Grundlagen eines Formalisierungsframeworks für Vorgehensmodelle

Edward Fischer
Institut für Software- und Multimediatechnik
Technische Universität Dresden
eadfischer@web.de

Abstract: Zur Definition von Vorgehensmodellen wird häufig die natürliche Sprache verwendet. Diese ist ungenau, wodurch Interpretationsspielräume entstehen. Erschwert wird dadurch die Überprüfbarkeit einer korrekten Anwendung eines Vorgehensmodells zum einen, die Entwicklung von Werkzeugen zur Steuerung des Projektablaufes zum anderen. Dieser Beitrag erörtert wie diesen Problemen mit einem Framework zur Formalisierung von Vorgehensmodellen begegnet werden kann.

1 Einleitung

Vorgehensmodelle geben zu einem Projektablauf Anworten auf die Fragen *wer* macht *was*, *wie*, *womit* und in welcher *Reihenfolge?* [Ben04]. Je nach Vorgehensmodell werden solche Fragen nicht im vollen Umfang generisch, sondern auch unter Verwendung der natürlichen Sprache, beantwortet. Diese ist jedoch nicht eindeutig, so dass sich oftmals viele Auslegungsmöglichkeiten ergeben [kbs]. Als Folge bleibt die Frage offen, ob ein Vorgehensmodell korrekt angewendet wird. Ist beispielsweise die Überprüfung der richtigen Phasenabfolge noch überschaubar (da meist generisch genug definiert), stellen Inhalte von Produkten, sowie deren Abhängigkeiten größere Herausforderungen dar. Neben dem analytischen Überprüfen ist auch die konstruktive Projektsteuerung durch Werkzeuge ein wichtiger Faktor für den effektiven Einsatz von Vorgehensmodellen. Dies gilt um so mehr, je komplexer die Vorgaben gestaltet sind. Für die Entwicklung passender Werkzeuge ist jedoch eine exakte Spezifikation unumgänglich. Ist diese nicht von vornherein Teil des Vorgehensmodells, müssen mehrdeutige Vorgaben durch Werkzeughersteller interpretiert werden. Für diese bietet es sich dabei an, die Auslegung derart zu gestalten, dass existierende Lösungen mit entsprechenden Modifikationen weiterverwendet werden können. Aktuell ist dieser Weg am Beispiel des V-Modell XT [vmo06] und den Werkzeugen Innovator [inn], in-Step [ins] und dem Rational Method Composer [rat] nachzuverfolgen. Obgleich diese Praxis Vorteile im Bezug auf gewohnte Arbeitsumgebungen haben mag, dürfte sie dennoch nicht immer im Sinne von Vorgehensmodell-Autoren liegen. Wegen notwendiger Kompromisse können Modifikationen bisheriger Werkzeuge zu einer verzerrten Wiedergabe der Vorgehensmodelle führen.

Ziel ist es, ein Formalisierungsframework zu konstruieren, mit dem verschiedene Vorgehensmodelle verlustfrei und präzise beschrieben werden können. Der Nutzen liegt in

der Möglichkeit, die korrekte Anwendung eines Vorgehensmodells zu überprüfen, sowie die Entwicklung passender Werkzeuge zu vereinfachen und auch qualitätiv zu erweitern. Insbesondere können Aktivitätsbeschreibungen, anstatt sie wie bisher nur allgemein formuliert zu verwenden, auf aktuelle Daten instanziiert werden. Beispielsweise kann die Anweisung *"Ist für eine Softwareeinheit laut QS-Handbuch eine eigenständige QS notwendig, so ist deren Bearbeitungszustand auf vorgelegt, sonst auf fertig gestellt zu setzen"*, je nach tatsächlichem Inhalt des QS-Handbuchs durch *"auf vorgelegt setzen"* oder durch *"auf fertig gestellt setzen"* instanziiert werden. Erforderlich ist hierfür die generische Erfassbarkeit von Produktinhalten.

2 Bestehende Ansätze formaler Beschreibungen

In [Gna05] wird eine Methodik entwickelt, mit der Vorgehensmodelle als Generalisierung von Projektplänen verstanden werden können. Orientiert wird sich dabei am V-Modell XT, wobei sowohl Rollen, Aktivitäten als auch Produkte und deren Abhängigkeiten Beachtung finden. Der Abstraktionsgrad des Beschreibungsapparates ist dennoch kleiner als der des V-Modells gewählt, bedingt durch den Focus der Arbeit. Zur Beschreibung werden Klassendiagramme und Prädikatenlogik eingesetzt.

In [NSSW99] wird ein Beschreibungsapparat, bestehend aus den drei Modellen Objektmodell (Produkte), Koordinationsmodell (Aktivitäten) und Organisationsmodell (Rollen), vorgeschlagen. Im Objektmodell wird die Produkthierarchie definiert. Je Produkt gibt es im Koordinationsmodell ein Statechart, der mögliche Aktiväten in Abhängigkeit des Produktzustandes angibt. Abhängigkeiten zwischen Produkten werden durch Ereignisse modelliert, die Zustandsänderung betroffener Produkte auslösen. Ausgeführt werden die Definitionen durch die Prozesssteuerungs-Komponente von Merlin die um die beiden Aspekte Konfigurationsmanagement und Transaktionsmanagement erweitert wird. Nicht modelliert werden die in Vorgehensmodellen typischen Phasen. ebenfalls nicht addressiert werden die konkreten Inhalte der Produkte. Folglich können Produktabhängigkeiten nicht direkt über den Produktinhalten modelliert werden, sondern sind nur indirekt durch Erweiterung des Produktzustandmodells beschreibbar. Als Modellierungssprache wird die an UML angelehnte Sprache ESCAPE+ benutzt.

In [Fis06] wird ein nur auf das V-Modell XT ausgerichteter Beschreibungsapparat entwickelt. Grundlage bildet dabei die Definition von Produkten über die Spezifikation ihrer Inhalte. Der Projektzustand wird einzig als Summe aller aktuellen Produktversionen beschrieben, da sämtliche Projektmetadaten (wie Bearbeitungszustände u.ä.) durch Inhalte von Produkten abgedeckt werden. Projektzustandsübergänge erfolgen durch das Hinzufügen neuer Produktversionen, wobei prädikatenlogische Regeln über die jeweilige Zulässigkeit entscheiden. Als Folge entfällt ein explizites Konfigurationsmanagement, da alle Zustandsübergänge durch ein Konfigurationsverwaltungs-Werkzeug exakt nachverfolgt werden können. Zur einfachen Anwendbarkeit werden aus den Regeln Arbeitsanweisungen generiert, die über ein Workflowsystem an die Nutzer verteilt werden.

3 Formalisierungsframework

Als wichtigstes Gütekriterium eines Formalisierungsframeworks sei die Fähigkeit festgelegt, nicht nur zukünftige Vorgehensmodelle, sondern auch bereits existierende, verlustfrei abbilden zu können. Motiviert ist dies durch den Wunsch, möglichst viele Vorgehensmodelle samt unternehmensspezfischen Anpassungen anzusprechen. Weiterhin ist die Einfachheit des Beschreibungsapparates zu berücksichtigen - gemessen am Abstraktionsgrad und Umfang der Spezifikationen. Dieses Kriterium kann maßgeblich die Akzeptanz in der Praxis beeinflussen.

Obwohl auf das V-Modell XT ausgerichtet, besitzt der Ansatz aus [Fis06] keine gravierenden Einschränkungen, und soll daher als Grundlage eines Formalisierungsframeworks dienen. Die Idee dabei ist, die V-Modell-spezifischen Anteile zu entfernen, und als variable Anwendung des Frameworks zu verstehen. Die vom Vorgehensmodell unabhängigen Bestandteile bilden somit genau das Formalisierungsframework, welches im Folgenden festgehalten wird.

3.1 Produkte

Die (Zwischen-)Ergebnisse eines Projektes seien als Produkte bezeichnet. Die Struktur von Produkten wird abstrakt[1] über Eigenschaften definiert. Eigenschaften stellen die Inhalte der Produkte dar. Jedes Produkt besitzt mindestens die Eigenschaften Id und Version. Erstere wird zur Identifizierung von Produkten verwendet, letztere zur Unterscheidung von Produktversionen, da Produkte im Laufe eines Projektes überarbeitet werden können. Zusätzlich sei eine lineare Versionierung vereinbart. So existiert für jedes Produkt höchstens eine aktuellste Version. Weitere Eigenschaften können je nach Vorgehensmodell vorgesehen werden.

In einem Projekt sei zu jedem Zeitpunkt nur die aktuellste Version eines jeden Produktes betrachtet. Die Menge der Versionen zu einem konkreten Zeitpunkt t kann somit als der Projektzustand für t angesehen werden (Abbildung 1). Insbesondere sind sämtliche Metainformationen zum Projekt durch Eigenschaften von Produkten auszudrücken.

Wird Inhalt oder Anzahl von Produkten verändert, erfolgt ein Zustandsübergang. Zulässige Änderungen seien auf die beiden Arten *Hinzufügen* und *Ersetzen* beschränkt: beim Hinzufügen wird ein neues Produkt p erstellt und zum bisherigen Projektzustand z_1 hinzugefügt. Als Voraussetzung kommt die ID von p nicht in z_1 vor. Andernfalls ist ein Ersetzen vorzunehmen, indem das Produkt p_1 (mit der gleichen ID wie p) aus dem bisherigen Projektzustand z_1 durch die neue Version p ersetzt wird. Das Hinzufügen / Ersetzen darf nicht beliebig erfolgen, sondern wird durch die Validierungsfunktion f kontrolliert, die einen Projektzustand und ein Produkt p als Parameter erhält, und auf einen Wahrheitswert abbildet. Dieser bestimmt ob die Zustandsänderung zulässig ist. Gegenstand der Validierung sind die Eigenschaften des betrachteten Produktes p sowie Eigenschaften der Produkte in z_1. Prädikatenlogik [Höl01] kann verwendet werden, um gültige Zustandsänderungen

[1] In Anlehnung an die Spezifikation von Prozesstypen durch Coinduktion in [Rei04]

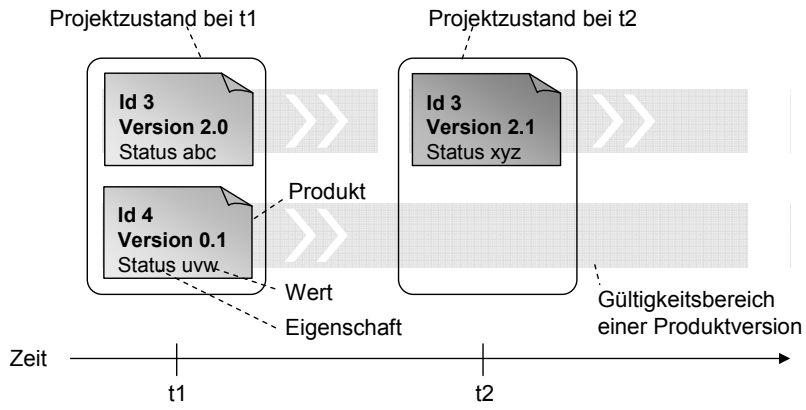


Abbildung 1: Projektzustände

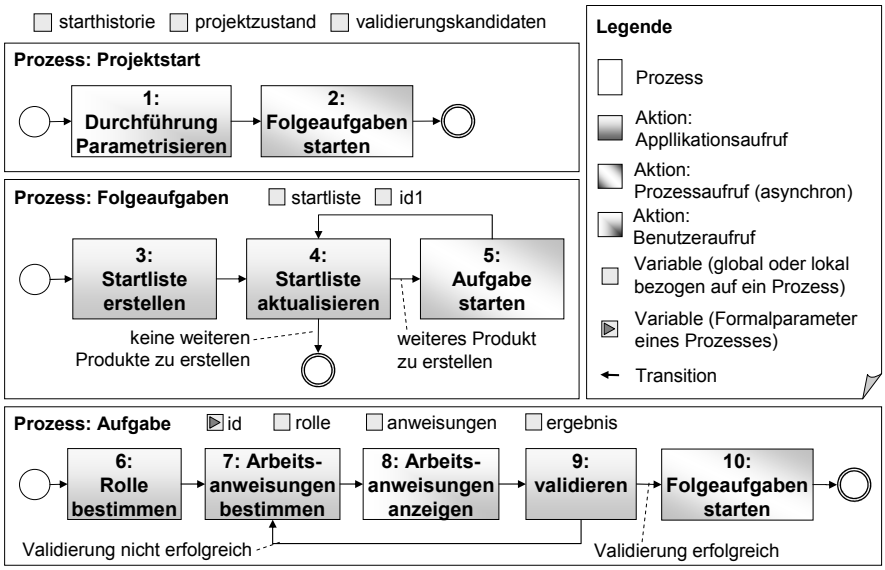
auszudrücken. Hierzu ist in der Menge der Variablen ein Symbol für den bisherigen Projektzustand z_1, die neue Produktversion p, sowie das gegebenenfalls zu ersetzende Produkt p_1, vorzusehen.

3.2 Projektdurchführung

Die Projektdurchführung kann als Folge von Projektzustandsübergängen verstanden werden. Ausgelöst werden diese durch Hinzufügen/Ersetzen von Produkten, wobei die Validierungsfunktion über die Zulässigkeit des Übergangs entscheidet. Um zu bestimmen, welche Produkte als nächstes zu erstellen sind, und mit welchen Werten deren Eigenschaften belegt sein dürfen, ist eine Analyse der Validierungsbedingungen im Bezug auf den aktuellen Projektzustand erforderlich. Dies ist dem Benutzer, in Anbetracht der Prädikatenlogik als Beschreibungsmittel, nicht zuzumuten. Statt dessen können Arbeitsanweisungen in natürlicher Sprache generiert werden. Dies führt das Framework jedoch nicht ad absurdum, da sich an der formalen Validierung nichts verändert.

3.3 Workflow

Die Gewinnung der Arbeitsanweisungen aus den Validierungsbedingungen sowie die Bestimmung der jeweils als nächstes zu erstellenden Produkte kann automatisiert in Form eines Workflows erfolgen. Ein Workflow besteht aus Prozessen die aus Aktionen und Transitionen zusammengesetzt sind, wobei Transitionen definieren, in welcher Reihenfolge Aufgaben abzuarbeiten sind. Variablen werden genutzt um Arbeitsergebnisse zu verwalten, oder um, im Zusammenspiel mit Transitionsbedingungen, Ablaufsteuerungen zu realisieren. Komplexe Berechnungen, die zudem keine Benutzerinteraktion erfordern, werden durch Applikationen aus dem Workflow ausgelagert. Diese Strukturierung eines



Projektstart
1: Benutzer setzt: projektzustand
2: Neuer Prozess: Folgeaufgaben

Folgeaufgaben
5: Neuer Prozess: Aufgabe(id1)
4: aktualisiereStartliste(*startliste*,*id1*)
3: erstelleStartliste(projektzustand,*startliste*,*starthistorie*)

Aufgabe
6: bestimmeRolle(id,*rolle*)
7: ladeAnweisungen(projektzustand,id,*anweisungen*)
8: Benutzer setzt: validierungskandidaten
9: validiere(*projektzustand*,*validierungskandidaten*,id,*ergebnis*)
10: Neuer Prozess: Folgeaufgaben

kursiv: durch Applikationsaufruf verändert

Abbildung 2: Frameworkprozesse

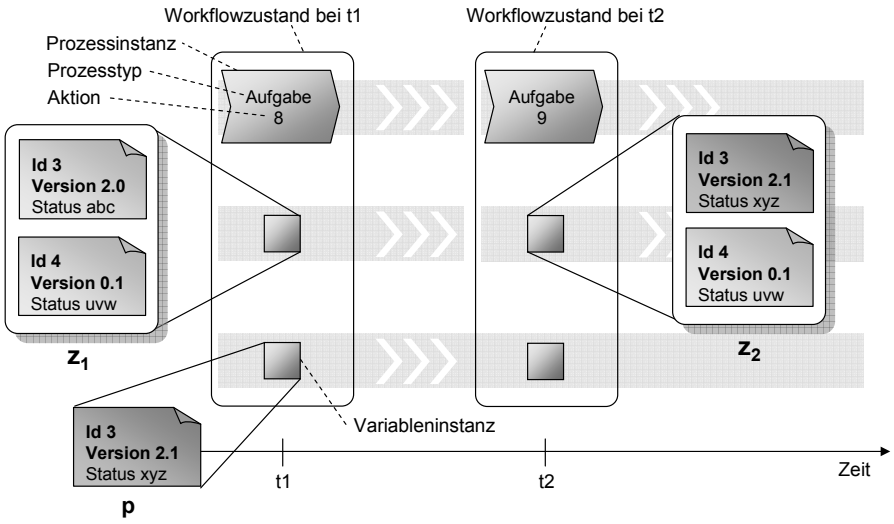


Abbildung 3: Einbettung Projektzustandsübergang in Workflowzustandsübergang

Workflows entspringt der XPDL [Coa02], und wird für die Prozesse des Frameworks (Abbildung 2) verwendet.

Begonnen wird ein Projekt mit dem Prozess *Projektstart*, bei dem im Workflow der Projektanfangszustand etabliert wird. Anschließend wird der Prozess *Folgeaufgaben* ausgeführt, der für jedes neu zu erstellende Produkt einen Prozess *Aufgabe* ins Leben ruft. Durch eine Starthistorie wird sichergestellt, dass dies pro Produktversion höchstens einmal geschieht (im Hinblick auf weitere Ausführungen des *Folgeaufgaben*-Prozesses). Kern eines *Aufgabe*-Prozesses ist die Bestimmung der zuständige Rolle, sowie die auszuführenden Arbeitsanweisungen. Ein zur Rolle zugeordneter Benutzer kann Anhand der Anweisungen ein neues Produkt erstellen, und dem Workflow übergeben. Ist die anschließende Validierung erfolgreich, wird der Prozess beendet und gegebenenfalls Folgeaufgaben ausgelöst. Andernfalls wird eine Fehlermeldung generiert und dem Benutzer zugestellt. Der Projektzustand sowie die vom Benutzer an den Workflow zur Validierung übergebenen Produkte, werden durch Variablen verwaltet.

Für ein konkretes Projekt werden Instanzen der definierten Prozesse und Variablen erzeugt. Im Laufe der Projektdurchführung werden sich diese verändern. Der Workflowzustand zu einem Zeitpunkt t kann folglich als eine Menge von Prozessinstanzen und Variableninstanzen verstanden werden. Workflowzustandsübergänge sind durch Regeln abgegrenzt, die vom Vorgehensmodell unabhängig sind.

Projektzustandsübergänge sind in Workflowzustandsübergänge eingebettet (Abbildung 3), und werden durch Änderung der entsprechenden Variableninstanz ausgelöst. Interaktion mit dem Benutzer erfolgt ebenfalls über das Verändern von Variableninstanzen, beispielsweise zum Setzen des zu validierenden Produkts.

3.4 Anwendung

Eine Spezifikation eines Vorgehensmodells ergibt sich durch eine Instanziierung des Formalisierungsframeworks, indem eine konkrete Ausgestaltung der Produktinhalte und Validierungsbedingungen vorgenommen wird. Die Anwendbarkeit dieses Frameworks zur Beschreibung des V-Modells ergibt sich unmittelbar aus dessen Ausgangspunkt. Interessant ist daher die Frage, ob und in welcher Weise andere Vorgehensmodelle beschrieben werden können. Betrachtet sei dazu skizzenhaft der Rational Unified Process [Cor98]. Obwohl dieser aktivitätsorientiert ist, kann er dennoch durch das produktorientierte Formalisierungsframework verlustfrei nachgebildet werden. Während Artefakte des RUP direkt durch Produkte abgebildet werden können [Gru05], müssen auch sämtliche Metadaten (gewählte Anzahl von Iterationen, konkrete Workflows u.ä.) ebenfalls in Produkten festgehalten werden. Insofern nicht alle Aspekte zu Artefakten passend zuordenbar sind, können diese in einem speziellen Metadaten-Produkt verwaltet werden. Validierungsbedingungen haben hierrauf Zugriff, wodurch ein Einfluss der Metadaten auf die Projektsteuerung gewährleistet ist.

4 Weiterführende Arbeit

Für eine breiten Einsatz des Frameworks ist dessen Güte anhand der Kriterien der Verlustfreiheit und Einfachheit zu optimieren. Insbesondere wirkt die verwendete Prädikatenlogik einem einfachen Framework entgegen, so dass graphische Beschreibungsmechanismen ergänzend eingesetzt werden könnten. Eine weitere Möglichkeit zum Senken des Beschreibungsumfanges ergibt sich durch Verlagerung allgemeiner Grundmechanismen von Vorgehensmodellen in das Framework selbst (vorgeschlagen in [NSSW99]).

Literatur

[Ben04] John Benad. Projektmanagement und Qualitätssicherung unter dem V-Modell XT. Diplomarbeit, Technische Universität Dresden, 2004.

[Coa02] Workflow Management Coalition. *Workflow Standard, Workflow Process Definition Interface – XML Process Definition Language*, 2002. Document Number WFMC-TC-1025.

[Cor98] Rational Software Corporation. *Rational Unified Process – Best Practices for Software Development Teams*. 1998.

[Fis06] Edward Fischer. Entwicklung und Erprobung einer formalen Darstellung des V-Modell XT für die Prozesssteuerung. Diplomarbeit, Technische Universität Dresden, 2006.

[Gna05] Michael Andreas Josef Gnatz. *Vom Vorgehensmodell zum Projektplan*. Dissertation, Technische Universität München, 2005.

[Gru05] Markus Grundmann. Prozessmodellintegration – Am Beispiel einer RUP-Erweiterung durch das V-Modell XT. Diplomarbeit, Hochschule Reutlingen, 2005.

[Höl01] Steffen Hölldobler. *Logik und Logikprogrammierung*. Kolleg Synchron, 2001.

[inn] MID, Innovator, *www.mid.de*.

[ins] microTOOL GmbH, in-Step, *www.microtool.de/instep*.

[kbs] Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung, Offizielles Forum zum V-Modell XT, *www.kbst.bund.de/-,279/V-Modell.htm*.

[NSSW99] Ulrich Nickel, Sabine Sachweh, Wilhelm Schäfer und Jörg Wadsack. Realisierung einer verteilten prozeßgesteuerten Arbeitsumgebung mit ESCAPE+/Merlin. In *Tagungsband zum Workshop des Forschungsverbundes NRW: Die Virtuelle Wissensfabrik.*, 1999.

[rat] IBM, Rational Method Composer, *www.ibm.com/software/rational*.

[Rei04] Horst Reichel. *Skript zur Lehrveranstaltung 'Formal Specification of Data and Process Types'*. 2004. Technische Universität Dresden.

[vmo06] *V-Modell XT, Release Version 1.2*, 2006.