Integration von Prozeßmodellierungsmethoden im Rahmen einer Prozeßzentrierten Entwurfsumgebung

Dirk Jesko, Martin Endig

Otto-von-Guericke-Universität Magdeburg, Institut für Technische und Betriebliche Informationssysteme, Postfach 4120, 39016 Magdeburg, Deutschland, Tel.: +49(391)67–11420, Fax.: +49(391)67–11216, Email: {jesko|endig}@iti.cs.uni-magdeburg.de

Zusammenfassung. Die Entwicklung komplexer Produkte erfordert die Zusammenarbeit von Entwicklern verschiedener Fachbereiche. Hinzu kommt, daß die Entwicklung meist nicht mehr nur in einem einzelnen Unternehmen, sondern in einem Verbund von Unternehmen durchgeführt wird. Daher kommen sowohl bei der Entwicklung der Produktmodelle, als auch bei der Spezifikation der Entwicklungsprozesse verschiedene Methoden und Werkzeuge zum Einsatz. Ziel dieses Beitrages ist die Darstellung eines allgemeinen Ansatzes für eine Integration der verschiedenen Entwicklungsmethoden. Weiterhin wird eine prozeß-zentrierte Produktentwicklungsumgebung vorgestellt und gezeigt, wie verschiedene Prozeßmodellierungsmethoden unter Verwendung des vorgestellten Ansatzes zur Integration in dieser Umgebung genutzt werden können.

1 Motivation

Die Unternehmen reagieren durch einen immer besseren Einsatz von technischen und organisatorischen Hilfsmitteln auf die sich ständig ändernde wirtschaftliche Lage des Maschinen und Anlagenbaus. Eine Forderung im Bereich der Produktentwicklung ist die Verkürzung der Produktfertigungszeiten bei gleichbleibenden oder sinkenden Kosten bei gleichbleibender oder steigender Qualität. Diese zum Teil gegensätzlichen Forderungen lassen sich beispielsweise durch die Einführung von neuen Konzepten der Produktionsorganisation oder durch die Reduzierung der Fertigungstiefen innerhalb eines Unternehmens erreichen. Darüber hinaus ist der Einsatz von modernen informationstechnischen Methoden und Techniken in allen technologisch notwendigen Entwicklungsphasen unabdingbar. Dabei wird der auszuführende Prozeß im allgemeinen mit *Engineeringprozeß* bezeichnet. Dieser umfaßt alle technologischen Prozesse, die für die Entwicklung, Fertigung und Wartung eines Produktes notwendig sind und ist insbesondere durch Benutzerinteraktionen gekennzeichnet (Gausemeier/Hahn/Schneider 1996). In diesem Beitrag wird insbesondere auf die Prozesse der Produktentwicklung (Produktplanung, Konstruktion und Arbeitsplanung) eingegangen. Dabei werden verschiedene Methoden und Werkzeuge für den Aufbau, die Analyse und die Modifizierung des Produktstrukturmodells mit dem Ziel der Herstellung von Fertigungsunterlagen eingesetzt. Diese Arbeiten sind meist durch die Teamarbeit verschiedener Entwickler gekenn-

zeichnet. Einzelne Entwickler arbeiten dabei meist nur an einer bestimmten Aufgabe und nutzen spezialisierte Softwarewerkzeuge, z.B. CAD-Systeme für den geometrischen Entwurf.

An der Entwicklung eines Produktes ist meist eine Vielzahl von Spezialisten beteiligt, die auf Grund der Komplexität vieler Produkte nur an einem bestimmten Teil dieses Produktes arbeiten. Während der Entwicklung dieser Teile werden verschiedene Methoden genutzt, die jeweils einen Teil der Entwicklung unterstützen bzw. mit denen jeweils ein bestimmter Aspekt des Produktes spezifiziert werden kann. Diese Vielfalt der Methoden und Notationen steht aber im Widerspruch zu der geforderten integrierten Darstellung der Produktinformationen über den gesamten Produktlebenszyklus. Daher wurde in (Grabowski, Geiger 1997) u.a. die Weiterentwicklung und Integration von Produktentwicklungsmethoden als Maßnahme für eine Verbesserung der Produktentwicklung genannt. In vielen Fällen ist es aber so, daß die Entwickler bzw. Unternehmen die bereits verwendeten Methoden auch weiterhin verwenden möchten, da eine Umstellung meist mit einem nicht unerheblichen Schulungsaufwand verbunden ist und außerdem viele Methoden bereits durch Softwarewerkzeuge unterstützt werden, deren Ersatz oft erhebliche Kosten verursacht. In diesem Beitrag soll daher ein Ansatz vorgestellt werden, der zum einen die Autonomie der verschiedenen Entwicklungsmethoden bewahrt. Andererseits aber eine Integration der mit den verschiedenen Methoden entwickelten Informationen ermöglicht.

Eine Möglichkeit die Produktentwicklungsprozesse innerhalb der Unternehmen besser unterstützen zu können, ist die Einführung *erweiterter* Technologien, die aufbauend auf den vorhandenen Softwarewerkzeugen innerhalb eines Unternehmens erweiterte Funktionalitäten anbieten. Spezielle *Entwurfsumgebungen* oder *CAx-Umgebungen* können dieses leisten, indem sie eine Menge von Werkzeugen bereitstellen, die eine einheitliche Bearbeitung aller während der Entwicklung anfallenden Entwurfsdokumente ermöglichen. Entwurfsumgebungen stellen eine Arbeitsumgebung dar, die Daten und Entwurfsdokumente verwalten, den Austausch von Informationen zwischen verschiedenen Werkzeugen oder auch zwischen Entwicklern ermöglichen, sich wiederholende Arbeitsschritte automatisieren und den gesamten Produktentwicklungsprozeß steuern (Sattler 1998). An dieser Stelle sei betont, daß die eigentliche Entwurfsumgebung nur eine Menge von *Basisfunktionalitäten* bereitstellt, die es ermöglicht, die Funktionalitäten der einzubeziehenden Softwarewerkzeuge innerhalb der Gesamtumgebung flexibler einzusetzen.

Für den Entwurf einer solchen integrierten, unternehmensweiten Entwurfsumgebung können generell drei Ansätze unterschieden werden, die unterschiedliche Aspekte in den Mittelpunkt der Betrachtung stellen. Zur Ausführung des Produktentwicklungsprozesses ist eine Menge von *Werkzeugen* notwendig, um eine Vielzahl von Produktinformationen zu erzeugen. Die Betonung des *Werkzeugaspektes* zur Umsetzung einer Entwurfumgebung ist der klassische Ansatz der Werkzeugintegration (Schefström/van den Broek 1992). Auch die Betonung des *Produktaspektes* zur Umsetzung einer Entwurfsumgebung ist bereits bekannt und wird in der Regel ausgehend von einem integrierten *Produktmodell* (Grabowski/Anderl/Polly 1993) in den EDM-Technologien verwendet. Alternativ dazu steht im Rahmen der Produktentwicklung die Betonung des *Prozeßaspektes*. Dieser wichtige Aspekt steht bisher in noch keinem unterstützenden Softwaresystem in den Mittelpunkt der Betrachtungen. In Anlehnung an die Konzepte aus dem Bereich *der Process-centered Software Engineering Environments* (Garg/Jazayeri 1995) und (Pohl 1995) ist aber eine entsprechende Unterstützung möglich.

Der in diesem Beitrag vorgeschlagene Ansatz *einer Process-centered Product Engineering Environment* zur unternehmensweiten Unterstützung des Produktentwicklungsprozesses ver-

sucht die Vorzüge einer prozeßzentrierten Betrachtungsweise mit einer entsprechenden Berücksichtigung des Werkzeugs- und Produktaspektes innerhalb einer Entwurfsumgebung zu kombinieren. Damit kann auf die Forderungen der Unternehmen nach besseren Technologien zur Unterstützung der Produktentwicklung eingegangen werden. Dieser Ansatz basiert generell auf dem in (Paul/Endig/Sattler 1998) vorgeschlagenen Integrationsrahmen für Ingenieursysteme, der von einer Integration auf vier unterschiedlichen Ebenen ausgeht (Wasserman 1990). Im Abschnitt 2 wird zunächst die Architektur der Umgebung beschrieben. Insbesondere soll dabei auf die Komponenten zur Spezifikation und Ausführung von Prozessen eingegangen werden. Anschließend wird im Abschnitt 3 ein auf UML basierender Ansatz für die Integration verschiedener während der Produktentwicklung eingesetzter Methoden vorgestellt. Im Abschnitt 4 soll dann kurz erläutert werden, wie dieser Ansatz auf die in der PPEE verwendeten Prozeßmodellierungsmethoden angewendet werden kann. Abschließend werden die Ergebnisse und offenen Probleme zusammengefaßt (Abschnitt 5).

2 Prozeßzentrierte Entwurfsumgebung

Die explizite Betrachtung der auszuführenden Prozesse innerhalb integrierter Entwurfsumgebungen führt zu einer neuen Herangehensweise der Unterstützung der Entwicklung von komplexen Produkten. Dabei wird ein Prozeß vor seiner eigentlichen Ausführung zunächst mit Hilfe entsprechender Methoden und Techniken der Umgebung spezifiziert. Der spezifizierte Prozeß kann dann innerhalb der Entwurfsumgebung beispielsweise durch einen Benutzer instantiiert und ausgeführt werden. Im Bereich der Produktentwicklung beinhaltet die Spezifikation eines Prozesses u.a. die Zuordnung von *Rollen*, einer Menge von *Eingabe-*, sowie einer Menge von *Ausgabedokumenten*. Diese Betrachtungsweise führt zu *einer prozeßzentrierten Produktentwicklungsumgebung* (Process-centered Product Engineering Environment - PPEE) (Endig 1999).

Bevor kurz auf die wesentlichen Komponenten der Umgebung eingegangen wird, soll zunächst der Begriff der Komponente definiert werden. Aufgrund der Heterogenität der betrachteten Softwarewerkzeuge wird im Rahmen der zu realisierenden Entwurfsumgebung von den konkreten Systemen hinsichtlich der Einführung eines Komponentenbegriffs abstrahiert. Dabei kann in Anlehnung an (Allen 1998) eine Komponente wie folgt definiert werden:

> *Eine Komponente ist ein wiederverwendbarer Software-Baustein, der eine Black-Box darstellt, die die interne Realisierung der komponentenspezifischen Funktionalitäten verbirgt und auf die nur über bekannte (standardisierbare) Schnittstellen zugegriffen werden kann. Darüber hinaus muß eine Komponente Schnittstellen besitzen, um eine Integration mit anderen Komponenten zu ermöglichen.*

Für die Realisierung dieser prozeßzentrierten Entwurfsumgebung wird ein komonentenbasierter Ansatz genutzt, um die Einbindung verschiedener Werkzeuge in die Umgebung zu ermöglichen. So kann beispielsweise das erfolgreiche Abschließen eines Projektes die auftragsgebundene Integration von speziellen Werkzeugen in die Umgebung oder die Verwendung von speziellen auftragsgebundenen Prozessen erfordern. Jede Komponente dieser Umgebung stellt eine Reihe von komponentenspezifischen Funktionalitäten bereit. Dabei wird von jeder Komponente eine Menge von Funktionalitäten zur Lösung eines speziellen Aufgabenbereiches, zum Beispiel eine Komponente zur thermische Simulation von Gußstücken, angeboten. Einige Komponenten werden unabhängig vom spezifischen Einsatzfall benötigt und stellen somit das Basissystem der Umgebung bereit (vgl. Bild 1). Zu diesen Komponenten gehören u.a.:

- *Prozeß Management Komponente*: Diese Komponente stellt alle Funktionalitäten zur Modellierung und Ausführung von Prozessen bereit. Dazu gehören auch Funktionalitäten zur Verifikation und Validierung der Prozesse. Diese Komponente steht im Rahmen einer prozeßzentrierten Entwurfsumgebung im Mittelpunkt der Betrachtungen.
- *Zustands Management Komponente*: Im Rahmen des Engineeringbereichs wird in der Regel der Lebenszyklus von Entwurfsdokumenten mit Hilfe von Zuständen und Zustandsübergängen beschrieben. Darüber hinaus werden auch bezüglich der Prozeßausführung entsprechende Zustände verwendet. Diese Komponente stellt die dafür notwendigen Funktionalitäten zur Modellierung und Ausführung von Zustandsdiagrammen bereit.
- *Zugriffs Management Komponente*: Die von dieser Komponente bereitgestellten Funktionalitäten werden verwendet, um einen gesicherten Zugriff auf Entwurfsdokumente und Prozesse zu gewährleisten.
- *Dokumenten Management Komponente*: Zur einheitlichen Verwaltung aller Entwurfsdokumente innerhalb der Umgebung stellt diese Komponente entsprechende Funktionalitäten bereit. Dazu gehören u.a. Funktionen zum Verwalten, zum Recherchieren und zum Archivieren von Dokumenten.
- *Produktstruktur Management Komponente*: Für die Verwaltung von Produkten ist es erforderlich nicht nur die Produkte selbst, sondern auch ihre Struktur zu verwalten. Dazu werden von dieser Komponente die Funktionalitäten bereitgestellt.

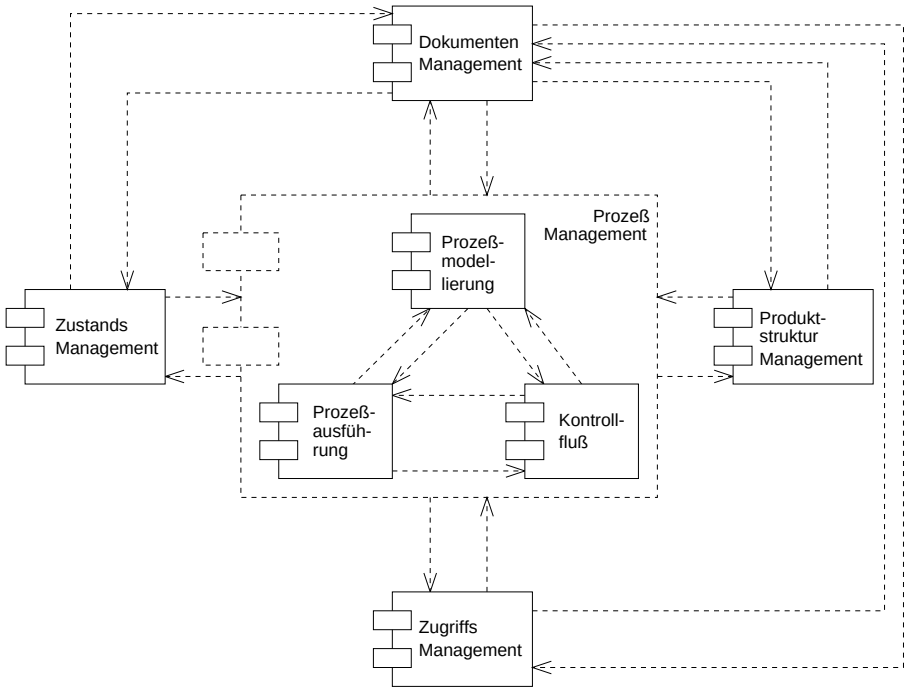

Bild 1: Komponentenbeziehungen der prozeß-zentrierten Entwurfumgebung

Diese in der Regel voneinander unabhängigen Komponenten werden im Rahmen der Entwurfsumgebung zusammengefaßt. Dazu ist es notwendig, die entsprechenden Schnittstellen der einzelnen Komponenten zu spezifizieren und mit Hilfe eines entsprechenden Modells zu verknüpfen. Ein solches Modell wird beispielsweise in (Sattler 1998) vorgestellt. So ist beispielsweise eine Verknüpfung der Zugriffs Management Komponente mit der Prozeß Mana-

gement Komponenten und der Dokumenten Management Komponente erforderlich. Die Realisierung der Prozeß Management Komponente erfolgt wiederum in drei unabhängigen Teilkomponenten:

- *Prozeßmodellierungskomponente*: Mit Hilfe der von dieser Komponente bereitgestellten Funktionalitäten ist es möglich, Prozeßmodelle zu definieren. Dabei sollen bekannte Modellierungsmethoden zum Einsatz kommen.
- *Prozeßausführungskomponente*: Die Aufgabe dieser Komponente ist die spezifische Ausführung der atomaren Prozesse. Dazu sind Funktionalitäten zum Instantiieren, Ausführen und Monitoring von Prozessen notwendig.
- *Kontrollflußkomponente*: Die eigentliche Logik des Prozesses wird mit Hilfe des Kontrollflusses beschrieben. Die dafür notwendigen Funktionalitäten werden von dieser Komponente bereitgestellt.

Diese Unterteilung soll wiederum gewährleisten, daß die Komponente flexibel an spezifische Anwendungsumgebungen bzw. -fälle angepaßt werden kann. Unter anderem soll durch die Einführung der separaten Prozeßmodellierungskomponente die Nutzung unterschiedlicher Prozeßmodellierungsmethoden, wie *FunSoft-Netze* (Gruhn 1991) *oder Ereignisgesteuerte Prozeßketten* (Staud 1999) ermöglicht werden. Andererseits kann die Ausführung der spezifizierten Prozesse unabhängig von der jeweiligen Modellierungsmethode erfolgen, z.B. auf einer anderen Hardwareplattform.

Der Einsatz unterschiedlicher Modelle innerhalb der verschiedenen Komponenten setzt einheitliche Schnittstellen zwischen diesen Komponenten voraus. Nur so ist eine „Zusammenarbeit“ der Komponenten zu erreichen. In der Prozeß Management Komponente können beispielsweise in der Prozeßmodellierungskomponente FunSoft-Netze und in der Kontrollflußkomponente Petri-Netze eingesetzt werden. Durch die Einführung eines *Generischen Prozeßmodells* (Endig/Jesko/Paul 1999) als komponentenübergreifendes Anwendungsmodell ist es möglich, diese drei Teilkomponenten so zusammenzufassen, daß die erforderlichen Funktionalitäten der Prozeß Management Komponente für die Entwurfsumgebung mit Hilfe der zugehörigen Komponentenschnittstellen bereitgestellt werden können.

Das generische Prozeßmodell soll dabei im wesentlichen als einheitliche Notation für die unterschiedlichen eingesetzten Modellierungsmethoden dienen. Der Ansatz geht davon aus, daß alle verwendeten Prozeßmodellierungsmethoden auf dieses generische Modell abgebildet werden können. Eine Anforderung an das generische Modell ist daher die Möglichkeit der eindeutigen Abbildung der Elemente der verschiedenen Prozeßmodelle auf Elemente des generischen Modells. Im folgenden Abschnitt soll ein allgemeiner Ansatz für eine Integration verschiedener Methoden vorgestellt und auf seine Eignung im Rahmen der Prozeßmodellierungskomponente hin untersucht werden.

3 Ansatz zur Methodenintegration

Der in diesem Abschnitt vorgestellte Ansatz zur Integration soll es zum einen ermöglichen, die durch die verschiedenen Methoden erzeugten Modelle in einem Modell zusammenzufassen. Andererseits soll aber auch die Autonomie der verschiedenen Methoden erhalten bleiben, wodurch es einem Entwickler ermöglicht wird, seine Entwicklungen in das Gesamtmodell zu integrieren, aber andererseits seine Methode wie gewohnt zu verwenden. Die Probleme, die durch die Nutzung vieler verschiedener Methoden während der Produktentwicklung auftreten,

sollen an dieser Stelle nicht weiter erörtert werden, da sie keine Auswirkungen auf den eigentlichen Ansatz haben.

Bevor der Ansatz zur Integration vorgestellt wird, soll kurz erläutert werden, wie der Begriff der *Methode* verwendet wird. In (Löhr-Richter 1993) wird eine Methode definiert als „Eine systematische Vorgehensweise zur Erreichung eines bestimmten Ziels“. Eine Methode umfaßt demnach nicht nur eine Sprache für die Darstellung eines bestimmten Sachverhaltes, sondern auch ein Vorgehensmodell. Die Sprache (oder Notation) beschreibt die Konstrukte die innerhalb der Methode verwendet werden können und die Regeln für deren Verknüpfung, d.h. die Syntax. Das Vorgehensmodell beschreibt, wie die Sprache in konkreten Situationen einzusetzen ist.

Eine Anforderung der in diesem Beitrag vorgestellten Methode zur Integration ist, daß die von den Entwicklern verwendeten Methoden in der gewohnten Weise erhalten bleiben, es aber trotzdem möglich sein soll, die mittels der verschiedenen Methoden erzeugten Informationen zu integrieren. Daher ist die Integration der Vorgehensweisen von untergeordneter Bedeutung, da sich diese in der Regel nur auf eine bestimmte Methode beziehen. Dieser Beitrag fokussiert daher auf die Integration der verschiedenen Notationen. Wenn nicht explizit das Vorgehensmodell einer Methode genannt wird, bezieht sich der Begriff der Methode im folgenden auf die Notation dieser Methode.

Der generelle Ansatz geht davon aus, daß eine Methode als Basis für die Integration dient. Diese Methode wird als *Basismethode* bezeichnet. Die spezialisierten von den Entwicklern eingesetzten Methoden werden dem gegenüber als *spezifische Methoden* bezeichnet. Die Basismethode umfaßt zum einen eine Notation für die Darstellung „aller“ Informationen mittels einer Notation und zum anderen Vorgehensmodelle für die Integration der Notationen der spezifischen Methoden. Bevor eine Integration erfolgen kann ist die Wahl einer geeigneten Basisnotation und der zugehörigen Vorgehensmodelle notwendig. Obwohl die Definition der Vorgehensmodelle entscheidend von der gewählten Basisnotation abhängt. wird in unserem Ansatz drei grundlegende Strategien der Integration unterscheiden, die vollständige, die strukturelle und die lose Integration. Bevor auf diese im einzelnen eingegangen wird, soll zunächst die Wahl einer geeigneten Basisnotation erfolgen.

Auf Grund der Vorteile und der Flexibilität objektorientierter Konzepte bietet sich deren Nutzung auch als Basisnotation im Bereich des vorgestellten Integrationsansatzes an. Ziel der Integration der spezifischen Methoden in die Basismethode ist zum einen die Darstellung aller Informationen in einer Notation. Zum anderen soll aber auch der Austausch von Modellen oder Modellteilen zwischen Entwicklern ermöglicht werden, die u.U. unterschiedliche Methoden verwenden. Dieser Austausch ist nur möglich, wenn gemeinsame Objekte in den verschiedenen Methoden existieren, die durch gemeinsame Objekte in der Basismethode bereitgestellt werden. Die Basismethode muß also entweder genügend Konzepte zur Verfügung stellen, um alle (notwendigen) Elemente der spezifischen Methoden darstellen zu können oder sie muß Konzepte bereitstellen, die eine Erweiterung der Basismethode zulassen.

Im folgenden soll kurz auf die drei zuvor genannten Integrationsstrategien eingegangen werden. Die Unterscheidung verschiedener Strategien wurde eingeführt, um die Möglichkeit zu haben, die verschiedenen spezifischen Methoden auf verschiedenen Abstraktionsebenen zu integrieren. Als Basisnotation wird die Unified Modeling Language (UML) (OMG 1999, Rumbaugh/Jacobson/Booch 1999) verwendet. Diese bietet sich aus verschiedenen Gründen als an, u.a. weil Konzepte für die Erweiterung der Notation zur Verfügung stehen, die insbesondere bei der strukturellen Integration benötigt werden. In der aktuellen Version 1.3 sind

diese Konzepte noch eingeschränkt. Für die Version 2.0 ist aber geplant die Konzepte für die Spracherweiterung auszubauen (Kobryn 1999).

Die zur Spracherweiterung in UML zur Verfügung stehenden Konzepte sollen an dieser Stelle kurz erläutert werden. Die Erweiterungsmöglichkeiten erlauben die Definition von *Profiles*, die Modellelemente für einen bestimmten Anwendungsbereich bereitstellen. Der UML-Standard (OMG 1999) definiert bereits zwei solcher Erweiterungen, u.a. das *UML Profile for business modeling*, in dem Elemente für die Modellierung von Business Prozessen bereitgestellt werden. Dies ist aber eher als Beispiel für die Erweiterungsmöglichkeiten von UML zu verstehen, denn für den praktischen Einsatz (Nüttgens/Feld/Zimmermann 1997). Daher soll auf diesen Ansatz nicht weiter eingegangen werden. Eines der Erweiterungskonzepte von UML ist der *Stereotype*, der als „eine Art virtuelle Metamodell Klasse die innerhalb des Modells hinzugefügt wird, ohne das vordefinierte Metamodell von UML zu verändern“ (Rumbaugh/Jacobson/Booch 1999) definiert ist. Ein Stereotype erlaubt die Definition eines neuen Modellelementes basierend auf einem in UML bereits definierten Modellelement. Damit ist es möglich die existierenden Modellelemente an neue Aufgabenbereiche anzupassen. Für das Profile for business modeling werden beispielsweise die Stereotypen „worker" und „entity“ definiert, die auf dem UML-Metamodell-Element Klasse basieren. In einem Modell, das diese Erweiterung nutzt, kann dann beispielsweise eine konkrete Klasse „Administrator“ mit dem Stereotype „worker“ versehen werden. Die Darstellung eines Stereotype erfolgt entweder durch Angabe dessen Namen in ‚<<‘ und ‚>>‘ über dem Namen des entsprechenden Elements oder durch ein eigenes grafisches Symbol. Ein Stereotype kann zusätzlich um *Constraints* (Bedingungen an das Element) erweitert werden. Diese gelten dann für alle Elemente, die diesen Stereotype besitzen. Für die Spezifikation dieser Constraints bietet UML seit der Version 1.3 *die Object Constraint Language* (OCL) (Warmer/Kleppe 1999) an.

Die Integration verschiedener Methoden, Darstellungsformen soll es u.a. ermöglichen, die unterschiedlichen Konzepte verschiedener Techniken zusammenzubringen und somit eine einheitliche Darstellung der Informationen aus unterschiedlichen Teilbereichen realisieren. Für eine derartige Integration wird oft so vorgegangen, daß die Konzepte der verschiedenen Methoden vereinigt werden, so daß sie als eine neue Methode verwendet werden können. Ein Beispiel dieser Strategie ist die in (Scheer/Nüttgens/Zimmermann 1997) und (Nüttgens/Feld/Zimmermann 1998) vorgestellte Integration von EPK und UML zu objektorientierten EPK (oEPK). Damit wird u.a. das Ziel verfolgt eine integrierte Beschreibung sowohl der Prozesse, als auch deren Ergebnisse zu erreichen. Ziel ist es in diesem Fall, Gemeinsamkeiten zwischen den verschiedenen Methoden zu finden und damit die jeweiligen Vorteile auszunutzen.

Das Vorgehen, bei dem die Methoden so integriert werden, daß eine neue wird im folgenden als vollständige Integration bezeichnet. Dies schließt sowohl den Fall ein, daß eine Integration vorgenommen wird, bei der eine neue Methode entsteht, wie im Beispiel der oEPK, als auch der Fall, daß eine Abbildung von einer der Notationen auf Elemente der anderen Notation vorgenommen wird. Im folgenden soll nur auf den Fall der Integration durch Abbildung genauer eingegangen werden, da dieser im Zusammenhang mit dem generischen Prozeßmodell angewendet wird. In unserem Ansatz wird zwischen einer Basismethode und spezifischen Methoden unterschieden. Daher ist zunächst eine Abbildung der Notation jeder spezifischen Methode in die Notation der Basismethode zu finden. Dadurch wird erreicht, daß die in einer spezifischen Methode modellierten Modelle in der Basismethode darstellbar sind. Werden Abbildungen verschiedener spezifischer Methoden erzeugt, können die Informationen der verschiedenen Modelle in einem einheitlichen Modell dargestellt werden.

Eine Forderung, die an den vorgestellten Ansatz gestellt wurde war die Beibehaltung der spezifischen Methoden, d.h. der Entwickler soll mit seiner gewohnten Methode weiterarbeiten können. Daher ist es nicht nur erforderlich, die Abbildung von der spezifischen Methode in die Basismethode vornehmen zu können, sondern auch den umgekehrten Weg, d.h. die Rückgewinnung der Informationen aus dem transformierten Modell. Im Bereich der Prozeßmodelle ist dies beispielsweise von Bedeutung, wenn die Modelle verändert oder nach deren Ausführung in abgewandelter Form wiederverwendet werden sollen. Da die Modelle nach der Integration vollständig in der Notation der Basismethode vorliegen, ist nicht mehr nachzuvollziehen, wie diese entstanden sind, d.h. mit welcher spezifischen Methode sie entwickelt wurden. Eine mögliche Lösung ist die Definition einer bijektiven Abbildung zwischen der Basismethode und den spezifischen Methoden. Eine solche Abbildung ist aber aufgrund der Unterschiede in der Ausdrucksmächtigkeit und der Abstraktion der Methoden oft nicht einfach realisierbar. Einige dieser Probleme werden in Abschnitt 4 am Beispiel des generischen Prozeßmodells kurz erläutert.

Im Gegensatz zur vollständigen Integration wird bei der strukturellen Integration nur die Struktur der spezifischen Methode in die Basismethode integriert, d.h. die Struktur der spezifischen Methoden bleibt erhalten, wird aber mit Elementen der Basismethode dargestellt. Dazu ist es erforderlich innerhalb der Basismethode neue Elemente erzeugen zu können, die für die Darstellung der Notation der spezifischen Methoden angewendet werden. Mit Hilfe der neuen Elemente können dann die Modelle aus der spezifischen Methode direkt in der Basismethode dargestellt werden. Da jedes dieser neuen Elemente direkt mit einem Element derspezifischen Methode in Beziehung steht, ist es in diesem Fall problemlos möglich das Modell in der Originalnotation wiederherzustellen. Andererseits ist es aber auch möglich in der Basismethode eine Beziehung zwischen den neuen Elementen und den bereits existierenden Elementen herzustellen. Durch die Beibehaltung der Struktur der Modelle und die Möglichkeit der Rückabbildung wird erreicht, daß die bestehenden Methoden weiter benutzbar sind. Da der Ansatz davon ausgeht, daß der Notation der Basismethode „neue“ Elemente hinzugefügt werden, ist es notwendig oder zumindest wünschenswert, daß die Basismethode Konzepte für derartige Erweiterungen zur Verfügung stellt. Ist dies nicht der Fall, erfordert die Integration einer spezifischen Methode u.U. eine grundlegende Änderung der Basismethode. Da innerhalb unseres Ansatzes die UML zum Einsatz kommt, stehen Mittel zur Erweiterung der Notation zur Verfügung, so daß eine entsprechende Ergänzung der Basismethode vorerst nicht notwendig ist.

Der Ansatz der strukturellen Integration geht davon aus, daß für jede zu integrierende spezifische Methode ein Profile erstellt wird. Für jedes Element der spezifischen Methode wird ein geeignetes Element aus dem UML Metamodell gewählt und ein auf diesem Element basierender Stereotype erzeugt. Alternativ kann auch einer aus einer bereits bestehenden Erweiterung gewählt werden, wenn dieser den selben Sachverhalt darstellt. Damit steht eine Grundlage für die Verbindung verschiedener spezifischer Methoden zur Verfügung. Beziehungen zwischen Stereotypen lassen sich in einem Klassendiagramm darstellen. Nachdem alle notwendigen Stereotypen erzeugt wurden, können bei Bedarf passende „Symbole“ (alternative grafische Darstellungen) für deren Darstellung festgelegt werden. Existieren in der spezifischen Methode beispielsweise bereits grafische Darstellungen für die einzelnen Elemente, dann bietet es sich an, diese zu verwenden. Nach der Definition der Stereotypen werden, wenn dies erforderlich ist, Constraints angegeben, mit denen eine Einschränkung der Anwendung der Elemente erfolgen kann. Bei Methoden, die eine auf Graphen basierende Notation besitzen, kann mit Constraints beispielsweise sichergestellt werden, daß nur bestimmte Kno-

tentypen miteinander verbunden werden dürfen, wie es beispielsweise bei Petri-Netzen oder EPK der Fall ist. Abschließend können noch „Well-Formedness Rules“ definiert werden, die angeben, wie die neuen Elemente miteinander verknüpft werden dürfen. Eine übliche Regel ist beispielsweise die Einschränkung, daß nur Elemente des selben Stereotype mittels Generalisierung verbunden werden dürfen.

Der Vollständigkeit halber soll abschließend noch kurz auf die lose Integration eingegangen werden. Diese bietet sich insbesondere dann an, wenn eine Aufschlüsselung der einzelnen Elemente einer Methode nicht erforderlich, nicht erwünscht oder möglicherweise nicht möglich ist. Ein solcher Fall ist beispielsweise bei einem Simulationsmodell denkbar. Existiert ein solches Modell, dann genügt es innerhalb des Gesamtmodells eine Art Referenz auf dieses zu setzen. Unter Verwendung von UML als Basisnotation bietet sich die Darstellung dieses Modells durch das UML-Element Notiz oder durch ein Element mit einem entsprechenden Stereotype an. Die Anwendung der Notiz ist von Vorteil, wenn das Modell mit einem beliebigen anderen Element verbunden werden soll, da dies mit einer Notiz problemlos möglich ist. Der Ansatz über den Stereotype bietet sich demgegenüber an, wenn die Möglichkeit einer späteren Erweiterung bestehen soll. An dieser Stelle soll aber nicht weiter auf diese Form der Integration eingegangen werden, da sie für die folgenden Ausführungen nicht verwendet werden soll.

Die in Abschnitt 2 vorgestellte Entwicklungsumgebung besitzt eine Komponente für die Prozeßmodellierung, die aber keine spezifische Prozeßmodellierungsmethode voraussetzt. Vielmehr soll die eigentliche Modellierung der Prozesse durch den Entwickler, mit einer beliebigen Prozeßmodellierungsmethode erfolgen. Im nächsten Abschnitt wird beschrieben, wie dies mit Hilfe des in diesem Abschnitt vorgestellten Ansatzes möglich ist und welche Probleme dabei auftreten.

4 Methodenintegration in der Prozeßmodellierungskomponente

In Abschnitt 2 wurde für die Prozeßmodellierungskomponente ein generisches Prozeßmodell vorgeschlagen, daß als Grundlage für die Nutzung beliebiger Prozeßmodellierungsmethoden innerhalb dieser Komponente dienen soll. Aus diesem generischen Modell werden dann die für die anderen Komponenten notwendigen Modelle, z.B. das für die Kontrollflußkomponente abgeleitet. Dem Ansatz aus Abschnitt 3 folgend dient das generische Modell als Basisnotation für die Integration und die spezifischen Prozeßmodellierungsmethoden werden in Form einer vollständigen Integration mit dieser integriert, wobei eine Abbildung der Notationen der spezifischen Methoden auf die Basisnotation zur Anwendung kommt. Der Vorteil dieses Ansatzes liegt vor allem in der einheitlichen Grundlage für Generierung der Modelle für die anderen Komponenten. Es treten aber die bereits erwähnten Probleme auf. Zum einen muß das generische Modell die gesamte Syntax und Semantik der zu integrierenden Methoden darstellen können, was insbesondere dann problematisch ist, wenn sich die verschiedenen Methoden wesentlich unterscheiden. Sollen beispielsweise FunSoft-Netze und Petri-Netze auf das generische Modell abgebildet werden, entstehen Probleme, da FunSoft-Netze abstraktere Elemente besitzen.

Ein weiteres Problem ergibt sich bei der Wahl der Elemente für das generische Prozeßmodell. Da innerhalb dieses Modells sowohl Elemente auf hoher Abstraktionsstufe, z.B. eine Funktion einer EPK, als auch Elemente auf niedriger Abstraktionsstrufe, z.B. eine Transition in einem Petri-Netz, abgebildet werden müssen ist es erforderlich, daß die Elemente des generischen Prozeßmodells diese Abstraktionsstufen berücksichtigt. Daraus ergibt sich folgendes

Problem: Werden die Elemente so allgemein gehalten, daß beispielsweise Petri-Netze direkt auf das generische Modell abgebildet werden können, so ist bei der Abbildung der FunSoft-Netze keine 1-zu-1-Abbildung der Elemete möglich, vielmehr ist es erforderlich, die einzelnen Elemente der FunSoft-Netze auf Strukturen des generischen Modells abzubilden. Diese Abbildung wäre ähnlich der bei der Definition der FunSoft-Netze verwendeten Abbildung auf Prädikaten-Transitions-Netze (Gruhn 1991). Werden die Elemente des generischen Modells andererseits so allgemein gewählt, daß die Elemente der FunSoft-Netze direkt abgebildet werden, ist die Integration einfacher Petri-Netze schwierig, da die höheren Konzepte innerhalb des generischen Modells oft nur durch Gruppen von Elementen in den Petri-Netzen dargestellt werden können. Modelliert ein Entwickler „seine“ Prozesse mit Petri-Netzen, dann müßte er demnach auch diese Gruppen von Elementen verwenden. Ist dies nicht der Fall, ist die Abbildung der Elemente auch nicht eindeutig möglich.

Daher scheint die Anwendung eines etwas anderer Ansatz für die Integration sinnvoll. Dieser basiert auf der Annahme, daß innerhalb des „vollständigen“ Prozeßmodells verschiedene Methoden für verschiedene abgeschlossene Teilbereiche angewendet werden. Beispielsweise die EPK für die Modellierung der übergreifenden Geschäftsprozesse und FunSoft-Netze zur Modellierung eines einzelnen Teilprozesses, der beispielsweise den Prozeß der Erstellung eines Stückes Software, das zu dem Produkt gehört, beschreibt. In diesem Fall bietet sich die strukturelle Integration an. Die einzelnen Methoden werden wie im vorherigen Abschnitt beschrieben unter Verwendung der Erweiterbarkeit von UML integriert. Damit stehen die verschiedenen Prozeßmodelle in einer einheitlichen Notation zur Verfügung und es können innerhalb dieser Notation Beziehungen zwischen den Modellen oder auch Elemente aufgebaut werden.

Durch die Anwendung der strukturellen Integration kann es dazu kommen, daß Elemente gleicher Semantik mit verschiedenen Elementen der Basismethode dargestellt werden (unterschiedliche Stereotypen). Dadurch ist keine direkte Ableitung der Modelle der anderen Komponenten (Prozeßausführung, Kontrollfluß) mehr möglich. Um dies zu ermöglichen müßten die für die jeweilige Komponente relevanten Daten während der Integration der Methoden gesondert gekennzeichnet werden. Inwieweit dies möglich ist bleibt noch zu untersuchen.

5 Zusammenfassung und Ausblick

Für die Modellierung der Prozesse innerhalb der Umgebung soll mit beliebigen Prozeßmodellierungsmethoden möglich sein. Daher wurde zunächst ein genereller Ansatz für die Integration verschiedener Notationen in eine Basisnotation vorgestellt, die eine einheitliche Darstellung der verschiedenen Notationen ermöglicht. Ziel dieser Darstellung ist, den anderen Komponenten eine einheitliche Schnittstelle für den Zugriff auf das gesamte Prozeßmodell zu ermöglichen. Über diese Schnittstelle können dann die Modelle der beiden anderen Komponenten abgeleitet werden, ohne das diese Kenntnis von den real verwendeten Modellierungsmethoden haben. Für das generische Modell wurde zunächst ein Ansatz vorgestellt, der von einer vollständigen Abbildung der spezifischen Methoden in das generische Modell ausgeht. Durch diese Abbildung wird das Gesamtmodell, das sich aus den verschiedenen Teilmodellen zusammensetzt, in einem einheitlichen Modell dargestellt. Dieses Modell beschreibt somit den gesamten Prozeß. Es hat sich aber gezeigt, daß sich der Ansatz als problematisch erweist, wenn sich die spezifischen Methoden in der Ausdrucksmächtigkeit bzw. in der Abstraktion der verfügbaren Elemente unterscheiden. Aus diesem Grund wurde ein Ansatz vorgestellt der die Integration der spezifischen Methoden nur auf struktureller Ebene vornimmt. Dieser An-

satz geht davon aus, daß eine bestimmte Modellierungsmethode auch nur für einen bestimmten abgeschlossenen Teil des Prozeßmodells verwendet wird. Die strukturelle Integration der verschiedenen Methoden in die Basismethode ermöglicht dann wiederum die Darstellung der Modelle in einer Notation und die Spezifikation von Zusammenhängen der verschiedenen spezifischen Modelle. Die Einbindung der Notationen der spezifischen Methoden in die Notation der Basismethode kann dabei teilautomatisch realisiert werden und ist somit einfacher als die Definition der Abbildungen auf das generische Modell.

Ungeklärt bleibt bei der Nutzung der strukturellen Integration allerdings noch die Generierung der Modelle für die anderen Komponenten aus dem Gesamtmodell. Damit werden sich weitere Forschungsarbeiten befassen. Weiterhin soll der Ansatz praktisch getestet werden, d.h. die in diesem Beitrag theoretisch beschriebenen Ansätze sollen auf die verschiedenen Prozeßmodellierungsmethoden angewendet werden. Dazu soll zunächst der allgemeine Integrationsansatz auf Basis von UML weiter untersucht werden. Insbesondere ist die Spezifizierung der Vorgehensweisen bei der Integration nach den drei Strategien notwendig.

Literatur

Allen, P.: Component-based Architecture: A Call for Pragmatism. In (Grundy 1998), pages 5-12.

Endig, M.; Jesko, D.; Paul, G.: Konzepte zur Prozeßintegration in Rechnerunterstützten Ingenieursystemen. PrePrint 17, Otto-von-Guericke-Universität Mageburg, Dezember 1998.

Endig, M.: Ansatz einer Komponenten-basierten Prozeßsteuerung zur Modellierung und Ausführung von Engineeringprozessen. In: *Turowski, K. (Hrsg.)*: Tagungsband zum 1. Workshop Komponentenorientierte betriebliche Anwendungssysteme" (WKBA 1), Seiten 65-71, Magdeburg, März 1999.

Grabowski, H.; Anderl, R.; Polly, A.: Integriertes Produktmodell. Entwicklungen zur Normung von CIM. DIN Deutsches Institut für Normung e.V., Beuth, Berlin, Wien, Zürich, 1. Auflage, 1993.

Genderka, M.: Objektorientierte Methode zur Entwicklung von Produktmodellen als Basis Integrierter Ingenieursysteme. Berichte aus der Konstruktionstechnik. Shaker Verlag, 1995. Dissertation, Universität Paderborn.

Grabowski, H,; Geiger, K. (Hrsg.): Neue Wege zur Produktentwicklung. RAABE, 1997.

Gausemeier, J.; Hahn, A.; Schneider, W.: Kooperatives Modellieren auf Basis transienter Objekte. In*: Ruland, D. (Hrsg)*: Verteilte und intelligente CAD-Systeme: Tagungsband; Kaiserslautern, 7. und 8. März 1996; Fachtagung der Gesellschaft für Informatik/CAD '96, Informatik Xpress 8, Seiten 311-325, Bonn, 1996. Detlev Ruland. Gesellschaft für Informatik e.V., Fachgruppe 4.2.1: Rechnergestütztes Entwerfen und Konstruieren (CAD).

Garg, P. K.; Jazayeri M.: (Hrsg.): Process-Centered Software Engineering Environments, Los Alamitos, California, Januar 1995. IEEE Computer Society Press.

Gruhn, V.: Validation and Verification of Software Process Models. Dissertation, Forschungsbericht 394/1991, Universität Dortmund, Fachbereich Informatik, 1991.

Grundy, J. (Hrsg): Proceedings of Workshop on Component-based Information Systems Engineering (CBISE98), Working Paper 98/12 (Department of Computer Science, University of Waikato Hamilton, New Zealand), Pisa, Italy, Juni 1998.

Hahn, A.: Integrationsumgebung für verteilte objektorientierte Ingenieursysteme. Dissertation, Heinz Nixdorf Institut, Universität-GH Paderborn, 1998. HNI-Verlagsschriftenreihe, Band 33; Rechnerintegrierte Produktion

Humpert, A.: Methodische Anforderungsverarbeitung auf Basis eines objektorientierten Anforderungsmodells. Dissertation, Heinz Nixdorf Institut, Universität-GH Paderborn, 1995. HNI-Verlagsschriftenreihe, Band 9; Rechnerintegrierte Produktion

Kobryn, C.: UML 2001: A Standardization Odyssey. Communications of the ACM, 42(10):29-37, October 1999.

Löhr-Richter, P.: Methodologie - Methodeik - Methode. Was steckt dahinter? EMISA FORUM Mitteilungen der GI-Fachgruppe "Entwicklungsmethoden für Informationssysteme und deren Anwendung", Heft 1:39-41, 1993.

Nüttgens, M.; Feld, T.; Zimmermann, V.: Business Process Modeling with EPC and UML. In *Schader, M.; Korthaus M. (editors)*: The Unified Modeling Language: Technical Aspects and Applications, pages 250-261, Heidelberg, 1998. Physica-Verlag. Contains revised versions of papers presented at the 1st GROOM-Workshop on the Unified Modeling Language (UML) in Mannheim (Germany) on Oct. 10./11., 1997.

OMG (Hrsg.): Unified Modeling Language Specification, June 1999. Version 1.3.

Paul, G.; Endig, M.; Sattler, K.-U.: A Component-based Integration Framework for Technical Information Systems. In (Grundy 1998), pages 27-33.

Pohl, K.: A Process Centered Requirements Engineering Environment. Dissertation, RWTH Aachen, 1995.

Rumbaugh, J.; Jacobson, I.; Booch, G.: The Unified Modeling Language Reference Manual. Object Technology Series. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

Sattler, K.-U.: Tool-Komposition in integrierten Entwurfsumgebungen. Dissertation, Otto-von-Guericke-Universität Magdeburg, 1998.

Schefström, D.; van den Broek, G.: Tool Integration - Environments and Frameworks. Wiley Series in Software Based Systems. John Wiley Ltd., 1992.

Schneider, W.: Anwenderorientierte Integration von CAE-Systemen – Ein Verfahren zur Realisierung eines durchgehenden Informationsflusses entlang des Produktentwicklungsprozesses. Dissertation, Heinz Nixdorf Institut, Universität-GH Paderborn, 1998. HNI-Verlagsschriftenreihe, Band 37; Rechnerintegrierte Produktion;

Scheer, A.-W.; Nüttgens, W.; Zimmernann, V.: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) - Methode und Anwendung. Veröffentlichungen des Instituts für Wirtschaftsinformatik 141, Institut für Wirtschaftinformatik – Universität des Saarlandes, Sarbrücken, Germany, Mai 1997.

Staud, J.: Geschäftsprozeßanalyse mit Ereignisgesteuerten Prozeßketten. Springer-Verlag Inc., New York, NY, USA, 1999.

Wasserman, A. I.: Tool Integration in Software Engineering Environments. In Long, F. (editor): Software Engineering Environments: Proceedings of the International Workshop on Environments, volume 467 of Lecture Notes in Computer Science, pages 137-149, Berlin, September 1990. Springer-Verlag Inc.

Warmer, J.; Kleppe, A.: The Object Constraint Language – Precise Modeling with UML. Object Technology Series. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.