, die Dokumentation und die (internen) Abhängigkeiten festgelegt. Im Anschluss werden die benötigten Ressourcen allokiert (dies können zum Beispiel Beispieldokumente oder aber andere Prozesskomponenten sein, die referenziert werden müssen). Die fertige Prozesskomponente wird über die Plattform bereit gestellt und zur Nachnutzung angeboten.

Schnittstelle. Die Schnittstelle dieses Prozesses (Tebelle 5.7) ist sehr schmal gehalten. Er benötigt als Eingaben lediglich das Metamodell und die Vorgaben aus der Domäne, um die Prozesskomponenten umzusetzen. Sollen Prozesskomponenten geändert oder aktualisiert werden, so erwartet dieser Prozess eine entsprechende Änderungsspezifikation. Als Ausgaben produziert diese Prozess ausschließlich Prozesskomponenten, die einerseits der Referenzmodellentwicklung zur Verfügung stehen, andererseits auch direkt der Plattform zur Verfügung gestellt werden.

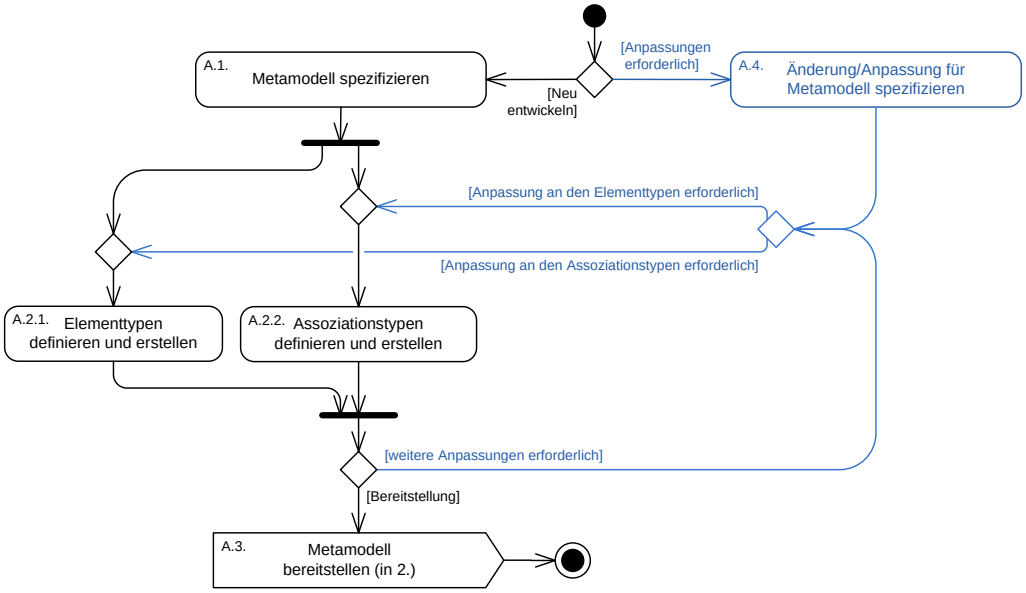

Abbildung 5.29.: Verfeinerung des Subprozess A. Metamodellentwicklung

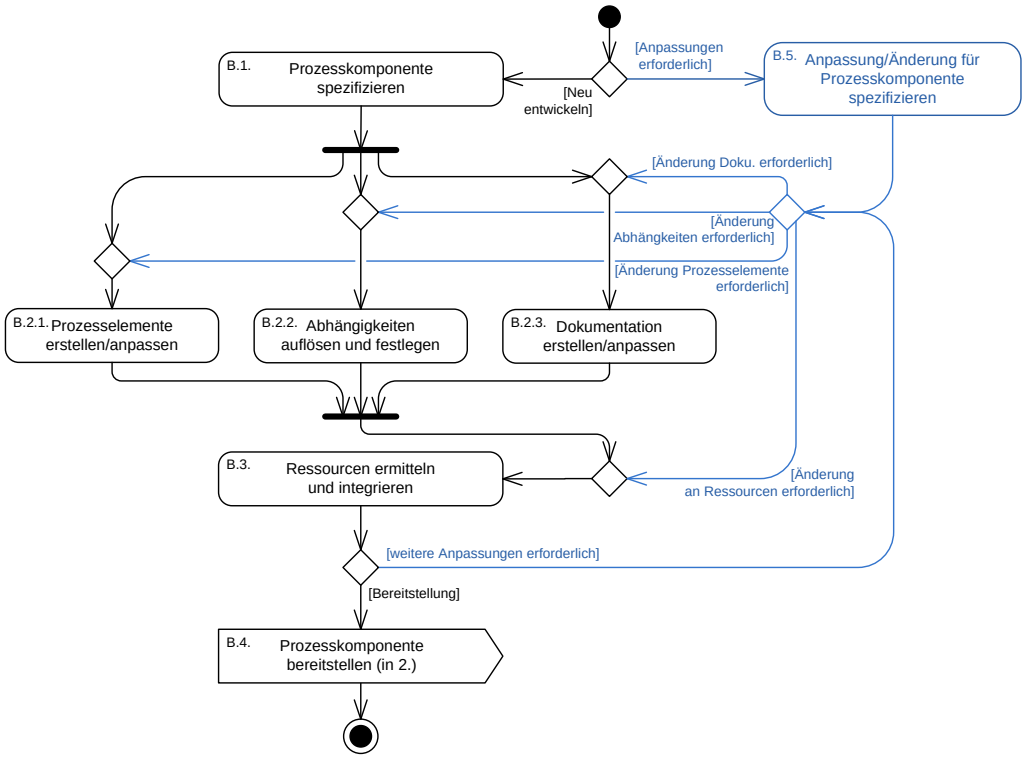

Abbildung 5.30.: Verfeinerung des Subprozesses B. Prozesskomponentenentwicklung

Von	An	Beschreibung
4.		– Metamodell – (Referenzinhalte)
6.		– Beschreibung der Domäne, Normen, Vorgaben etc. – Prozessbeschreibungen – (Dokument-)Vorlagen
	2.	– Prozesskomponenten
	4.	– Prozesskomponenten

Tabelle 5.7.: Schnittstelle von Prozess 5. im Kontext Linienentwicklung

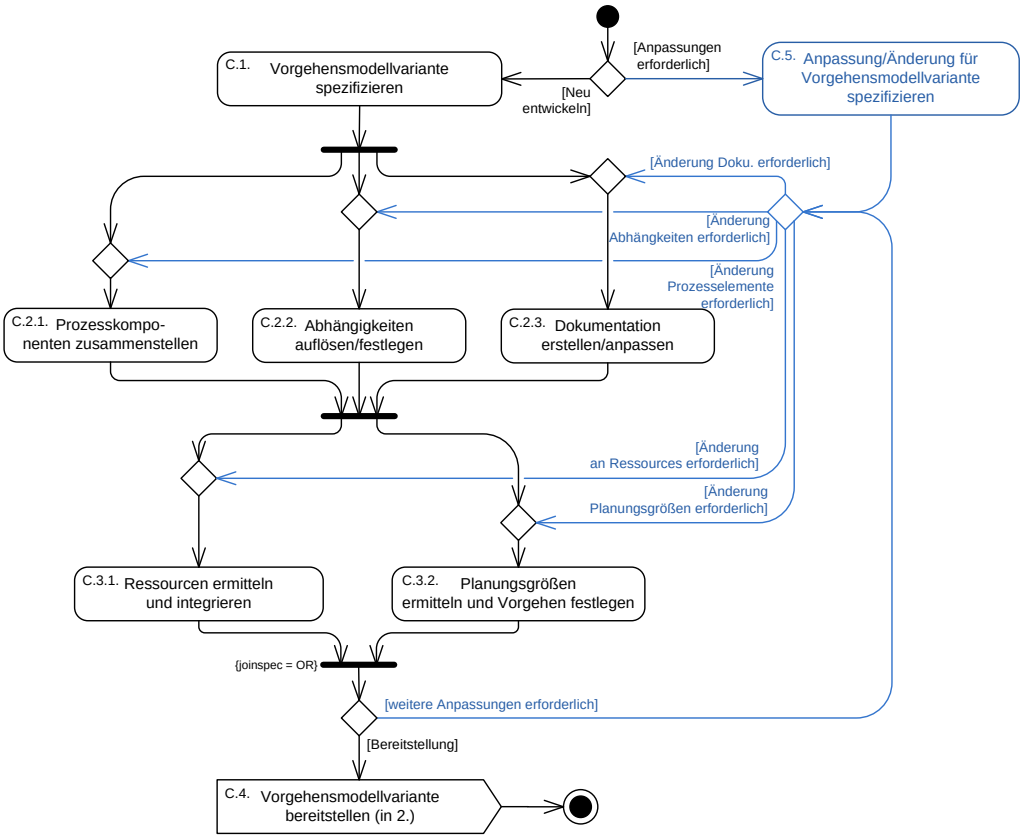

Abbildung 5.31.: Verfeinerung des Subprozess C. Vorgehensmodellvariantenentwicklung

5.2.8. Subprozess C. Vorgehensmodellvariantenentwicklung

5.2.9. Ein Zustandsmodell für Vorgehensmodelle

Betrachten wir Vorgehensmodelle im Rahmen eines Lebenszyklusmodells, müssen wir mögliche Zustände eines Vorgehensmodells berücksichtigen. Jedes Mal, wenn eine Aktivität der gerade vorgestellten Modelle ausgeführt wird, ist in der Regel ein Vorgehensmodell, beziehungsweise eine Instanz davon entweder Eingabe- oder Ausgabeparameter.

[[Zustandsmodell/Automat noch einführen]]

5.3. Werkzeugkonzept und -unterstützung

Sowohl das Metamodell als auch das Lebenszyklusmodell weisen trotz des erklärten Ziels der maximalen Einfachheit bereits ein hohes Maß an Komplexität und Integration auf. Eine Unterstützung durch Werkzeuge sehen wir nicht nur als erwünscht, sondern auch als erforderlich an. In diesem Abschnitt konzipieren wir ein Werkzeugkonzept, das insbesondere die Umsetzung des Metamodells und des Lebenszyklusmodells berücksichtigt. Im Anhang C stellen wir einen entsprechenden Prototyp kurz vor.

[[Werkzeugkonzept vorstellen, insbesondere die Abschnitte 5.1.5 und 5.3 berücksichtigen]]

– Strukturdefinition (Schemahierarchie in Verzeichnisstruktur)

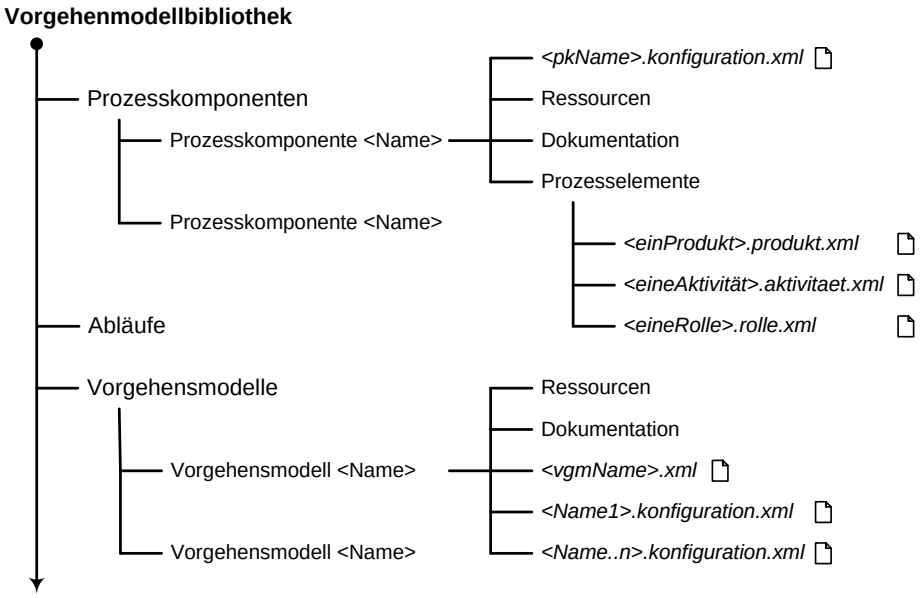

Abbildung 5.32.: Verteilungsarchitektur und Verzeichnishierarchie des Modells

Zusammenfassung

[[Ergebnisse zusammenfassen. . .]]