Prozessmodellierung mit dem Eclipse Process Framework

Elke Löschner
Siemens AG

C-LAB Report

Vol. 8 (2009) No. 06

Cooperative Computing & Communication Laboratory

ISSN 1619-7879

C-LAB ist eine Kooperation der
Universität Paderborn und der Siemens AG
www.c-lab.de
info@c-lab.de

C-LAB Report

Herausgegeben von
Published by

Dr. Wolfgang Kern, Siemens AG
Prof. Dr. Franz-Josef Rammig, Universität Paderborn

Das C-LAB - Cooperative Computing & Communication Laboratory - leistet Forschungs- und Entwicklungsarbeiten und gewährleistet deren Transfer an den Markt. Es wurde 1985 von den Partnern Nixdorf Computer AG (nun Siemens AG) und der Universität Paderborn im Einvernehmen mit dem Land Nordrhein-Westfalen gegründet.
Die Vision, die dem C-LAB zugrunde liegt, geht davon aus, dass die gewaltigen Herausforderungen beim Übergang in die kommende Informationsgesellschaft nur durch globale Kooperation und in tiefer Verzahnung von Theorie und Praxis gelöst werden können. Im C-LAB arbeiten deshalb Mitarbeiter von Hochschule und Industrie unter einem Dach in einer gemeinsamen Organisation an gemeinsamen Projekten mit internationalen Partnern eng zusammen.

C-LAB - the Cooperative Computing & Cooperation Laboratory - works in the area of research and development and safeguards its transfer into the market. It was founded in 1985 by Nixdorf Computer AG (now Siemens AG) and the University of Paderborn under the auspices of the State of North-Rhine Westphalia.
C-LAB's vision is based on the fundamental premise that the gargantuan challenges thrown up by the transition to a future information society can only be met through global cooperation and deep interworking of theory and practice. This is why, under one roof, staff from the university and from industry cooperate closely on joint projects within a common research and development organization together with international partners. In doing so, C-LAB concentrates on those innovative subject areas in which cooperation is expected to bear particular fruit for the partners and their general well-being.

ISSN 1619-7879

C-LAB
Fürstenallee 11
33102 Paderborn
fon: +49 5251 60 60 60
fax: +49 5251 60 60 66
email: info@c-lab.de
Internet: www.c-lab.de

Inhalt

Abbildungen

Abkürzungen

ATESST	Advancing Traffic Efficiency and Safety through Software Technology (Forschungsprojekt, siehe atesst.org)
AUTOSAR	Automotive Open System Architecture, siehe autosar.org
BPM	Business Process Management
CIP	Continuous Improvement Process
CMMI	Capability Maturity Model Integration, siehe sei.cmu.edu/cmmi/
EAST	Electronic Architecture & Software Technology
EAST-ADL	EAST Architecture Description Language, siehe [1]
EPF	Eclipse Process Framework, siehe eclipse.org/epf
GI	Gesellschaft für Informatik e. V., siehe gi-ev.de
ITIL	Information Technology Infrastructure Library
MOF	MetaObject Facility, siehe omg.org/mof/
OMG	Object Management Group, siehe omg.org
PMBOK	Project Management Body of Knowledge
SEM	Systementwicklungsmethodik
SPEM	Software & Systems Process Engineering Metamodel, siehe omg.org/spec/SPEM
TIMMO	Timing Model (Forschungsprojekt, siehe www.timmo.org)
UML	Unified Modeling Language, siehe omg.org/spec/UML

1 Einleitung und Überblick

Die Entwicklung informationstechnischer Systeme steht heute unter hohem Druck, rasant zunehmende Komplexität zu wettbewerbsfähigen Preisen beherrschen zu müssen. Die Anforderungen hinsichtlich Funktionalität, Leistungsfähigkeit, Qualität, Sicherheit, Umweltverträglichkeit und Nachhaltigkeit steigen. Kurzfristig orientierte wirtschaftliche Ziele und globaler Wettbewerb verlangen immer kürzere Entwicklungszyklen.

Es erfordert auf einander abgestimmte Anstrengungen, Exzellenz und stetige Verbesserung in vielen Disziplinen, um die wachsenden Herausforderungen dauerhaft erfolgreich zu meistern. Diese Erkenntnis führt zu einer prozessorientierten Vorgehensweise: Definierte Prozesse erzeugen zuverlässig Ergebnisse geplanter Qualität; Messwerte, abgeleitet von regelmäßig überprüften und angepassten Zielvorgaben, erfassen Prozessparameter und Ergebniseigenschaften und dienen zur Bewertung und weiteren Verbesserung der Prozesse und ihrer Ergebnisse. Ein solches Vorgehen folgt selbst einem Prozessmuster, und zwar dem eines kontinuierlichen Verbesserungsprozesses[1], der auf das Prinzip der ständigen Verbesserung von Deming [4] zurückgeht.

Der vorliegende Bericht befasst sich mit der Definition von Prozessen als Muster für deren praktische Ausführung. Das Forschungsprojekt „TIMMO – Timing Model" [17] behandelte konkret den Prozess der Fahrzeugentwicklung, in den Aspekte des Zeitverhaltens beteiligter Komponenten möglichst von Beginn an einbezogen werden sollten. Im Projekt wurden Recherchen zur Auswahl geeigneter Mittel für die Beschreibung der Entwicklungsmethodik angestellt und praktische Erfahrungen bei der Anwendung gesammelt. Der Bericht beruht im Wesentlichen auf den dabei gewonnenen Erkenntnissen, die in anderen Entwicklungsdomänen ebenso nutzbar sind. So richtet sich der Bericht an alle Leser, die selbst an der Gestaltung und Verbreitung von Methoden und Prozessen arbeiten, eventuell mit den bisher eingeschlagenen Wegen unzufrieden sind und nach Ansätzen suchen, die ihren Bedürfnissen besser gerecht werden.

In Kapitel 2 wird die Verwendung der für das weitere Verständnis wichtigen Begriffe „Prozess" und „Prozessmodell" erläutert. Das Kapitel 3 geht auf Motive und Ziele der Prozessmodellierung ein und stellt die UML-basierte Modellierung mit dem Standard SPEM 2.0 und dem Eclipse Process Framework vor. Als Praxisbeispiele präsentiert Kapitel 4 auszugsweise die TIMMO-Methodik aus dem bereits erwähnten Forschungsprojekt und die interne Systementwicklungsmethodik „SEM New Generation" von Siemens IT Solutions and Services. Der Bericht schließt mit einer zusammenfassenden Bewertung in Kapitel 5.

[1] im anglo-amerikanischen Sprachraum: Continuous Improvement Process (CIP), in Japan: Kaizen

2 Begriffe

Die folgenden Erläuterungen dienen dazu, eine gemeinsame Basis für das Verständnis der weiteren Kapitel zu schaffen.

2.1 Prozess

Gemeint ist hier die Beschreibung einer zielgerichteten Folge von Aktivitäten, die ausgeführt werden müssen, um den Zweck des Prozesses zu erfüllen. Ein Prozess stellt also ein Handlungsmuster dar. Wenn die Handlungen tatsächlich vollzogen werden, sprechen wir von der Ausführung eines Prozesses, was im Kontext der Prozessmodellierung auch als „Instanziierung“ bezeichnet wird. Auf dieses Konzept geht die Erklärung des Begriffs „Prozessmodell“ noch näher ein.

Dieser Bericht beschäftigt sich mit Entwicklungsprozessen. Es handelt sich um Prozesse, deren Zweck die Entwicklung von Produkten[2] ist. Typisch für solche Prozesse ist, dass sie meist Aktivitäten umfassen zur Erhebung und Festlegung von Anforderungen an das Produkt, zur Planung der Gestaltung des Produktes, zur eigentlichen Realisierung des geplanten Produktes und zur Prüfung des Produktes gegen seine Anforderungen. Im Fokus der Prozessdefinition liegen eher technische oder fachliche Aspekte als wirtschaftliche, geschäftsbezogene. Entwicklungsprozesse können sich je nach Entwicklungsdomäne und nach Art des Produktes stark unterscheiden. Die Differenzierung wird umso stärker, je genauer der Prozess definiert werden soll. Diese Genauigkeit, auch als Granularität bezeichnet, hängt wiederum von den Zielen ab, die mit der Prozessbeschreibung verfolgt werden. Auf Motive und Ziele wird in Kapitel 3.1 näher eingegangen werden. Die Entwicklung von Produkten ist oft eingebettet in umfassendere Prozesse, die z. B. den Lebenszyklus von Produkten oder die Abwicklung von Projekten beschreiben.

Eine weitere wichtige Kategorie von Prozessen, mit denen sich die Prozessmodellierung befasst, sind Geschäftsprozesse[3]. Die Betrachtung von Geschäftsprozessen ist immer an den Geschäftszielen einer Unternehmung ausgerichtet. Sie geschieht in der Regel unter dem Blickwinkel einer anschließenden Geschäftsprozessoptimierung. Bei der Erfassung von Geschäftsprozessen erfolgt eine auf das Unternehmen bezogene Gliederung der Prozesse z. B. in Kern-, Stütz- und Management-Prozesse. Wie bei Entwicklungsprozessen sind auch bei Geschäftsprozessen die jeweiligen Modellierungsziele ausschlaggebend dafür, wie bei Modellierung vorgegangen wird, welche Granularität gewählt wird und welche Standards oder Werkzeuge angewandt werden. Die Nutzung des Instrumentariums der Wirtschaftsinformatik soll zu einer besseren Synchronisation zwischen Geschäfts- und IT-

[2] *Produkt* ist hier allgemein als Ergebnis eines Entwicklungsprozesses zu verstehen. Es umfasst sowohl Produktentwicklung in Hinblick auf Serienprodukte als auch einmalig zu entwickelnde Ergebnisse beliebiger Art.

[3] Ein Standardwerk zu diesem Thema ist z. B. [13].

Strategie führen und letztere stärker auf die Unterstützung der Geschäftsziele ausrichten.

Obwohl prinzipiell nichts dagegen spricht, Entwicklungs- und Geschäftsprozesse auf gleiche Weise zu behandeln, finden sich doch Unterschiede in den einschlägigen Methoden und Werkzeugen sowie in den generischen Ansätzen, deren Ursachen vermutlich hauptsächlich in den unterschiedlichen Wurzeln und Entstehungsgeschichten beider Domänen (eher technische versus eher wirtschaftliche Orientierung) liegen. Dies näher zu untersuchen, ist nicht das Ziel des vorliegenden Berichts. Er befasst sich vorrangig mit Entwicklungsprozessen, da die ihm zu Grunde liegenden Projekterfahrungen in diesem Bereich gesammelt wurden. Wenn sich Berührungspunkte zur Geschäftsprozessmodellierung ergeben, wird darauf entsprechend hingewiesen.

2.2 Prozessmodell und Modellierungsebenen

In der Terminologie der Modellierung bezeichnet „Prozessmodell“ einen Prozess als Handlungsmuster, wie er im vorangehenden Unterkapitel erläutert wurde. Der Zusatz „-modell“ macht explizit deutlich, dass es sich nicht um eine konkrete Prozessinstanz handelt. Das Modell ist also, vereinfacht dargestellt, eine Vorschrift, die beschreibt, wie ein Prozess auszuführen ist, um mit dem Prozessmodell konform zu sein. Es lassen sich zunächst die folgenden zwei Modellierungsebenen identifizieren:

Modellierungsebene	**Objekt**	**Beispiel**
Modellebene (Modell)	Prozessmodell	V-Modell XT® [4] Version 1.3
Ausführungsebene (Instanz)	Prozess, der ausgeführt wird	Systementwicklungsprojekt für das System xyz ...

Im Kapitel 3.2 „Standards und Werkzeuge“ wird auf Ansätze eingegangen werden, die die Modellierung von Prozessen auf eine gemeinsame Grundlage stellen, d. h. formalisieren und standardisieren. Dadurch können bestimmte für Prozessmodellierungen notwendige und geeignete Konstrukte allgemein bereitgestellt werden und müssen nicht jedes Mal „neu erfunden“ werden. Standardisierung ermöglicht außerdem den Vergleich und den Austausch von Prozessmodellen und bietet eine Basis für die Entwicklung von unterstützenden Werkzeugen. Diesen Standard für die Modellebene schafft die so genannte Metamodellebene:

[4] V-Modell XT® ist eine geschützte Marke der Bundesrepublik Deutschland. Näheres siehe [18].

Modellierungsebene	Objekt	Beispiel
Metamodellebene (Metamodell)	Metamodell für Prozessmodelle	V-Modell XT Metamodell Version 1.3
Modellebene (Modell)	Prozessmodell	V-Modell XT Version 1.3 (oder organisations-spezifische Ableitung)
Ausführungsebene (Instanz)	Prozess, der ausgeführt wird	Systementwicklungsprojekt für das System xyz ...

Die Entwickler des V-Modells, das hier als Beispiel genannt wird, haben ein eigenes Metamodell definiert, das es erleichtert, organisationsspezifische Ableitungen des Prozessmodells zu erzeugen, die mit dem Metamodell konform sind. So wird sichergestellt, dass alle abgeleiteten Prozessmodelle definierte Grundeigenschaften haben, die für das V-Modell wesentlich sind. Auf Basis des Metamodells wird ein V-Modell XT Editor als Werkzeug angeboten, der die Konformität automatisch garantiert. Das Metamodell bildet außerdem auch die Basis für Weiterentwicklungen des V-Modells selbst.

3 Modellierung von Entwicklungsprozessen

3.1 Motivation und Ziele

Eingangs wurde die Hinwendung zur Prozessorientierung mit dem Prinzip der ständigen Prozessverbesserung in Verbindung gebracht. Dieser Ansatz fand in der Praxis zuerst Verbreitung in der industriellen Produktion. Als Vorreiter gilt hier die japanische Automobilindustrie (vgl. [9]). Auch im Bereich der Software-Entwicklung wurde die Bedeutung dieses Ansatzes für Forschung und Praxis erkannt. In die Forschung fand die Prozessorientierung Eingang auf Basis der Arbeit von Osterweil [12], der den Prozess der Software-Entwicklung selbst als Software bezeichnete, die modelliert werden könne. Humphrey begründete am Software Engineering Institute der Carnegie Mellon University mit seinen Arbeiten zum Software-Prozess [8] die Entwicklung des heutigen CMMI[5], das eines der wesentlichen Assessment-Modelle für Software-Entwicklungsprozesse und -Organisationen ist. Die beiden Autoren repräsentieren nach Botella et al. in [3] zwei wichtige Motive zur Modellierung von Software-Entwicklungsprozessen:

- besseres, genaueres Verständnis und Formalisierung des Prozesses,
- Bewertung und Verbesserung des Prozesses (Assessment & Improvement).

Ein weiteres Motiv, das bei Gnatz et al. in [5] zu finden ist, ist die

- Weitergabe von kollektivem Projektmanagement-Know-how.

Letztere betrachten konkrete Software-Projekte als Instanzen eines Software-Prozessmodells, das evolutionär unter Einbeziehung der kollektiven Projektmanagement-Erfahrungen aller Projektleiter entsteht. Der Gedanke lässt sich erweitern, indem man nicht nur die Projektmanagement-Erfahrungen, sondern auch das Wissen und die Erfahrungen aller übrigen an Software-Projekten beteiligten Rollen einbezieht. So erhält man als erweitertes Motiv für die Entwicklung und Pflege von Prozessmodellen die

- Erfassung, Verbreitung und Fortschreibung von Best Practices.

Es wird deutlich, dass die unterschiedlichen Motive auch unterschiedlichen Zielen dienen. Der stärker forschungsorientierte Formalisierungsansatz verfolgt letztlich das Ziel der Automatisierung von Software-Prozessen (Enactment). Dies ist in letzter Konsequenz nur möglich, wenn Prozesse vollständig und detailliert beschrieben sind. Die Assessment-Ansätze dienen dagegen einem ganzheitlichen, sehr generischen Konzept der Prozessverbesserung und befassen sich nicht damit, wie die betrachteten Prozesse in allen Details aussehen, solange sie geforderte Prozessparameter erfüllen. Voraussetzung für den Verbesserungsprozess ist die standardisierte Prozessdefinition. Der dritte oben genannte Ansatz verfolgt das Ziel der automatisierten Ableitung von konkreten Projektstrukturplänen aus Prozessmodellen, während die

[5] Capability Maturity Model Integration

Ausführung der Projekte auf konventionelle Weise „manuell" erfolgt. Strikte Formalisierung erfordern hier also nur die Aspekte, die die modellkonforme Ableitung einer konkreten Projektstruktur in Form von Aktivitäten und Produkten und deren logischen Abhängigkeiten sicherstellen.

Eine Kategorie von Prozessmodellen, deren Ziel die Vorgabe eines Prozessstandards ist, sind so genannte „Vorgehensmodelle". Ein besonders in Deutschland verbreitetes Vorgehensmodell für Systementwicklungsprojekte ist das V-Modell® [6], das in Kapitel 2.2 bereits als Beispiel erwähnt wurde. Typisch für Vorgehensmodelle ist, dass sie eher allgemeine Prozessmodelle sind, deren Instanzen nicht zur automatisierten Ausführung bestimmt sind. Sie sind in einer bestimmten Ausprägung standardisiert und modernere bieten in der Regel Spielraum zur Anpassung an spezifische Projektkontexte. Die derzeit aktuelle Version des V-Modells, das V-Modell XT in der Version 1.3 vom Februar 2009, drückt diese Intention im Namenszusatz „XT" für „Extreme Tailoring" deutlich aus. Zu diesem Zweck wird das als Dokumentation vorliegende Vorgehensmodell durch das Werkzeug „Projektassistent" ergänzt, mit dessen Hilfe das „Zuschneiden" auf ein Projekt modellkonform vorgenommen werden kann.

Im Benutzungskonzept des V-Modells ist außer dem Tailoring für einzelne Projekte auch vorgesehen, organisationsspezifische Vorgehensmodelle abzuleiten. Allerdings ist dabei zu bedenken, wie neue Versionen des allgemeinen V-Modells mit der Weiterentwicklung des abgeleiteten organisationsspezifischen Modells synchronisiert werden können[7]. Erleichtert wird dieser Vorgang durch die Vorgabe eines gemeinsamen Metamodells, ein Konzept, das im Kapitel 2.2 bereits vorgestellt wurde. Das Konzept der „Variabilität von Inhalten", z. B. implementiert im Eclipse Process Framework, ist ebenfalls ein Ansatz zur Lösung solcher Probleme (vgl. Kapitel 3.2.5.3).

Das V-Modell als Beispiel eines Vorgehensmodells zeigt einen weiteren Trend in der Behandlung von Entwicklungsprozessen auf: Es wird nämlich die Entwicklung von „Systemen" adressiert, nicht spezifisch Software, Hardware oder andere Entwicklungsdomänen. Das ist nicht nur als Symptom der Verallgemeinerung zu sehen, sondern trägt vor allem der Tatsache Rechnung, dass moderne Systeme oft aus Bestandteilen verschiedener Domänen bestehen, deren Entwicklung mit ihren komplexen Wechselwirkungen von Beginn an im Zusammenhang betrachtet werden muss. Zur Erfüllung dieser Anforderung zeigt das managementorientierte V-Modell nur generisch Wege auf. Wie sie in der Praxis in einer konkreten Entwicklungssituation fachlich umzusetzen ist, muss Gegenstand eines wesentlich spezifischeren Prozessmodells mit entwicklungstechnischem Fokus sein. Ein praktisches Beispiel aus dem Automobilbereich zeigt das Kapitel 4.1 anhand der TIMMO-Methodik.

---

6 V-Modell® ist eine geschützte Marke der Bundesrepublik Deutschland. Näheres siehe [18]

7 Auf die Relevanz dieser Überlegung deutet z.B. der 4. Workshop „Vorgehensmodelle in der Praxis - Evolution und Wandlungsfähigkeit" im Rahmen der 39. GI-Jahrestagung Informatik 2009 hin (siehe www.informatik2009.de).

3.2 Standards und Werkzeuge

3.2.1 Überblick

Im vorangehenden Kapitel wurden unterschiedliche Motive und Ziele der Modellierung von Entwicklungsprozessen behandelt. Es wurde auf die damit in Beziehung stehende Granularität der Prozessmodelle hingewiesen. Jedoch wurde noch nicht erläutert, wie Prozesse modelliert werden können, d. h. Vorgehensweise, Ausdrucksformen und Hilfsmittel. Dieser Bericht beabsichtigt nicht, hierzu eine vollständige Übersicht zu liefern. Wie einleitend erwähnt, werden vor allem Erkenntnisse und Erfahrungen aus dem Forschungsprojekt TIMMO (siehe [17]) und seinem Umfeld präsentiert.

Im Folgenden wird zunächst wegen ihrer großen Verbreitung eine Vorgehensweise skizziert und bewertet, die hier „Prozessmodellierung mit einfachen Mitteln“ genannt wird. Anschließend wird auf die Unified Modeling Language (UML) als wesentlicher Modellierungsstandard für informationstechnische Systeme eingegangen. Danach wird die UML-basierte Prozessmodellierung mit dem Software and Systems Process Engineering Metamodel (SPEM) vorgestellt und eine mögliche praktische Vorgehensweise unter Nutzung des Eclipse Process Framework (EPF) erläutert.

3.2.2 Prozessmodellierung „mit einfachen Mitteln“

Beschränkt man sich auf eine recht allgemeine Interpretation der in Kapitel 2 eingeführten Begriffe Prozess und Prozessmodell, so ist Prozessmodellierung spontan mit einfachen Mitteln computergestützt möglich, z. B. mit einem einfachen Zeichenprogramm oder einem Textverarbeitungsprogramm oder einer Kombination von beiden. Beliebt ist auch die Verwendung von Programmen, die zur Aufbereitung von Präsentationen konzipiert sind. Das Mittel der Wahl hängt meist vom Erfahrungshorizont des Autors und seiner gewohnten Arbeitsumgebung ab. Dieser Weg wird häufig eingeschlagen, wenn ohne genaue Analyse der Ziele und des Prozesses, in den sich die Modellierung einbettet, schnelle Ergebnisse benötigt werden, z. B. als einfache Prozessdokumentation für Assessments oder Audits. Der Vorteil der geringen zu erbringenden Vorleistungen und Investitionen wird durch weit reichende Nachteile mehr als kompensiert:

- Der Autor muss selbst manuell Konstrukte erstellen, um eine Prozessbeschreibungsform zu erfinden oder um entsprechende Diagramme aus geeigneten Methoden mit den Mitteln einfacher Zeichenprogramme abzubilden.
- Nicht formalisierte Beschreibungen fördern Missverständnisse und divergierende Interpretationen.
- Umfangreiche Prozessbeschreibungen werden schnell unübersichtlich, da die genutzten Werkzeuge keine für diesen Zweck geeigneten Strukturierungshilfen bieten.

- Aus den zuvor genannten Punkten folgt oft mangelnde Konsistenz in Darstellung und Inhalt des Modells, da es ausschließlich manueller Kontrolle unterliegt. Auch die Beteiligung mehrerer Autoren erhöht die Gefahr der Inkonsistenz durch uneinheitliche Darstellung und Semantik.
- Die Modelle sind wegen der manuellen Aufbereitung schlecht wartbar, was entweder zu hohem Wartungsaufwand oder zu mangelnder Aktualität der Modelle führt.
- Modelle, die in der beschriebenen Form dokumentiert sind, können oft nur schwierig mit aktuellen Arbeitshilfen für die praktische Anwendung hinterlegt oder verknüpft werden.
- Da die Modelle nicht formalisiert sind, ist eine automatisierte Ableitung von Instanzen oder eine automatisierte Ausführung nicht möglich. Die manuelle Ableitung ist anfällig für Fehler und kann nur schwer auf Konformität mit dem Modell geprüft werden.

Alle diese Punkte führen in Summe dazu, dass auf diese Art „modellierte" Prozesse in der Regel nicht in der Praxis gelebt werden und umgekehrt Weiterentwicklungen der Prozesse in der Praxis keinen Eingang in die Modelle bzw. deren Dokumentation finden. In den folgenden Kapiteln wird eine Alternative gezeigt, die zwar im Vorfeld etwas mehr Aufwand erfordert, in der kontinuierlichen Anwendung aber deutliche Vorteile bietet.

3.2.3 Unified Modeling Language (UML)

Die Unified Modeling Language (UML)[8] ist hervorgegangen aus verschiedenen objektorientierten Analyse- und Designmethoden, die zahlreich mit der Verbreitung der Objektorientierung bereits Ende der achtziger / Anfang der neunziger Jahre entstanden sind. Die anfangs unabhängig voneinander arbeitenden Methodenentwickler Booch, Rumbaugh und Jacobson trafen im Jahr 1995 bei der Firma Rational Software[9] zusammen und brachten ihre jeweiligen Vorarbeiten in einen gemeinsamen Entwurf für die UML ein, der in der Object Management Group (OMG) zur Standardisierung eingereicht wurde. Zweck der UML ist die Beschreibung von Softwaresystemen mittels unterschiedlicher Diagramme, deren Elemente und Semantik festgelegt sind. Die erste als OMG-Standard verabschiedete Version war im Jahr 2000 die UML 1.3. Einen bedeutenden Entwicklungssprung markierte UML 2.0 im Jahr 2005. Ihre Entstehungsgeschichte („Synthese aller guten Ideen in der OO-Modellierung") und die erfolgte Standardisierung der UML erklären, weshalb diese Beschreibungssprache sich im Bereich der Software-Entwicklung durchgesetzt hat. Die Vorteile ihrer Standardisierung seien hier zusammengefasst:

- breiter Austausch und Kompatibilität von Modellen
- Ausbildungs- bzw. Lernaufwand rentabel, da breite Einsatzmöglichkeit

[8] Derzeit aktuell ist UML 2.2 vom Februar 2009, vgl. omg.org/spec/UML.
[9] Im Jahr 2003 wurde Rational Software von IBM gekauft.

- daher „genug“ ausgebildete Modellierer bzw. Bücher und Trainingskurse verfügbar
- Entwicklung von unterstützenden Werkzeugen möglich, die wiederum den effizienten Einsatz der Sprache ermöglichen

Einerseits als Mangel, andererseits auch als Vorteil von UML kann gesehen werden, dass im Rahmen der UML kein Prozess vorgeschrieben ist, der die genaue Vorgehensweise bei Anwendung der UML festlegt. Möglicherweise trug gerade dieser Umstand zur hohen Verbreitung der UML bei. So überrascht es kaum, dass auch versucht wurde, UML zur Modellierung von Software-Entwicklungsprozessen zu verwenden[10]. Da UML jedoch sehr allgemein gehalten ist, fehlen hier spezifische Konstrukte, die typische Elemente aus Prozessmodellen abbilden, wie Aktivitäten, Rollen, Meilensteine etc.

Will man UML um solche Konstrukte ergänzen, bieten sich zwei Wege zur Erweiterung:

- per Erweiterung des zugrunde liegenden Metamodells; als „heavyweight“ bezeichnet, da die abgeleiteten Prozessmodelle nicht mehr UML-konform sind;
- per Definition von Stereotypen in Form eines UML-Profils; als „lightweight“ bezeichnet, da die abgeleiteten Prozessmodelle UML-konform sind und daher mit UML-basierten Werkzeugen bearbeitet werden können.

Diese Erweiterungsmöglichkeiten wurden auf verschiedene Arten genutzt, um UML-basierte Prozessmodellierungssprachen zu definieren. Einen Überblick darüber gibt z. B. der Artikel von Botella et al. [3]. Einer dieser Ansätze wurde von der OMG standardisiert: das Software and Systems Process Engineering Metamodel (SPEM), das im nächsten Kapitel vorgestellt wird.

3.2.4 Software and Systems Process Engineering Metamodel (SPEM)

3.2.4.1 Historie

Im Jahr 2002 wurde von der OMG erstmalig das Software Process Engineering Metamodel[11] (SPEM) standardisiert, und zwar sowohl als eigenständiges Metamodell auf Basis der OMG MetaObject Facility (MOF)[12] als auch die „lightweight“-Variante als UML-Profil. Letzteres diente dem praktischen Zweck, die Nutzung von UML als Notation für SPEM-konforme Modelle zu ermöglichen.

SPEM als Metamodell für Prozesse resultiert aus der Erkenntnis, dass es selbst für den begrenzten Bereich der Software-Entwicklung niemals nur ein

[10] Es sei hier an Osterweils These erinnert: „Software processes are software, too.“ [12].

[11] Der erweiterte Anwendungsbereich „Software *and Systems* ...“ wurde erst mit der Version 2.0 adressiert.

[12] MOF ist, vereinfacht gesagt, das Meta-Metamodell, das der gesamten OMG Model Driven Architecture zugrunde liegt, siehe omg.org/mof/.

einziges Prozessmodell geben wird, das als Standard für alle Entwicklungsprozesse dienen könnte, sondern dass es im Gegenteil viele, teils sehr unterschiedliche Prozessmodelle gibt und langfristig geben wird. Die Standardisierung eines Metamodells für Prozessmodelle sollte alle Vorteile einer Standardisierung bieten, auf die schon im Kapitel 3.2.3 über die UML eingegangen wurde, aber dennoch allgemein genug sein, um die ganze Breite von Prozessmodellen[13] abdecken zu können. Das bedeutete gleichzeitig, dass sein Verwendungszweck zunächst auf die Definition der Prozesse und ihrer Bestandteile im Sinne einer Beschreibung beschränkt war und sich ausdrücklich nicht auf deren Ausführung (Enactment) erstreckte, da hierfür ein höherer Detaillierungsgrad und genauere Regeln (Constraints) für die Instanziierung erforderlich sind.

Heute ist die im Jahr 2008 verabschiedete Version SPEM 2.0 (siehe [11]) gültig, die wesentliche Fortschritte gegenüber älteren Versionen enthält:

- Anpassung an UML 2.0; SPEM XML-Schema basierend auf MOF 2.0
- Berücksichtigung der Kritik von Implementierern der älteren Versionen hinsichtlich Inkonsistenzen, Anwendbarkeit und Funktionalität
- Erweiterungen in Hinblick auf die Verwendung in Werkzeugen zur Prozessausführung bzw. -automatisierung
- Abgleich mit anderen in Entwicklung befindlichen OMG-Standards als UML, z. B. hinsichtlich Geschäftsprozessen (siehe [10])[14]
- Erweiterung des Anwendungsbereichs von Software- auf Systementwicklungsprozesse

Als Motive sind außer der bloßen Aktualisierung die Verbesserung der Anwendbarkeit sowie die Ausweitung der möglichen Anwendungsbereiche auch in Richtung auf die Ausführung und Automatisierung von Prozessabläufen erkennbar.

3.2.4.2 Anwendungskonzept

Die Spezifikation SPEM 2.0 bietet neben dem Metamodell und dem UML-Profil zur Beschreibung von Entwicklungsprozessen ein Anwendungskonzept (Conceptual usage framework), das Modellierung, Dokumentation, Verwaltung, Austausch und Ausführung von Entwicklungsmethoden und -prozessen umfasst. Dabei ist zu betonen, dass der direkte Anwenderkreis des Standards SPEM 2.0 in erster Linie Implementierer umfasst, die einzelne Werkzeuge oder integrierte Umgebungen auf Basis dieses Standards bereitstellen. Das Anwendungskonzept gibt den Implementierern einen Rahmen für mögliche Anwendungsszenarien, die sie mit ihren Werkzeugen unterstützen können.

---

[13] Der Begriff *Prozessmodell* sei hier verstanden, wie in Kapitel 2.2 erläutert.

[14] Hier findet sich ein Anknüpfungspunkt zu den in Kapitel 2.1 kurz erläuterten Geschäftsprozessen.

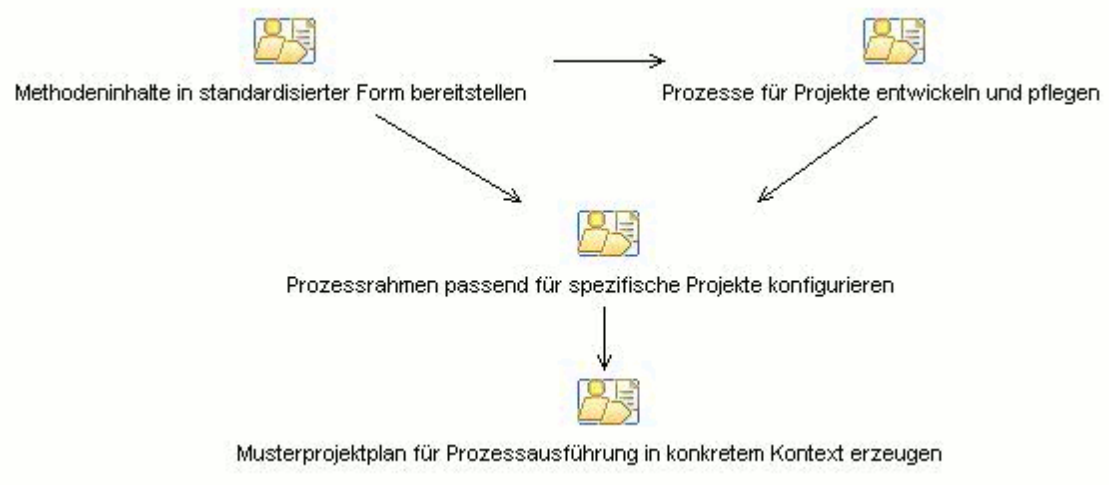


Abbildung 1: SPEM 2.0 Anwendungskonzept[15]

Abbildung 1 zeigt die vier miteinander in Beziehung stehenden Aktionsbereiche dieses Konzepts, die im Folgenden näher beschrieben werden:

1. Es werden *Methodeninhalte* in standardisierter Form erstellt und in Bibliotheken zur Nutzung abgelegt. Solche Inhalte sind z. B. Entwicklungsaktivitäten und Vorgaben, wie sie unter Anwendung geeigneter Methoden abzuwickeln sind, Arbeitsprodukte und ihre Strukturen sowie Fähigkeiten und Kenntnisse, die zur Ausführung der Aktivitäten benötigt werden.

   Akteure sind z. B. Methodenverantwortliche für Organisationen, Autoren oder Gremien, die Methoden entwickeln und allgemein als Wissensbasis bereitstellen, aber auch alle, die Methodeninhalte als Basis für die Entwicklung von Prozessen verfügbar machen möchten.

2. Darüber hinaus werden Methodeninhalte benutzt, um generische *Prozesse* zu entwickeln und zu pflegen. Vereinfacht dargestellt definieren Prozesse Abfolgen von Aktivitäten. Sie geben vor, welche Methoden in einem Projektlebenszyklus wann angewandt werden. Es werden strukturierende und für Abläufe wichtige Konstrukte, wie Phasen, Iterationen, Meilensteine etc., hinzugefügt. Gängige Vorgehensmodelle können auf diese Weise bereitgestellt werden.

   Akteure in diesem Bereich sind z. B. Autoren oder Gremien, die zentral an allgemeinen Prozessvorgaben arbeiten, oder Dienstleister, die solche Vorgaben für Unternehmen erstellen.

3. Ausgewählte Methodeninhalte und Prozesse werden für eine spezifische Einsatzumgebung, z. B. eine Entwicklungsabteilung, konfiguriert. Die *Konfiguration* definiert eine Auswahl von Methodeninhalten und Prozessen, die in einer spezifischen Umgebung zur Verfügung stehen. Generische Versionen von Prozessen können gezielt mit Hilfe von Variabilitätskonzepten auf die Einsatzumgebung zugeschnitten werden.

[15] Aktivitäten-Diagramm erstellt mit EPF Composer Version 1.5, vgl. Kapitel 3.2.5

Akteure sind hier z. B. Prozess- oder Methodenverantwortliche für eine Organisation oder Organisationseinheit.

4. Aus einer Konfiguration von Methoden und Prozessen kann ein Musterprojektplan für die *Ausführung* eines konkreten Projekts abgeleitet und bei Bedarf nochmals an den spezifischen Kontext angepasst werden. Dieser Aktionsbereich umfasst alles, was notwendig ist, um die Voraussetzungen für die Ausführung zu schaffen. Dahinter könnte z. B. der Übergang zu einem Werkzeug für Projekt- oder Ressourcenplanung oder zu einer Workflow-Engine liegen.

   Akteure sind in diesem Bereich z. B. Projektleiter von konkreten Projekten.

Neben den Akteuren, den aktiv gestaltenden Anwendern, gibt es für jeden Aktionsbereich „passive“ Nutzer. Die jeweiligen Ergebnisse (Methodeninhalte, Prozesse, Konfigurationen, Projektpläne) dienen ihnen entweder direkt als Dokumentation und Handlungsanleitung oder indirekt als Input für einen der anderen Aktionsbereiche.

3.2.4.3 Grundelemente für Methoden und Prozesse

Nach dem Überblick über das Anwendungskonzept werden hier wichtige Grundelemente von SPEM 2.0 (siehe Abbildung 2) für die Modellierung von Methoden und Prozessen kurz vorgestellt.[16]

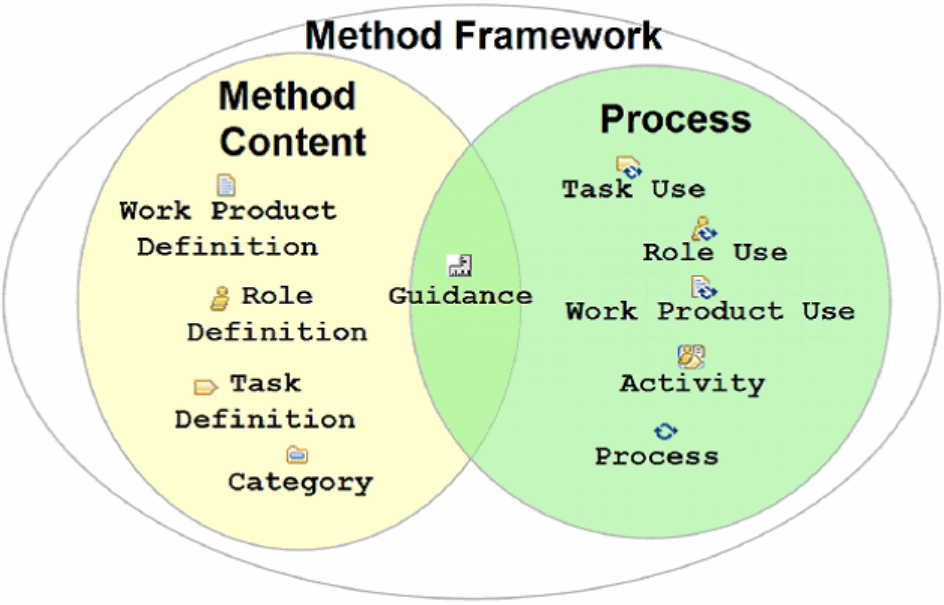


Abbildung 2: SPEM 2.0 Grundelemente (Quelle: [11])

[16] Da die SPEM-Spezifikation in englischer Sprache verfasst ist, werden hier zwecks Eindeutigkeit die englischen Namen für die Grundelemente verwendet.

Name	Beschreibung
Method Content	Grundelemente für Methoden
Work Product Definition	definiert ein Methodenelement, das mittels definierter Aufgaben (→Task Definition) erzeugt, benutzt oder verändert wird. Zwischen Arbeitsproduktdefinitionen können generische, nicht auf eine bestimmten Prozess oder Lebenszyklus bezogene Beziehungen bestehen, die z. B. Aggregation, Komposition oder logische Abhängigkeit ausdrücken.
Role Definition	definiert eine Rolle anhand von Fähigkeiten, Kompetenzen und Verantwortlichkeiten. Rollendefinitionen werden benutzt, um in Aufgabenbeschreibungen (→Task Definition) zu definieren, wer (im Sinne einer Rolle, nicht einer Person) die Aufgabe ausführt, sowie um festzulegen, für welche Arbeitsprodukte (→Work Product Definition) eine Rolle unabhängig von bestimmten Aufgaben generell verantwortlich ist.
Task Definition	definiert eine Aufgabe, welche von zugeordneten Rollen (→Role Definition) auszuführen ist. Eine Aufgabe ist verknüpft mit Input- und Output-Arbeitsprodukten (→Work Product Definition). Input-Produkte werden unterschieden nach verpflichtenden (mandatory) und optionalen (optional). Die Beziehungen zu den Arbeitsprodukten sind hier exemplarisch als Bestandteil der Definition zu verstehen, nicht als Verweise auf konkrete Instanzen von Arbeitsprodukten.
Category	ist ein beschreibendes Element, das zur Strukturierung beliebiger beschreibender Elemente dient. Mit Hilfe von Kategorien lassen sich nach beliebigen Kriterien hierarchische Strukturen innerhalb der Methodeninhalte aufbauen. Es gibt vordefinierte Arten von Kategorien wie z. B. „Rolesets“: Sammlungen von Rollendefinitionen nach frei wählbaren Kriterien.
Guidance	ist ein Element, das ergänzende Informationen oder Hilfsmittel sowohl zu Methoden- als auch zu Prozesselementen enthält. Es gibt unterschiedliche Arten dieses Elements, die sich z. B. durch ihren Aufbau oder ihre typischen Inhalte auszeichnen. Beispiele sind Richtlinien, Templates, Checklisten etc. Die Nutzung dieses Elements ermöglicht es, Definitionen relativ knapp und generisch zu halten und durch Verweise auf regelmäßig aktualisiertes „Hintergrundmaterial“ zu ergänzen.

Name	Beschreibung
Process	Grundelemente für Prozesse
Task Use	spezifiziert die Einbindung einer Aufgabendefinition in eine bestimmte Aktivität.
Role Use	spezifiziert eine Rolle, die eine bestimmte Aktivität ausführt oder an der Ausführung mitwirkt.
Work Product Use	spezifiziert ein Arbeitsprodukt, das an einer Aktivität beteiligt ist, und zwar direkt als Input oder Output der Aktivität oder indirekt innerhalb von Unter-Aktivitäten oder in Beziehungen anderer Elemente, die der Aktivität zugeordnet sind.
Activity	ist ein Oberbegriff für Prozessbestandteile. Auf oberster Betrachtungsebene ist ein Prozess selbst eine Aktivität. Spezielle, vordefinierte Arten von Aktivitäten sind Phase, Iteration und Meilenstein. Es kann eine Abfolge von Aktivitäten definiert werden. Aktivitäten können geschachtelt und zur Verknüpfung mit Methodendefinitionen genutzt werden (→Task Use, Role Use, Work Product Use).
Process: Delivery Process	repräsentiert einen vollständigen Prozess für einen kompletten Projektablauf von Start bis Ende mit vorgegebenen Phasen, Iterationen und Aktivitäten. Der Musterprozess definiert in Strukturplänen (Breakdown structures), welche Ergebnisse wie von wem erzeugt werden.
Process: Process Pattern	beschreibt einen Prozess, der eine wiederverwendbare Folge von Aktivitäten (mit Aufgaben, Arbeitsprodukten und Rollen) zusammenfasst, die einem bestimmten Zweck dient. So ein Prozessbaustein soll unabhängig von bestimmten Phasen oder Iterationen eines Projektlebenszyklus definiert werden und daher an beliebigen Stellen in Delivery-Prozessen einsetzbar sein.

Alle weiteren Details des Metamodells können bei Bedarf direkt in der SPEM-Spezifikation [11] nachgelesen werden. Für potenzielle Modellierer, die in diesem Bericht vorrangig angesprochen werden sollen, ist die Dokumentation SPEM-basierter Werkzeuge wichtiger, da sie den tatsächlichen Umfang und die Benutzung der jeweiligen Implementierung beschreibt.

Im Anhang der Spezifikation von SPEM 2.0 werden elf anschauliche Fallstudien und Anwendungsbeispiele für Methoden- und Prozessmodellierung mit SPEM-Konzepten mitgeliefert (siehe Annex C in [11]). Darunter sind z. B.

ITIL[17]-konforme Prozesse, eine Re-Modellierung des PMBOK[18] oder ein CMMI[19]-basiertes Prozess-Repository bei IBM. Mehrere Beispiele benutzen als Modellierungsumgebung das Eclipse Process Framework, mit dem sich das folgende Kapitel befasst.

3.2.5 Eclipse Process Framework (EPF)

3.2.5.1 Überblick

Im Hinblick auf Machbarkeitsprüfung und Verfügbarkeit von SPEM 2.0 weist bereits die Spezifikation auf die Open-Source-Software „Eclipse Process Framework“ (EPF) hin, die weite Teile des Standards implementiert und kostenfrei verfügbar ist[20]. Bestandteile von EPF sind derzeit das Werkzeug EPF Composer und einige damit modellierte Inhalte wie der iterative Software-Entwicklungsprozess OpenUP/Basic und einige allgemeine Beschreibungen von so genannten „Practices“, die in unterschiedlichen agilen Entwicklungsprozessen zum Einsatz kommen können.

Einen ausführlichen Einstieg in EPF, den EPF Composer und die Nutzung bieten die Dokumente [6] und [7]. Bisher wird mit dieser Werkzeugumgebung die Modellierung und Dokumentation von Methoden und Prozessen unterstützt, während die Ausführung von Prozessen innerhalb des EPF nicht vorgesehen ist. Verschiedene Export-Funktionen (XML-Format) ermöglichen die Übergabe an andere Werkzeuge, die eine Ausführung gestatten könnten.

EPF Composer ist verfügbar für Windows- und Linux-Systeme und kann nach dem Download und Entpacken direkt gestartet werden ohne Ausführung eines Installationsprogramms. Das Werkzeug ist mit einer guten Online-Hilfe ausgestattet, die auch nützliche Tutorials für die Einarbeitung enthält. Es sei hier besonders darauf hingewiesen, dass der EPF Composer nur von Anwendern benutzt wird, die selbst Autoren von Methoden und Prozessen sind oder zumindest vordefinierte Methoden und Prozesse anpassen oder zuschneiden. Alle anderen, die Ergebnisse dieser Arbeiten nutzen, greifen entweder mit ihrem gewohnten Webbrowser auf publizierte Inhalte zu oder verwenden exportierte Ergebnisse in ihrer jeweiligen Arbeitsumgebung (z. B. Import eines exportierten Musterprojektplans in ein Projektplanungswerkzeug).

Prinzipiell eignen sich bei Nutzung des UML-Profils von SPEM 2.0 auch UML-basierte Werkzeuge zur SPEM-Modellierung. Umfassender ist jedoch die mit dem EPF vorliegende Implementation, da sie neben den Elementen für Methoden- und Prozessmodellierung auch Konzepte für das Management von Methodenbibliotheken, für die Modularisierung in Form von Methoden-Plug-ins, für Methodenkonfiguration und Variabilität von Inhalten und für die automatisierte Publikation realisiert. Auf die Strukturierung und die Variabilität

---

17 Information Technology Infrastructure Library
18 Project Management Body of Knowledge
19 Capability Maturity Model Integration
20 Informationen und Download unter eclipse.org/epf/
Auf der EPF-Implementation setzt auch ein kommerzielles Werkzeug (IBM Rational Method Composer) auf, das hier nicht näher betrachtet werden soll.

von Inhalten als komplexere Konzepte gehen die beiden folgenden Unterkapitel ein. Ein weiteres Unterkapitel stellt das Konzept der Publikation von Inhalten vor, da es für die Nutzbarkeit der modellierten Methoden und Prozesse durch die jeweiligen Zielgruppen wesentlich ist.

3.2.5.2 Strukturierung von Inhalten

Im EPF Composer ist die größte Strukturierungseinheit für Inhalte die Methodenbibliothek (Method library). Zu einem bestimmten Zeitpunkt ist nie mehr als eine solche Methodenbibliothek geöffnet. Über Export-/Import-Funktionen ist ein Austausch von bzw. zwischen Methodenbibliotheken möglich. Eine Methodenbibliothek enthält Methoden-Plug-ins (Method plug-ins), von denen einige vielleicht importiert, andere selbst erstellt sind. Sie erlauben eine saubere Partitionierung von Inhalten. Außer den Plug-ins enthält jede Methodenbibliothek Konfigurationen (Configurations). Jede Konfiguration repräsentiert zum einen eine bestimmte Auswahl von Methodeninhalten, die für die Definition von Prozessen zur Verfügung stehen sollen, und zum anderen Sichten für die automatisierte Dokumentation (Publikation) der Konfiguration.

Die Struktur einer Methodenbibliothek mit nur einem Methoden-Plug-in ist in Abbildung 3 beispielhaft zu sehen und wird im Folgenden näher erläutert.

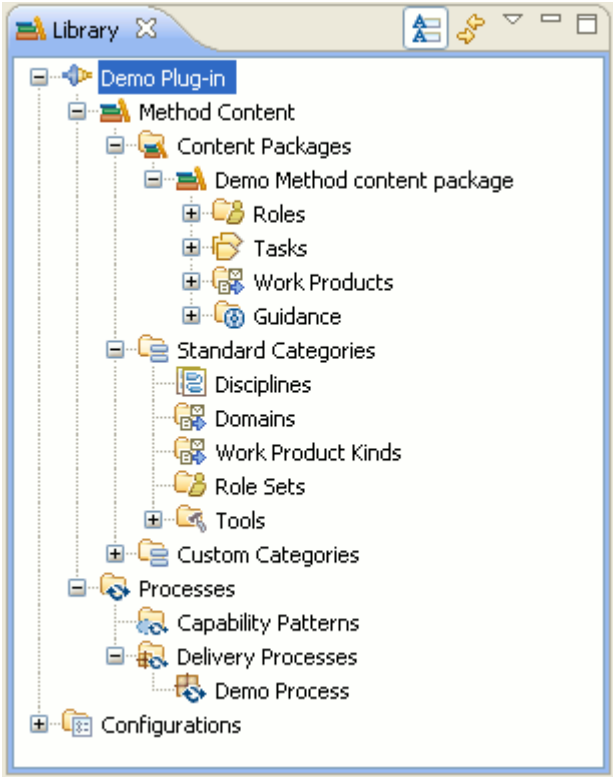


Abbildung 3: EPF Composer – Struktur einer Methodenbibliothek[21]

[21] Bild übernommen aus dem EPF Composer Version 1.5

Plug-ins bestehen aus Methodeninhalt (Method Content) und Prozessen (Processes). Der Methodeninhalt wird in Inhaltspakete (Content Packages) gegliedert, die eine weitere Modularisierung erlauben. In jedem Inhaltspaket werden die Grundelemente Rollen, Aufgaben, Arbeitsprodukte und Zusatzmaterial (Roles, Tasks, Work Products, Guidance) und deren Beziehungen definiert (vgl. Beschreibung der SPEM-Grundelemente in Kapitel 3.2.4.3).

Neben den Inhaltspaketen bietet jedes Methoden-Plug-in vordefinierte Standardkategorien (Standard Categories) und eine Oberkategorie für benutzerdefinierte Kategorien (Custom Categories), die eine paketübergreifende Gruppierung der Inhalte des Plug-ins ermöglichen. Diese Kategorien bilden die Grundlage für die spätere Definition von Sichten für die Publikation, sodass sie für eine zielgruppenorientierte Auswahl und Gliederung der Methodeninhalte genutzt werden können.

Prozesse werden zunächst in Prozessbausteine (Capability Patterns) und vollständige Prozesse (Delivery Processes) unterschieden (vgl. Beschreibung der SPEM-Grundelemente in Kapitel 3.2.4.3). Darunter werden entweder die einzelnen Prozesse abgelegt, oder es können Prozesspakete (Process Packages) gebildet werden, die Prozesse nach beliebigen Kriterien bündeln. Beim Erzeugen eines neuen Prozesses wird eine der in der Bibliothek definierten Konfigurationen ausgewählt, um festzulegen, welche Methodeninhalte bei der Modellierung des Prozesses sichtbar sind und benutzt werden dürfen.

Zur Beschreibung und Definition der verschiedenen Elemente im EPF Composer dient ein formular-basierter Editor, der dem Typ des jeweiligen Elements angepasst ist. Darin werden genau die für den Elementtyp erforderlichen und erlaubten Attribute und Beziehungen definiert oder bearbeitet. In den Editor für Prozesse sind außerdem Diagramm-Editoren eingebunden, mit denen Aktivitäten-Diagramme, Aktivitäten-Detail-Diagramme und Abhängigkeits-Diagramme für Arbeitsprodukte erstellt werden können.

3.2.5.3 Variabilität von Inhalten

Das Konzept der Variabilität, das mit dem Vererbungsmechanismus des zugrunde liegenden objektorientierten Paradigmas vergleichbar ist, bietet die Möglichkeit, bei der Definition neuer Methodeninhalte bereits vorhandene unterschiedlicher Herkunft als Basis zu benutzen und dabei zu ändern oder zu erweitern, ohne die Basisdefinition zu verändern und ohne Kopien zu erzeugen, die später zu Inkonsistenzen führen könnten.

Vorteilhaft ist, dass dieser Mechanismus angewandt werden kann, selbst wenn das Basiselement einem Methoden-Plug-in angehört, das gegen Änderungen geschützt ist – denn die Basisdefinition bleibt unverändert. So können vordefinierte Inhalte an eine spezielle Umgebung angepasst werden. Über Variabilität und kontrollierte Synchronisation können solche Anpassungen dann auch komfortabel auf importierte Updates der benutzten Basis-Plug-ins übertragen werden. So könnte z. B. die Lösung für das Problem der Aktualisierung organisationsspezifischer Ableitungen von einem allgemeinen Vorgehensmodell aussehen, das in Kapitel 3.1 aufgezeigt wurde.

Von Variabilität betroffen sind die Attribute eines Elements sowie die Beziehungen des Elements zu anderen Elementen. Welche Auswirkungen es im Einzelfall genau auf Attribute, Beziehungen und die Publikation der beteiligten Elemente gibt, hängt vom benutzten Variabilitätstyp ab. Unterstützt werden vier Typen: „Contributes“, „Extends“, „Replaces“ und „Extends and replaces“. Zu beachten ist außerdem, dass die Variabilitätsbeziehungen transitiv sind: Ein neues Element, das eine Variabilitätsbeziehung zu einem Basiselement hat, kann selbst wieder Basiselement einer weiteren Variabilitätsbeziehung sein usw., sodass sich die Auswirkungen über mehrere Ebenen erstrecken können. Es kann auch vorkommen, dass mehrere Elemente Variabilitätsbeziehungen zum selben Basiselement definiert haben. Dann gelten u. U. Vorrang- oder Ausschlussregeln. Diese Feinheiten sollen hier nicht näher erläutert werden, müssen aber bei Anwendung der Konzepte genau überprüft werden.

Fazit ist, dass es sich bei der Variabilität der Inhalte, wie generell beim Vererbungsprinzip in objektorientierten Sprachen, um einen mächtigen Mechanismus handelt, der gut überlegt und wohl dosiert sehr nützlich sein kann. Dabei unterstützt EPF Composer den Modellierer durch eine spezielle Browse-Funktion, die zu einem ausgewählten Methodenelement alle Variabilitätsbeziehungen in strukturierter Form anzeigt.

3.2.5.4 Publikation von Inhalten

Ein großer Vorzug des EPF Composers ist, dass die Nutzer der modellierten Methoden und Prozesse das Werkzeug selbst nicht zu beherrschen brauchen. Für sie stellen die Autoren ausgewählte Informationen in geeigneter Struktur als Website bereit, die mit dem gewohnten Webbrowser betrachtet wird. Bei der Aufbereitung der Dokumentation in Form einer Website werden die Beziehungen zwischen den Inhaltselementen durch Hyperlinks abgebildet. Die Navigation darin ist wesentlich komfortabler als das Blättern in herkömmlichen Dokumenten.

Für die Autoren wird die Erstellung der Dokumentation wesentlich dadurch erleichtert, dass das Erzeugen der Website nahezu vollautomatisch abläuft. Dieser Vorgang heißt „Publizieren einer Konfiguration“. Die Autoren konzentrieren sich allein darauf, die Konfiguration festzulegen und einige Parameter für die Publikation einzustellen.

Im Kapitel 3.2.5.2 über die Struktur der EPF-Inhalte war schon kurz erwähnt worden, dass eine Konfiguration zum einen dazu dient festzulegen, welche Methodeninhalte für eine Prozessmodellierung zur Verfügung stehen sollen, sowie zum anderen, welche Sichten mit welchen Inhalten als Website publiziert werden sollen. Die Auswahl von Methoden-Plug-ins und ggf. einzelnen Methodenpaketen für eine Konfiguration bestimmt also nicht nur die verfügbaren Elemente für die Prozessmodellierung sondern auch den maximal sichtbaren Umfang der publizierten Website.

Die Navigation der Website wird durch die in der Konfiguration angelegten Sichten (mindestens eine) vorgegeben. Jede Sicht stellt für den Benutzer einen „Einstiegspunkt“ in den Inhalt dar. Als mögliche Einstiege dienen vorde-

finierte und benutzerspezifische Kategorien (vgl. Abbildung 3), denen Methoden- oder Prozessinhalte zugeordnet sind. Es ist also wichtig, dass jedes Element, das in der generierten Dokumentation sichtbar sein soll, mindestens einer Kategorie zugeordnet ist, die in einer Sicht erscheint. Wenn eine Einstiegskategorie in weitere Kategorien gegliedert ist, ergibt sich daraus die hierarchische Navigationsstruktur der Sicht. Dieses Konzept kann sehr flexibel zur Präsentation derselben Methoden und Prozesse für unterschiedliche Zielgruppen genutzt werden. Beispiele hierfür sowie für die Methoden- und Prozessmodellierung mit dem EPF Composer allgemein sind im folgenden Kapitel zu finden.

4 Praxisbeispiele

4.1 Timing-orientierte Fahrzeugentwicklung – TIMMO-Methodik

4.1.1 Anwendungskontext und Modellierungsziele

Dieses Beispiel stammt aus der Mitarbeit im europäischen Forschungsprojekt TIMMO – Timing Model, dessen Website [17] ausführliche Projektinformationen bietet. In diesem Projekt arbeiten noch bis Ende 2009 Automobilhersteller und -zulieferer, Anbieter von Software-Werkzeugen, Forschungseinrichtungen und Siemens IT Solutions and Services zusammen. Wesentliches Projektziel ist die Verbesserung der Entwicklungsprozesse[22] für Fahrzeugelektronik hinsichtlich Dauer und Kosten. Die Reduzierung von Dauer und Kosten soll erreicht werden, indem Aspekte des Zeitverhaltens von Komponenten möglichst früh berücksichtigt und im modellbasierten Entwicklungsprozess durchgängig weiterverfolgt werden. Mängel hinsichtlich des Zeitverhaltens sollen nicht länger erst am Ende von Implementierung und Integration durch aufwändiges Testen identifiziert werden, da das zu umfangreichen Wiederholungen von früheren Entwicklungsphasen und erneuten aufwändigen Tests führt. Der verbesserte, timing-orientierte und modellbasierte Ansatz wird in TIMMO umgesetzt mittels Definition einer Sprache „Timing Augmented Description Language“ (TADL) zur Beschreibung der benötigten Timing-Konstrukte und einer Methodik zur Anwendung der Sprache in Entwicklungsprozessen („TIMMO Methodology“). Die Entwicklung und Modellierung dieser TIMMO-Methodik soll hier als Praxisbeispiel dienen[23].

Übergeordnete Vorgaben für die Modellierung der TIMMO-Methodik waren:

- Unterstützung der Anwendung der neuen Sprache TADL bei der Entwicklung von Fahrzeugelektronik
- Beschränkung auf fachliche, timing-relevante Aspekte im Entwicklungsprozess, Ausklammerung von Business- bzw. Management-Aspekten (d. h. kein vollständiger Prozess)
- Übertragbarkeit der TIMMO-Methodik auf verschiedenartige, unternehmensspezifische Entwicklungsprozesse
- Kompatibilität mit dem Industriestandard AUTOSAR (siehe [2])

Genutzt werden sollte die Methodik zum einen als Vorgehensleitlinie für die Entwickler von Validatoren im Rahmen des TIMMO-Projekts, zum anderen von Methoden- und Prozessautoren zur Integration in vorhandene und neue Entwicklungsprozesse, denen die Behandlung der Timing-Aspekte noch fehlt.

---

[22] Der Plural soll darauf hinweisen, dass es in der Industrie verschiedene Prozesse gibt und weiterhin geben wird, in die die Ergebnisse von TIMMO transferierbar sein sollen.

[23] Die vollständige Darstellung und Erläuterung der TIMMO-Methodik ist nach Projektabschluss öffentlich zugänglich in [16] über die Projekt-Website www.timmo.org.

4.1.2 Vorgehensweise und Ergebnisse

Zu Beginn der Methodik-Entwicklung wurden auf Grund von Erkenntnissen, wie sie im Kapitel 3 dargestellt wurden, der Standard SPEM 2.0 und seine Implementierung im EPF Composer 1.5 für die Formalisierung ausgewählt. Diese Entscheidung wurde zusätzlich dadurch begünstigt, dass SPEM bereits als Notation für die in TIMMO zu berücksichtigende AUTOSAR-Methodik eingesetzt worden war.

Aus dem Abgleich der Vorgaben für die TIMMO-Methodik mit dem Anwendungskonzept von SPEM 2.0 (vgl. Abbildung 1 auf Seite 15) ergab sich für TIMMO eine Beschränkung auf die Aktionsbereiche „Methodeninhalte in standardisierter Form bereitstellen“ und „Prozesse für Projekte entwickeln und pflegen“. Die Anpassung an eine konkrete Projektumgebung sowie die Ausführung darin waren keine Aufgaben im Rahmen der TIMMO-Methodik-Entwicklung. Gemäß TIMMO-Projektstruktur wurden die Aktivitäten der Methoden- und Prozessentwicklung zweimal durchlaufen, um eine initiale und darauf aufbauend eine zweite, verfeinerte Methodik-Version zu erzeugen. Im Folgenden wird diese Vorgehensweise als Beispiel für ein mögliches Vorgehen näher beschrieben und anhand ausgewählter Beispiele illustriert.

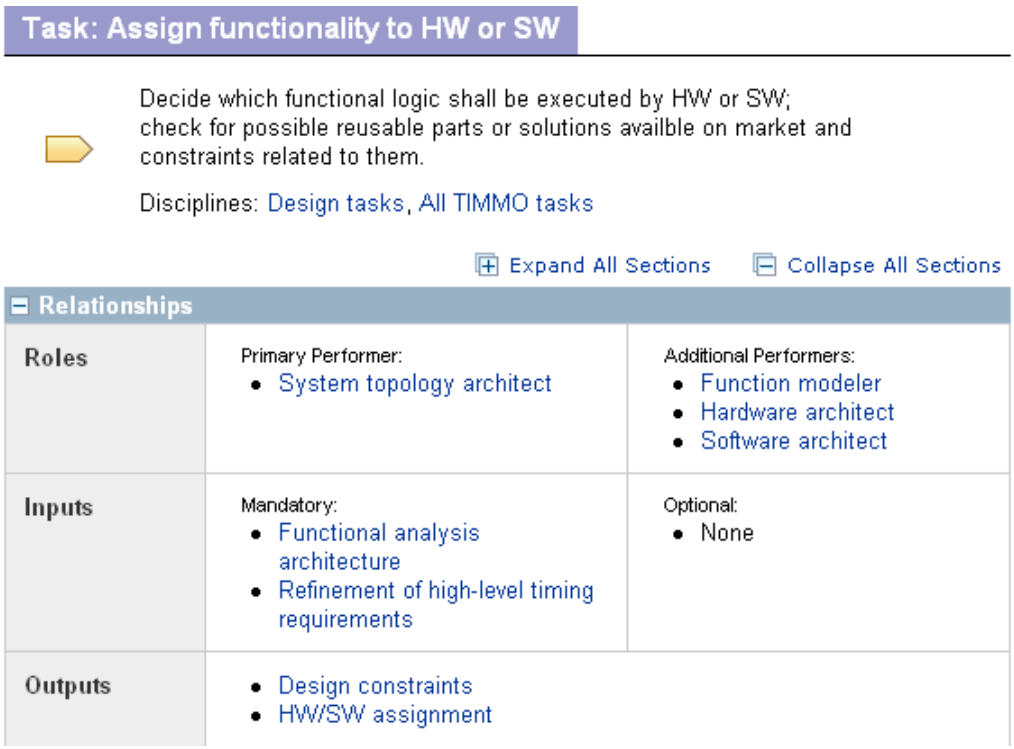
Phase I – TIMMO-Methodik Version 1

1. **Definition der relevanten Methodeninhalte**

   Zurückgegriffen wurde auf zuvor erarbeitete, nicht formalisierte Beschreibungen einiger typischer Entwicklungsszenarien. Aus diesen Szenarien wurde ein gemeinsamer Vorrat an Methodenelementen abgeleitet und im EPF Composer in einem „Method Content Package“ definiert, nämlich Rollen, Aufgaben, Arbeitsprodukte und ihre Querbeziehungen. Benutzt wurde dazu der formular-basierte, an den jeweiligen Elementtyp angepasste Editor.

   Abbildung 4 auf Seite 26 zeigt z. B. die Aufgabendefinition „Assign functionality to HW or SW“ mit Kurzbeschreibung, zugeordneten Rollen (Primary Performer, Additional Performers) und Arbeitsprodukten (Inputs, Outputs). Die Abbildung ist ein Schnappschuss der publizierten HTML-Darstellung, wie ein Nutzer sie im Browser sehen kann. Die Namen der Rollen und Arbeitsprodukte sind im Original Hyperlinks auf die jeweiligen Rollen- bzw. Arbeitsproduktdefinitionen.

   Die Vorgehensreihenfolge bei der Erfassung der Methodenelemente orientierte sich top-down an bereits eingeführten Abstraktionsebenen einer modellgetriebenen Fahrzeugentwicklung (vgl. EAST-ADL2 [1]): Feature level, Analysis level, Design level, Implementation level. Bei der Modellierung im EPF Composer wurde die Abstraktionsebene gleichzeitig als Kriterium zur Gruppierung von Rollen, Aufgaben und Arbeitsprodukten herangezogen. In Abbildung 4 erkennt man z. B. die Zugehörigkeit der abgebildeten Aufgabendefinition zur Disziplin „Design tasks“, die die Abstraktionsebene „Design level“ behandelt. „Disziplin“ (Discipline) ist eine vordefinierte Kategorie für Gruppen verwandter Aufgaben.



Task: Assign functionality to HW or SW

Decide which functional logic shall be executed by HW or SW; check for possible reusable parts or solutions availble on market and constraints related to them.

Disciplines: Design tasks, All TIMMO tasks

Expand All Sections Collapse All Sections

Relationships

Roles	Primary Performer: • System topology architect	Additional Performers: • Function modeler • Hardware architect • Software architect
Inputs	Mandatory: • Functional analysis architecture • Refinement of high-level timing requirements	Optional: • None
Outputs	• Design constraints • HW/SW assignment	

Abbildung 4: TIMMO V1 – Task Definition[24]

So bildeten die nach Abstraktionsebenen gruppierten Aufgaben-, Rollen- und Arbeitsproduktdefinitionen das Ergebnis dieses Arbeitsabschnitts. Im zweiten Abschnitt der Phase I folgte der ...

2. **Entwurf eines beispielhaften, idealisierten TIMMO-Musterprozesses unter Verwendung der zuvor definierten Methodeninhalte**

Gemäß SPEM-Anwendungskonzept bietet die Definition eines Vorrats an Methodenelementen große Freiheitsgrade in ihrer Verwendung und Kombination zu Prozessabläufen. Im Fall von TIMMO wäre durch die erstellte Sammlung von Methodenelementen allein die Anforderung einer Unterstützung der sinnvollen und wirksamen Anwendung der Sprache TADL noch nicht erfüllt. Daneben besteht jedoch auch die Anforderung, dass die TIMMO-Methodik in unterschiedliche Prozessumgebungen transferierbar sein muss, sodass sie nicht zu detailliert und restriktiv einen festen Prozessablauf vorschreiben kann, da er im Konflikt mit Vorgaben der Zielumgebung stehen könnte.

Als Lösungsweg wurde ein Kompromiss gewählt, bei dem die sinnvolle Verwendung der Methodenelemente mit Hilfe eines idealisierten Musterprozesses beispielhaft demonstriert wird. Dieser Musterprozess, „Common TIMMO Process“ genannt, wurde im zweiten Abschnitt der Phase I modelliert.

[24] HTML-Darstellung einer im EPF Composer 1.5 erstellten Task Definition

Idealisiert ist dieser Prozess hinsichtlich folgender Eigenschaften:

- Er repräsentiert die Entwicklung eines Systems von Start bis Ende in Form eines grob am V-Modell [18] orientierten Phasenmodells. – In der Praxis werden oft vorhandene Systeme oder Komponenten als Ausgangsbasis genutzt, oder es soll gar kein System entwickelt, sondern nur eine Machbarkeitsstudie erstellt werden.
- Er umfasst nur die timing-relevanten Methodenelemente. – Ein vollständiger Prozess umfasst weitere Engineering- und Management-Elemente, die später aus der gewählten Anwendungsumgebung übernommen werden müssen.
- Er verzichtet auf explizite Darstellung von Iterationen im Sinne von Wiederholungen einzelner oder mehrerer Aktivitäten einer Phase zur Verfeinerung oder Fehlerkorrektur. – Dies dient der vereinfachten Darstellung und der Flexibilität hinsichtlich der möglichen Entwicklungsprozesse, in die die TIMMO-Methodik transferierbar sein soll.

Beim Entwurf der Struktur des TIMMO-Prozesses wurde die Behandlung der Abstraktionsebenen auf jeweils korrespondierende Phasen abgebildet. Abbildung 5 zeigt die Prozessstruktur (Work Breakdown Structure) für die initiale Version des TIMMO-Musterprozesses im Prozess-Editor.

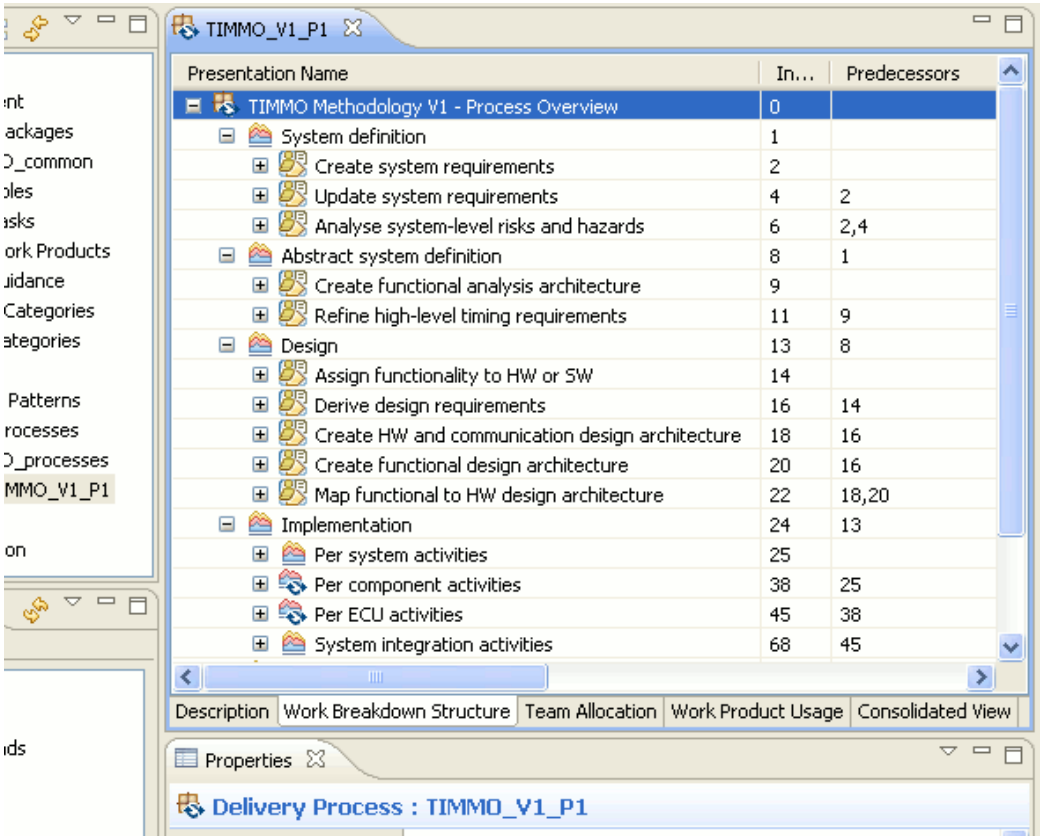


Abbildung 5:TIMMO V1 – Prozessstruktur[25]

[25] Ansicht übernommen aus dem EPF Composer 1.5

Zu erkennen sind die Prozess-Phasen (hier noch „System definition“, „Abstract system definition“, „Design“, „Implementation“ [26]). Die ersten drei Phasen sind direkt untergliedert in Aktivitäten, die Implementierungsphase in (Unter-)Phasen und Iterationen. Die komplexere Struktur der Implementierungsphase ergibt sich aus der Anlehnung an die AUTOSAR-Methodik, mit der Kompatibilität erreicht werden soll.

Durch die Prozessstruktur wird eine hierarchische Gliederung vorgegeben, die jedoch per se noch keinen Aktivitätenfluss (Workflow) definiert. Die Prozessstruktur in Abbildung 5 auf Seite 27 enthält jedoch bereits zusätzliche Informationen über den Ablauf, wie in der Spalte „Predecessors“ zu erkennen ist. Im formular-basierten Attribut-Editor unterhalb des Prozess-Editors stehen vier, auch aus Netzplänen bekannte, Typen von Vorgänger-Nachfolger-Beziehungen zur Auswahl, um Abläufe zu modellieren: „Finish to start“[27], „finish to finish“, „start to start“ und „start to finish“. Alternativ kann auch der Diagramm-Editor für Aktivitäten-Diagramme benutzt werden, um Aktivitätenflüsse graphisch zu modellieren (siehe Beispiel in Abbildung 6).

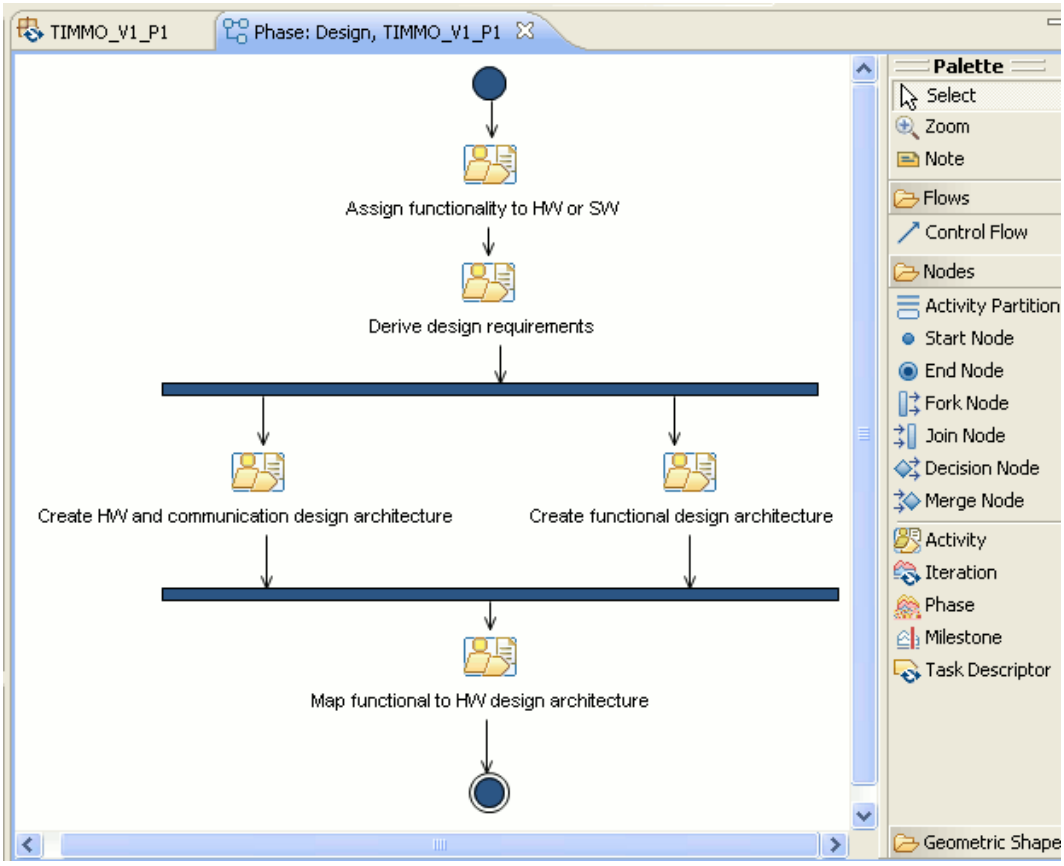


Abbildung 6: TIMMO V1 – Aktivitäten-Diagramm der Design-Phase[28]

[26] In Version 1 wurden noch nicht die endgültigen Phasen-Bezeichnungen verwendet.
[27] Aktivität A muss enden, damit Aktivität B starten kann. Die übrigen Typen sind analog dazu zu verstehen.
[28] Ansicht übernommen aus dem EPF Composer 1.5

Es wird nun gezeigt, wie zuvor definierte Methodenelemente in der Prozessstruktur wiederverwendet werden können. Im einfachsten Fall wird in eine Aktivität die Instanz einer Aufgabendefinition eingebunden. Abbildung 7 zeigt, wie die Aufgabendefinition „Assign functionality to HW or SW“ (vgl. Abbildung 4 auf Seite 26) in der gleichnamigen Aktivität der Phase „Design“ genutzt wird. Im Attribut-Editor werden die Eigenschaften (Properties) dieser Instanz im Prozess festgelegt. Dort wird im Feld „Method task“ die Beziehung zum Methodenelement gepflegt. Die Option „Synchronized with source“ entscheidet darüber, ob die Instanz ggf. bei Änderungen des Methodenelements aktualisiert werden soll oder generell nicht. Unter dem „Dach“ einer Aktivität im Prozess können aber auch mehrere vordefinierte Aufgaben zusammengefasst werden, deren Abfolge innerhalb dieser Aktivität durch die beschriebenen Vorgänger-Nachfolger-Beziehungen modellierbar ist.

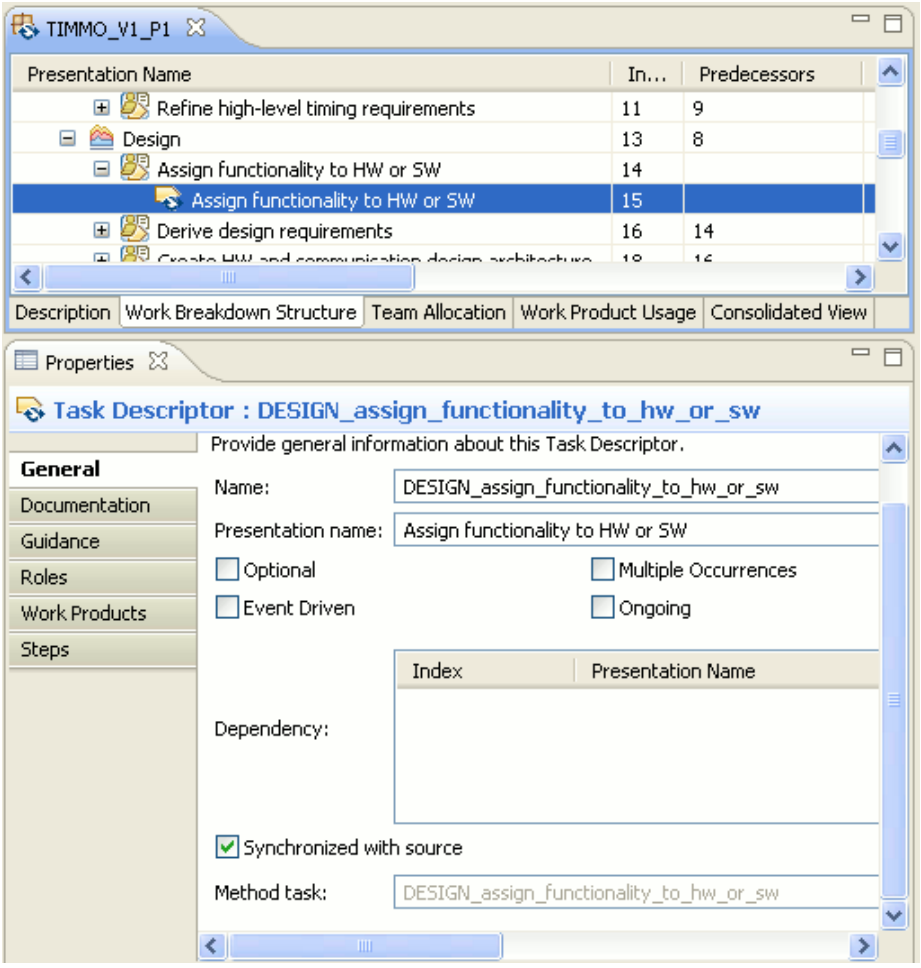


Abbildung 7:TIMMO V1 – Wiederverwendung einer Task Definition[29]

Gleichzeitig mit dem Instanziieren einer Aufgabendefinition werden alle ihre Beziehungen zu Rollen- und Arbeitsproduktdefinitionen als Instanzen erzeugt und eingebunden. Eine komplette Übersicht ist beim Expandieren

[29] Ansicht übernommen aus dem EPF Composer 1.5

der Aufgabeninstanz in der „konsolidierten Sicht“ (Consolidated View) des Prozess-Editors zu finden, wie das Beispiel in Abbildung 8 zeigt. Die Spalten „Type“ und „Model Info“ geben Aufschluss darüber, um welchen Elementtyp[30] es sich handelt und in welcher Beziehung das Element zu dem betrachteten Element steht. Beim Vergleich der Aufgabendefinition in Abbildung 4 auf Seite 26 mit der konsolidierten Liste findet man alle dort zugeordneten Rollen und Arbeitsprodukte hier als Instanzen wieder, z. B. das Arbeitsprodukt „Functional analysis architecture“ als „Mandatory Input“.

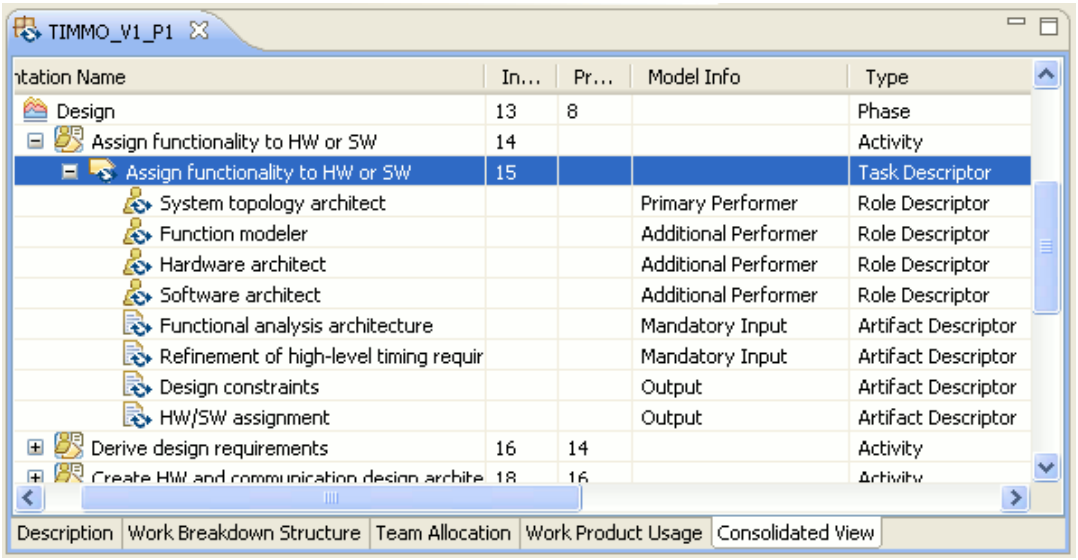


Abbildung 8: TIMMO V1 – Konsolidierte Sicht einer Aufgabeninstanz[31]

Ergebnis des zweiten Arbeitsabschnitts in Phase I ist also die erste Version des TIMMO-Musterprozesses, der durch Einbindung der vorher definierten, TIMMO-spezifischen Methodenelemente zeigt, wie sie in einem idealisierten Entwicklungsprozess, der sich an EAST-ADL2 und grob am V-Modell orientiert, sinnvoll benutzt werden können.

Das Modellierungsergebnis der Phase I besteht insgesamt aus einer EPF Method Library, die das TIMMO Method Plug-in, den Common TIMMO Process und eine Common TIMMO Configuration enthält, und einer aus der Konfiguration publizierten HTML-Darstellung der TIMMO-Methodik, in die ihre Benutzungsanleitung integriert ist. Die Methodenbibliothek wird von Methoden- und Prozessautoren benötigt, die die TIMMO-Methodik übernehmen, ggf. anpassen und in unternehmensspezifische Entwicklungsprozesse integrieren möchten. Die HTML-Darstellung ist für alle Nutzer geeignet, die sich über die TIMMO-Methodik informieren und sie direkt als Vorgehensleitfaden verwenden möchten.

---

30 Diese Instanzen haben im EPF Composer den Zusatz „Descriptor“ in der Typbezeichnung, z. B. „Role Descriptor“ für die Instanz einer Rollendefinition (Role).

31 Ansicht übernommen aus dem EPF Composer 1.5

Phase II – TIMMO-Methodik Version 2

1. Verfeinerung und Ergänzung der TIMMO-Methodeninhalte

Im TIMMO-Projekt fanden zwischen den beiden Entwicklungsphasen eine Überprüfung des Stands der Anforderungserfüllung und ein Ergebnisabgleich zwischen Sprach- und Methodik-Entwicklung statt, um die Arbeiten der zweiten Phase neu auszurichten und zu fokussieren. Die Erkenntnisse daraus wurden im ersten Abschnitt von Phase II genutzt, um die bereits definierten TIMMO-Methodeninhalte zu verfeinern und zu ergänzen, z. B. um Referenzen auf die jeweils relevanten TADL-Konstrukte für Timing-Informationen. Das geschah überwiegend durch Überarbeitung der Attribute der vorhandenen Elemente, z. B. durch Ergänzung der Beschreibungen.

Neu definiert wurden Methodenelemente zur Darstellung von Werkzeugen, die zur Ausführung definierter Aufgaben einsetzbar sind. Da viele Werkzeuge oft für mehrere Zwecke nutzbar sind, wurde jeder Anwendungszweck im EPF Composer getrennt als „Tool Mentor“[32] modelliert. Die verschiedenen Tool-Mentoren eines Werkzeugs wurden logisch zu einer Kategorie zusammengefasst, die das Werkzeug als Ganzes repräsentiert („Tools by name“ in Abbildung 9 auf Seite 32). Wegen ihrer Vielzahl wurden die Tool-Mentoren außerdem in die Funktionsgruppen Modellierung, Analyse, Simulation und Tracing eingeordnet („Tools by function“). Abbildung 9 zeigt z. B. die Kategorie der Simulations-Tools in der HTML-Darstellung. Die Namen der einzelnen Tool-Mentoren unter den Simulations-Tools sind im Original Hyperlinks auf die hinterlegten Definitionen. Im Bild auf der linken Seite ist die Navigationsstruktur innerhalb der Guidance-Elemente der TIMMO-Methodik erkennbar.

Im zweiten Schritt wurde für jede TIMMO-Aufgabendefinition geprüft, welche Werkzeuge aus dem Vorrat der Tool-Mentoren zu ihrer Ausführung benutzt werden können; zu den geeigneten wurden Querbeziehungen angelegt. Diese Information findet sich nun auch in der HTML-Darstellung der Aufgabendefinition „Assign functionality to HW or SW“, wie in Abbildung 10 auf Seite 32 im Abschnitt „More Information“ unter „Tool Mentors“ zu sehen ist.

[32] Im SPEM ist dies ein Typ des allgemeinen „Guidance“-Elements (vgl. SPEM-Grundelemente in Kapitel 3.2.4.3).

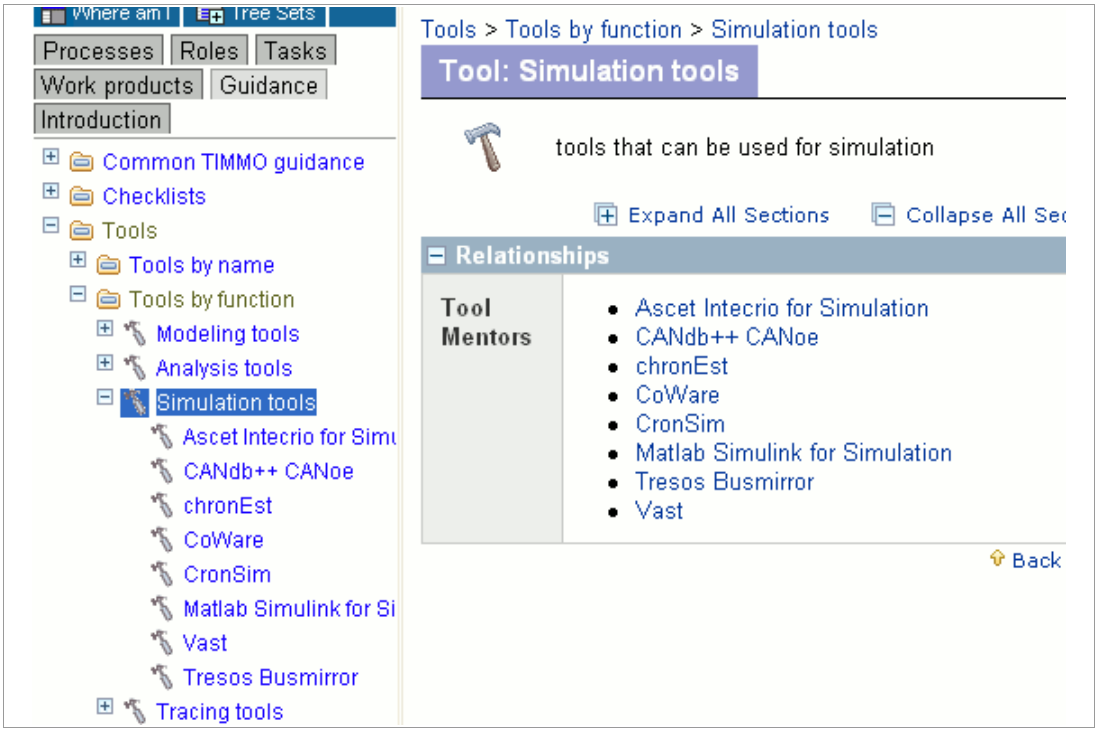


Abbildung 9: TIMMO V2 – Übersicht über Tools[33]

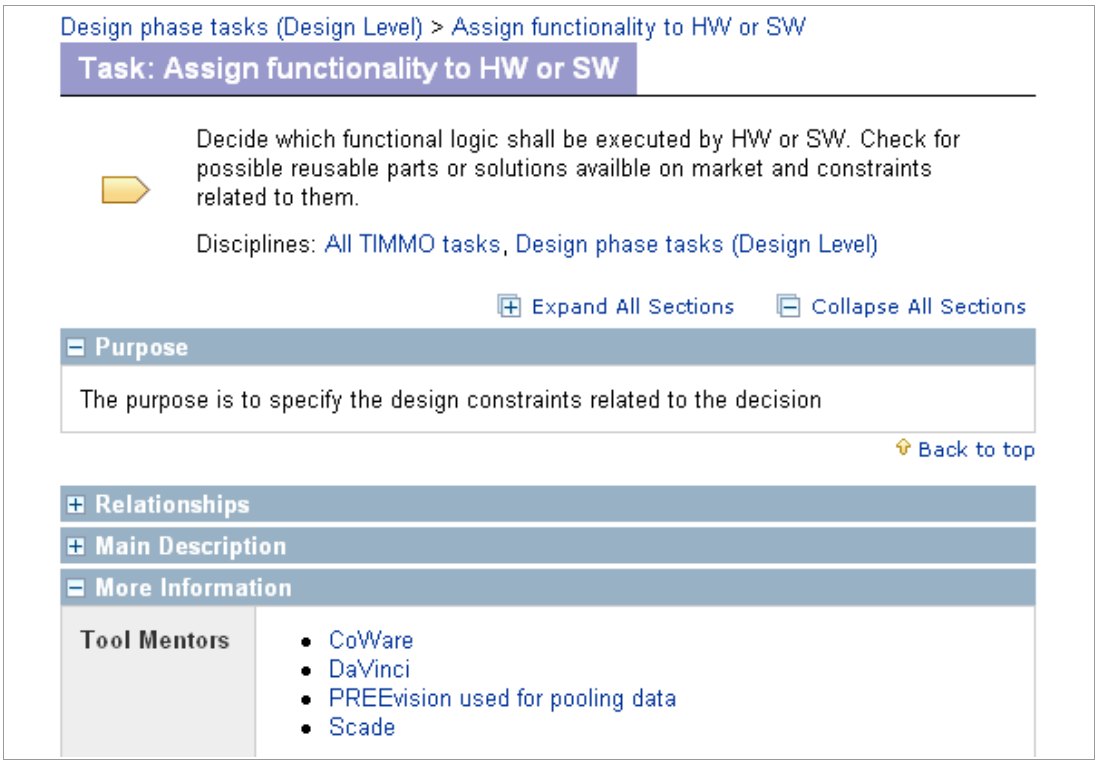


Abbildung 10: TIMMO V2 – Task Definition mit Tools[34]

[33] HTML-Darstellung der im EPF Composer 1.5 erstellten Tool-Definitionen
[34] HTML-Darstellung der im EPF Composer 1.5 erstellten Task Definition

Neben und nach der Überarbeitung der TIMMO-Methodeninhalte war ebenso erforderlich eine ...

2. Aktualisierung des TIMMO-Musterprozesses

Da Änderungen an Methodeninhalten nicht automatisch in Prozesse übernommen werden, die die Methodeninhalte benutzen, war nach der Überarbeitung eine kontrollierte Synchronisation zwischen den Methodenelementen und dem TIMMO-Musterprozess notwendig. Dieser Vorgang wird durch Funktionen des Prozess-Editors unterstützt, die dem Autor verschiedene Ausführungsoptionen bieten.

Darüber hinaus gab es im TIMMO-Prozess auch im Workflow Änderungen und Verfeinerungen. Es wurde z. B. am Ende jeder Phase ein Meilenstein eingefügt, um darzustellen, dass bestimmte Übergabekriterien zu erfüllen sind, damit eine Phase als einmal erfolgreich abgeschlossen betrachtet werden kann. Die jeweiligen Kriterien wurden mittels Checklisten erfasst, die als Guidance-Elemente neu definiert wurden. Gleichzeitig wurde in den Meilensteinen festgelegt, welche Arbeitsprodukte als Phasenergebnisse gefordert sind.

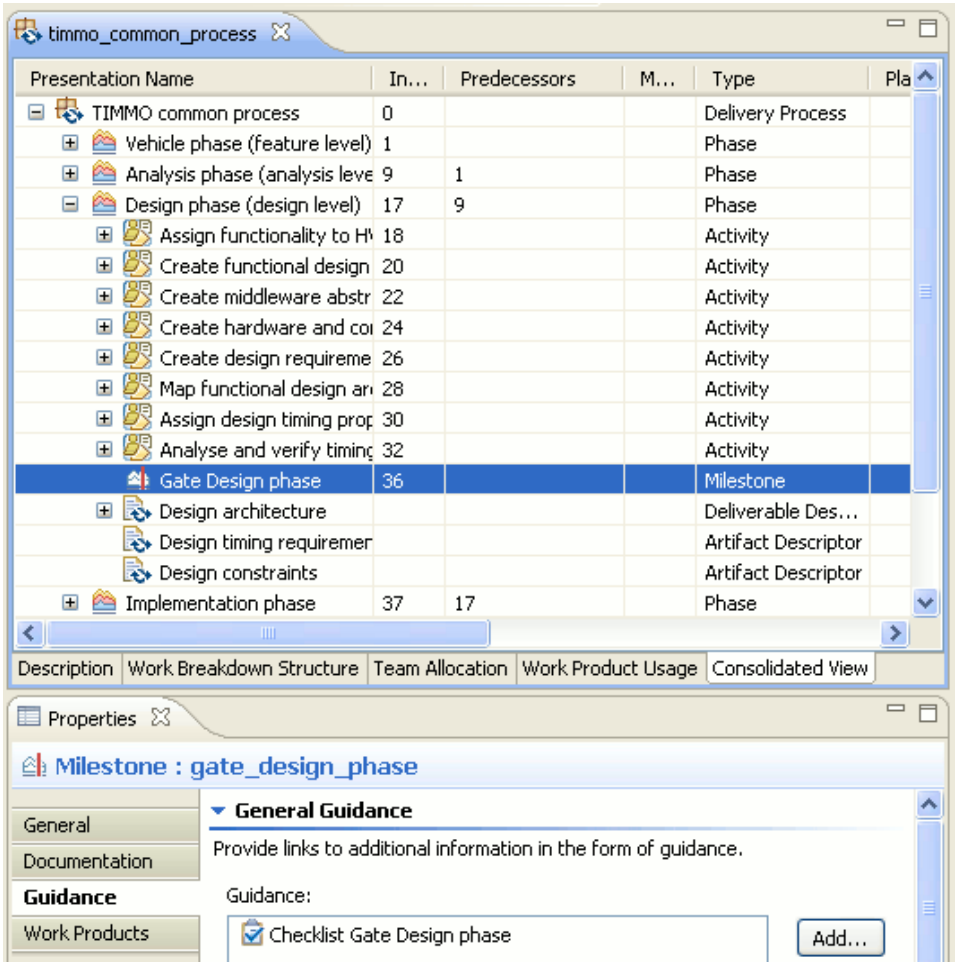


Abbildung 11: TIMMO V2 – Prozessstruktur mit Meilenstein[35]

[35] Ansicht übernommen aus dem EPF Composer 1.5

In Abbildung 11 auf Seite 33 ist ein Beispiel der konsolidierten Sicht der neuen Prozessstruktur zu sehen. Der Meilenstein „Gate Design phase“ steht am Übergang zwischen den Phasen „Design“ und „Implementation“. Vorzuliegen haben die Arbeitsprodukte „Design architecture“, „Design timing requirements“ und „Design constraints“.

Im Attribut-Editor im unteren Teil des Bilds ist die Rubrik „Guidance“ aufgeblättert, in der dem Meilenstein das Guidance-Element „Checklist Gate Design phase“ zugeordnet wurde.

Vergleicht man die neue Struktur der Design-Phase in Abbildung 11 mit der initialen in Abbildung 5 auf Seite 27, erkennt man, dass auch die Aktivitäten der Phase überarbeitet und ergänzt wurden. Nach Abschluss aller Anpassungen und ihrer Überprüfung in Reviews lagen die Modellierungsergebnisse (Methodenbibliothek und HTML-Darstellung, wie schon für Phase I erläutert) in der für das Projekt TIMMO endgültigen Version 2 vor.

Während das Praxisbeispiel „TIMMO-Methodik“ weitgehend aus Sicht der Modellierung beschrieben wurde, liegt der Schwerpunkt des folgenden Praxisbeispiels auf der Perspektive der Nutzung.

4.2 Systementwicklungsmethodik „SEM New Generation“

4.2.1 Anwendungskontext

Die Systementwicklungsmethodik SEM® ist als Vorgehensmodell für Software- und Systementwicklungsprojekte in einem Softwareentwicklungsbereich der Siemens AG in Österreich entstanden und wird heute innerhalb von Siemens IT Solutions and Services im Bereich „Software Development and Engineering“ angewandt. Das allgemeine Vorgehensmodell, das die Phasen „Initiierung“, „Durchführung“ und „Abschluss“ verbindlich vorschreibt, wird durch mehrere Ausprägungen der Phase „Durchführung“ konkretisiert. Die Ausprägungen spezialisieren sich in Hinblick auf den Anwendungsbereich und die Entwicklungsmethodik im engeren Sinne:

stdSEM®	allgemeine Methodik zur Entwicklung von Software, inklusive objektorientierter Entwicklung
e-SEM®	spezialisiertes Modell zur iterativ-inkrementellen Entwicklung von Lösungen im e-Business Umfeld
prodSEM®	Methodik zur Entwicklung von Produkten (HW+SW) aus Sicht des Produktmanagements
hsSEM®	Methodik zur Entwicklung von Elektronik, ASICs und Software

Eine bestimmte SEM-Ausprägung ist als Vorgehensrichtlinie für konkrete Projekte dieser Ausprägung konzipiert. Im Wesentlichen wird die Abfolge von Phasen mit ihren Aktivitäten vorgegeben sowie, welche Voraussetzungen zum Eintritt in Phasen bzw. zum Beginn von Aktivitäten erfüllt sein müssen und welche Ergebnisse zu erstellen sind. Dies wird ergänzt durch Methodenbeschreibungen und Praxis-Tipps für die Ausführung von Aktivitäten sowie durch Checklisten und Vorlagen für die geforderten Ergebnisse. Neben den phasenbezogenen, technischen Aktivitäten zur Erzeugung der Projektergebnisse im engeren Sinne berücksichtigt SEM auch phasenübergreifend projektsteuernde und qualitätssichernde Aktivitäten.

Das grundlegende SEM Vorgehensmodell (SEM-VM) ist in einem Textdokument beschrieben und nicht als Handbuch für die Projektarbeit bestimmt, denn diese Funktion erfüllen die davon abgeleiteten konkreteren Ausprägungen. Sie wurden in der Vergangenheit bereits auf Webseiten eines SEM-Portals [14] im Siemens-Intranet zugänglich gemacht und stehen so für die tägliche Arbeit online als Referenz zur Verfügung. Das SEM-VM erfüllt also den Zweck eines Prozessrahmens und im Sinne der Begriffsbestimmung in Kapitel 2.2 den eines Metamodells, während die Ausprägungen Prozessmodelle darstellen.

Im Jahr 2009 wurden Pilotversionen zweier SEM-Ausprägungen neu bereitgestellt, „agileSEM“ für die Abwicklung von Entwicklungsprojekten unter Anwendung agiler Prinzipien und „iiSEM“ für ein iteratives inkrementelles Vorgehen. Sie sind die ersten Repräsentanten einer „SEM New Generation“ (SEM NG), die mit dem Eclipse Process Framework modelliert und dokumentiert wurden. Dabei wurde inhaltlich auf eingeführte, bewährte SEM-

Ausprägungen zurückgegriffen, hauptsächlich auf „stdSEM“, die überarbeitet und neu strukturiert wurde. Das für die Anwender sichtbare Ergebnis ist z. Zt. die Pilotversion des Portals für SEM NG im Siemens-Intranet [15]. Abbildung 12 zeigt eine graphische Übersicht der Einbettung von SEM und ihren Ausprägungen in den Prozesskontext der Organisationseinheit, der wiederum von Unternehmensvorgaben und -richtlinien abgeleitet ist.

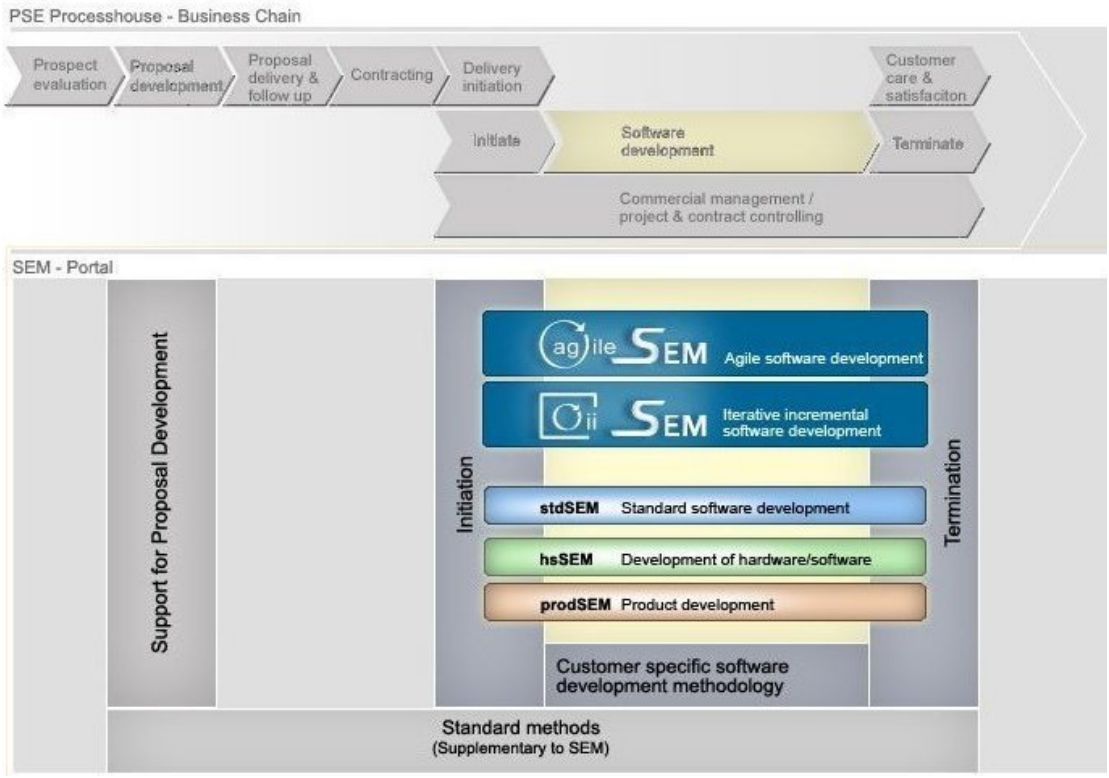


Abbildung 12: SEM NG Portal – Überblick (Quelle: [15])[36]

Unter den von SEM-Verantwortlichen als Vorzüge der neuen Ausprägungen genannten Eigenschaften können folgende auf die neue Modellierungsbasis zurückgeführt werden:

- rollenorientierte Sichten; dadurch klarere Zuständigkeiten und Verantwortlichkeiten
- flexibel kombinierbare „Iteration Patterns“ für bessere Abbildung von Projektspezifika
- Unterstützung von domänen- und organisationsspezifischem Customizing; dadurch u. a. erleichterte Kopplung mit vom Auftraggeber vorgegebenen Methoden

Die Präsentation von „iiSEM“[37] im Portal SEM NG soll hier als Praxisbeispiel für die Nutzung eines mit dem EPF erstellten Prozessmodells dienen.

---

[36] © Siemens AG 2009. Die Verwendung dieser und aller weiteren Abbildungen aus SEM NG erfolgt mit Zustimmung von Siemens IT Solutions & Services SDE QM.

[37] iiSEM Version 1.0-001 Copyright © Siemens AG Österreich 1997-2009. All rights reserved. Published on: 2009-06-29 09:18:55

4.2.2 Präsentation und Nutzung

Nutzer von SEM NG sind Projektverantwortliche und Entwickler in einer Organisationseinheit, die Software- und Systementwicklungsprojekte durchführt. Für Projekte, deren Durchführungsphase iterativ und inkrementell gestaltet werden soll, wenden sie die SEM-Ausprägung „iiSEM“ an. Die im Intranet zugängliche Website zu „iiSEM“ stellt einen Vorgehensleitfaden dar, der durch Informationen und Verweise die praktische Ausführung unterstützt, die unabhängig davon in der jeweiligen Projektarbeitsumgebung manuell erfolgt. „iiSEM“ wird also von Nutzern unterschiedlicher Projektrollen mit individuell verschiedenen Erfahrungshorizonten hinsichtlich der Methodik angewandt. Dem trägt die Website „iiSEM“ Rechnung, deren Aufbaustruktur und Inhalte per Konfiguration im EPF Composer gestaltet und automatisch publiziert werden (vgl. Kapitel 3.2.5.4).

Zum besseren Verständnis der nachfolgenden Beispiele ist in Abbildung 13 der vorgegebene Seitenaufbau EPF-generierter Webseiten schematisch dargestellt.

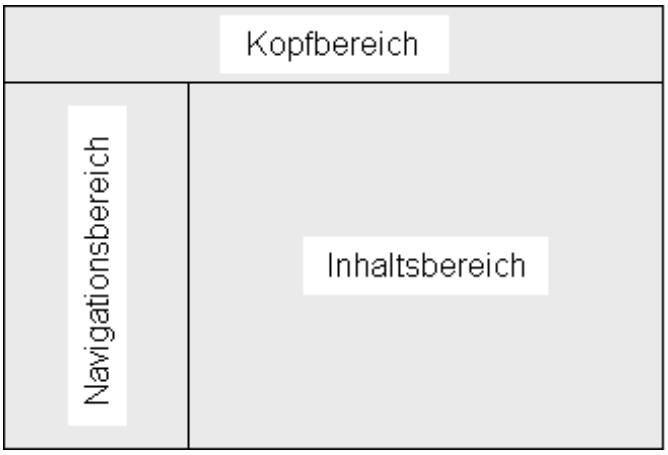


Abbildung 13: Allgemeiner Aufbau EPF-generierter Webseiten

Kopfbereich: ist für die gesamte Website identisch, dient zur Identifikation der Website und bietet allgemeine Funktionen wie z. B. Glossar oder Feedback-Link.

Navigationsbereich: repräsentiert die inhaltliche Struktur der Website, die der Autor durch die Sichten bestimmt, die er in der Konfiguration definiert. Die hierarchische Kategorienstruktur einer Sicht wird in Form eines expandierbaren Baumes, vergleichbar mit einem Dateisystembrowser, dargestellt.

Inhaltsbereich: zeigt das im Navigationsbereich ausgewählte Element in seiner durch den EPF Composer elementspezifisch vorgegebenen Darstellung, die die Attribute und Beziehungen des Elements präsentiert. Beziehungen werden mit Hyperlinks auf die in Beziehung stehenden Elemen-

te hinterlegt, die eine zweite Form der Navigation zusätzlich zum Strukturbrowser im Navigationsbereich ermöglichen.

Die vorgegebene Seitenstruktur ist in der Startseite von „iiSEM“ leicht wiederzuerkennen, wie Abbildung 14 zeigt.

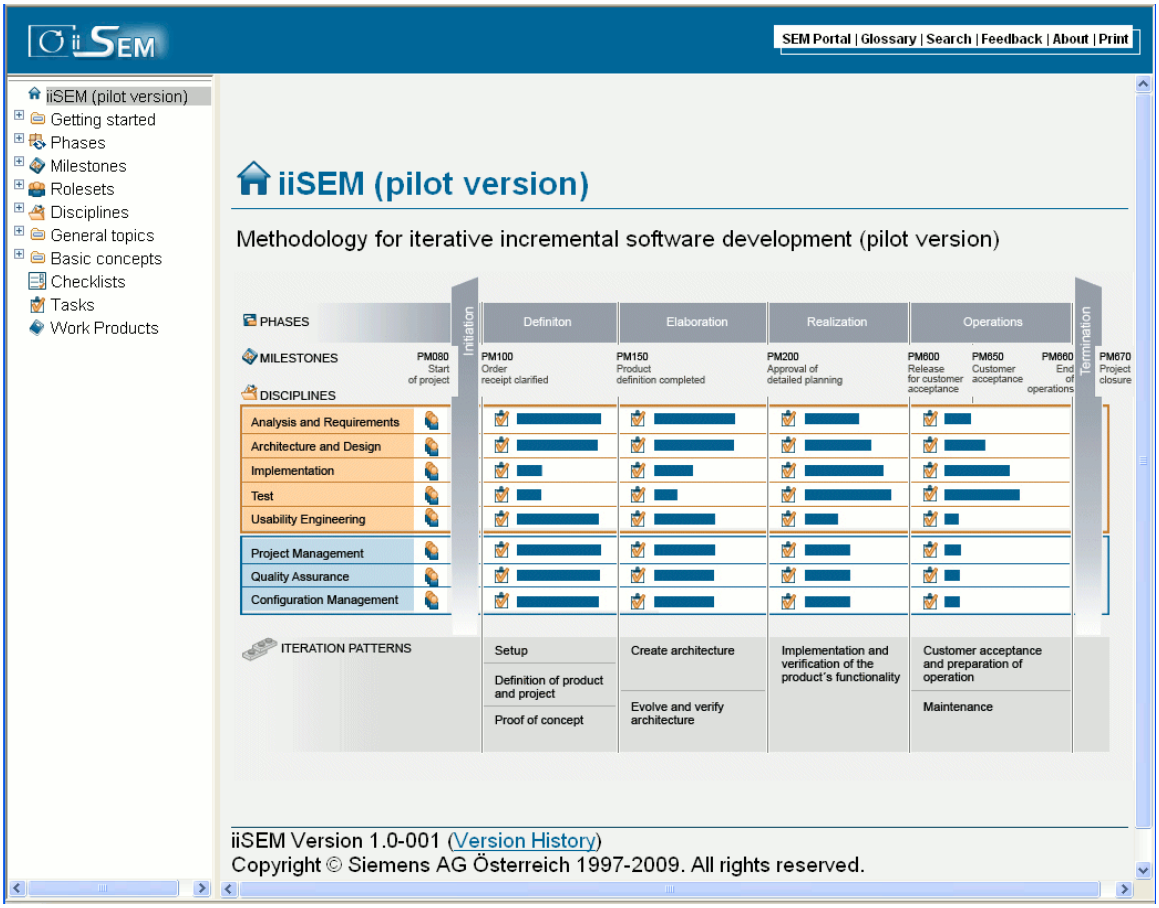


Abbildung 14: iiSEM – Startseite

Die „iiSEM“ Website bietet eine einheitliche Sicht, die alle Inhalte innerhalb einer Baumstruktur abbildet. Die oberste Ebene der Struktur im Navigationsbereich ermöglicht den passenden Einstieg je nach Hintergrund und Intention des Nutzers (siehe Details in Abbildung 15). Daneben ist auch der Einstieg über verlinkte Elemente im Inhaltsbereich möglich. Es zeigt sich hier, wie außer den Prozessmodellinhalten auf einfache und konsistente Art Hinweise für den neuen oder gelegentlichen Nutzer in die Website integriert werden können. Der Zweig „Getting started“ bietet einführende Informationen und eine detaillierte Benutzungsanleitung („Directions for use“).

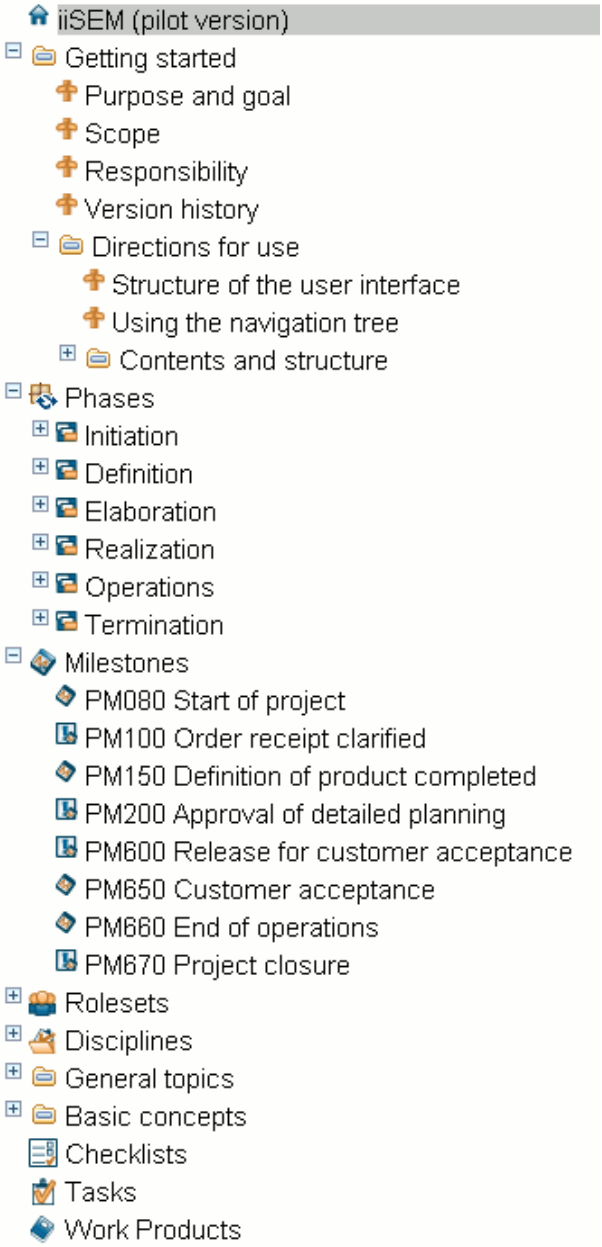


Abbildung 15: iiSEM – Navigationsstruktur, teilweise expandiert

Die übrigen Elemente eröffnen unterschiedliche inhaltliche Zugänge zum Prozessmodell. Die Zweige für Phasen, Meilensteine, Rollen, Disziplinen, Checklisten, Aufgaben und Arbeitsprodukte entsprechen den zuvor bereits erläuterten Grundelementen für Methoden- und Prozessinhalte (vgl. Kapitel 3.2.4.3). Wer, wie z. B. ein potenzieller Projektleiter, einen Einblick in den Prozessablauf gewinnen möchte, wird vermutlich am besten über den Zweig „Phases“ einsteigen. Ein Qualitätsmanager wird sich z. B. über die vorgegebenen Meilensteine im Zweig „Milestones“ informieren wollen. Die Zweige „General topics“ und „Basic concepts“ dienen dazu, phasen- oder disziplinenübergreifende Themen und Grundkonzepte des Prozessmodells darzustellen.

Ein wesentliches Grundkonzept ist das „Tailoring“, d. h. das Auswählen und Zuschneiden der passenden Prozessausprägung auf ein konkretes Projekt. Es wird ausführlich beschrieben, welche Wahl- und Anpassungsmöglichkeiten bestehen. Die getroffenen Entscheidungen sind laut Beschreibung im Qualitätssicherungsplan des Projekts zu dokumentieren. Ein Tailoring des EPF-Modells mittels Variabilität von Inhalten, wie in Kapitel 3.2.5.3 beschrieben, ist im Regelfall nicht vorgesehen, könnte aber für sehr große, lang laufende Projekte mit vielen Beteiligten oder für wiederkehrende Projekttypen sinnvoll und mit Unterstützung der Methodik-Autoren auch möglich sein. Es ließe sich dann auch eine angepasste HTML-Repräsentation für die jeweilige Zielgruppe publizieren. Deren Nutzung ist vermutlich wirksamer und leichter durchsetzbar als das Nachhalten von Abweichungen und Zusatzvereinbarungen, die in einem Projektdokument festgehalten sind.

Als konkretes Beispiel für einen rollenorientierten Zugang zum Prozessmodell sei hier die Rolle „User Interface Designer“ ausgewählt. In der Navigation findet man sie in der Rollenfamilie „System Development Roles“. Per Klick auf den Rollennamen wird im Inhaltsbereich die umfassende Darstellung der Rolle gezeigt, die aus den Rubriken „Description“ (siehe Abbildung 16), „Tasks“, „Work Products“ und „Guidance“ besteht.

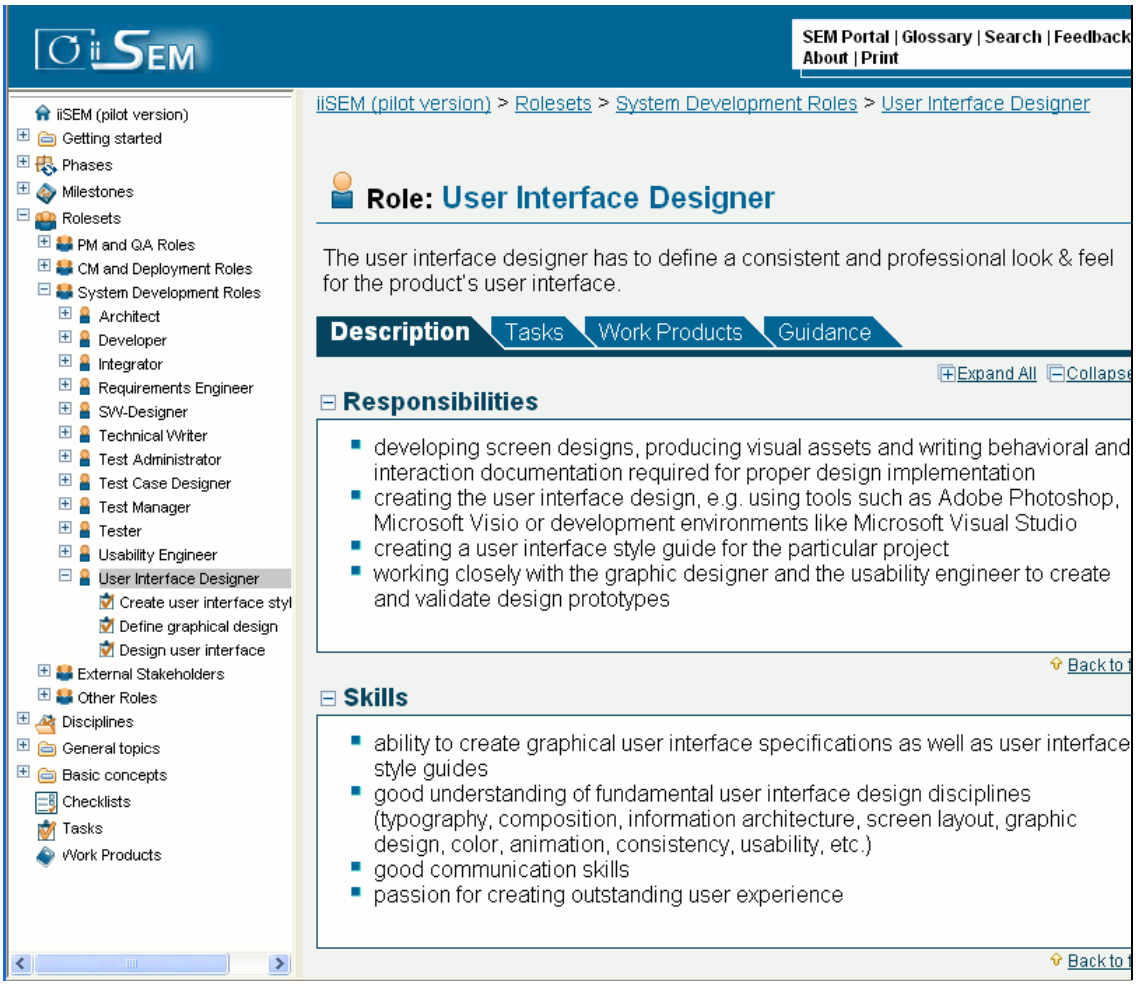


Abbildung 16: iiSEM – Beschreibung der Rolle „User Interface Designer“

Die Darstellung der Aufgaben („Tasks“) ist nach ihrer Zuordnung zu Phasen gegliedert und unterscheidet, ob die Rolle verantwortlich („responsible for“) für die Aufgabe oder nur daran beteiligt ist („involved in“). Die benutzten bzw. produzierten Arbeitsprodukte der Aufgabe werden ebenfalls aufgeführt. In Abbildung 17 ist im Inhaltsbereich nur die Aufgabe der Rolle „User Interface Designer“ aufgeblättert, für die sie in der Phase „Realization“ verantwortlich ist, nämlich „Design user interface“ mit den Input-Produkten „User interface style guide“, „User interface concept“, „Software requirements specification“ und „Naming conventions“ und dem Ergebnis „User interface design“. Im Navigationsbereich ist gleichzeitig zu erkennen, dass die Rolle an insgesamt drei Aufgaben beteiligt ist.

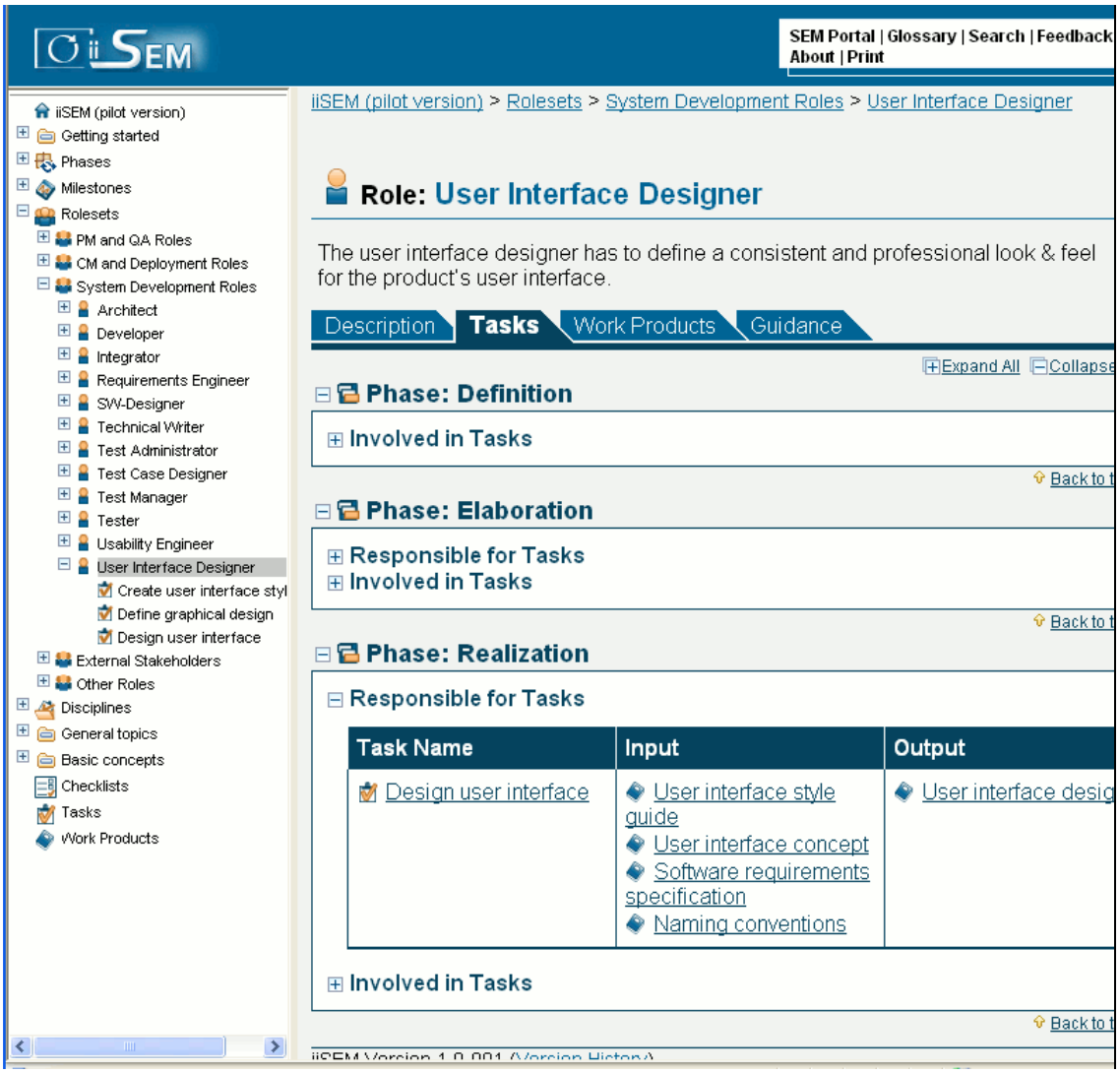


Abbildung 17: iiSEM – Aufgaben der Rolle „User Interface Designer“

Je nach Informationsbedarf können weitere Aufgaben der Rolle untersucht werden, oder es lassen sich durch Nutzung der Hyperlinks im Inhaltsbereich mehr Details über Aufgaben und Arbeitsprodukte erfahren. Betrachtet ein UI-Designer die Aufgabenbeschreibung „Design user interface“, so findet er darin auch Informationen über ihre Einbettung in den Prozess, die ihm eine überge-

ordnete Orientierung ermöglichen. Hier sind dies die Zugehörigkeit der Aufgabe zur Disziplin „Architecture and Design“, ihre Einordnung sowohl in Phase „Elaboration“ als auch in Phase „Realization“ sowie eine Liste von weiteren Rollen, die an der Ausführung beteiligt sind, nämlich „Developer“, „SW Designer“ und „Usability Engineer“ (siehe Abbildung 18).

Falls die Granularität der Rollendefinitionen als zu fein für kleinere Projekte erscheint, ist anzumerken, dass ihr Zweck in der möglichst klaren Definition von Verantwortlichkeiten und erforderlicher Qualifikation besteht. Im konkreten Projekt müssen verschiedene Rollen keineswegs mit verschiedenen Personen besetzt werden. Wenn es solche Bedingungen im Prozessmodell gibt, dann sind sie explizit zu beschreiben, z. B. wenn ausgeschlossen werden soll, dass in einem Projekt die Rollen Projektleiter und Qualitätsmanager von derselben Person übernommen werden.

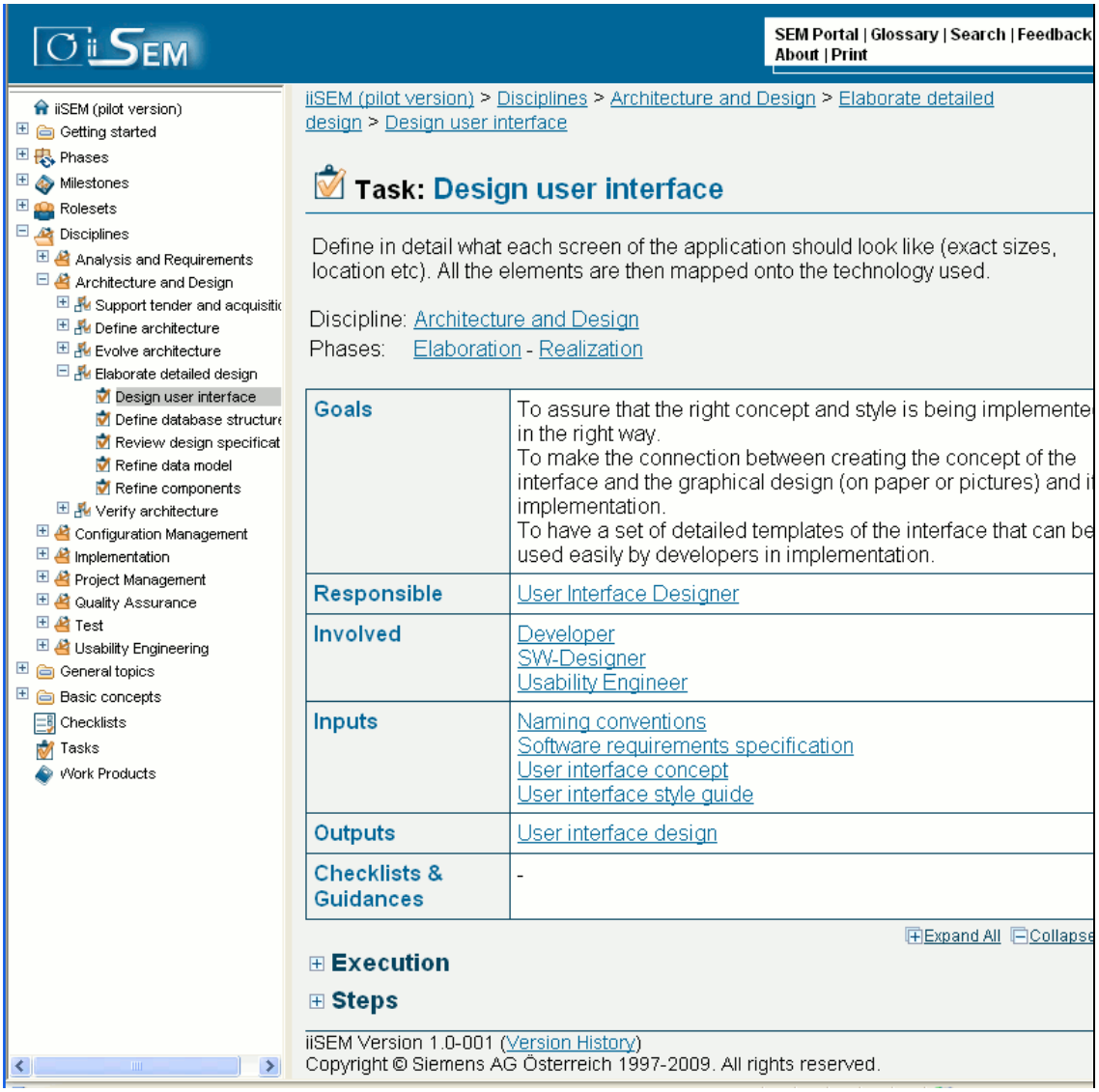


Abbildung 18: iiSEM – Aufgabe „Design user interface“ (1 v. 2)

In der folgenden Abbildung 19 sind die Abschnitte „Execution“ und „Steps“ der Aufgabendefinition „Design user interface“ expandiert. Sie dienen dazu, knappe, direkte Hinweise zur Ausführung der Aufgabe zu geben. Wenn dafür umfangreiche Beschreibungen nötig sind, die eventuell bereits in anderer Form vorliegen, dann sollte der Methodik-Autor über Guidance-Elemente darauf verweisen, um unnötige Redundanzen zu vermeiden. Die Definition von Arbeitsschritten („Steps“) ist für den Methodik-Autor besonders im Hinblick auf Wiederverwendung und Tailoring interessant, weil die Schritte eigenständige Methodenelemente einer Aufgabendefinition sind, die über das Konzept der Variabilität (vgl. Kapitel 3.2.5.3) im EPF-Modell anpassbar sind.

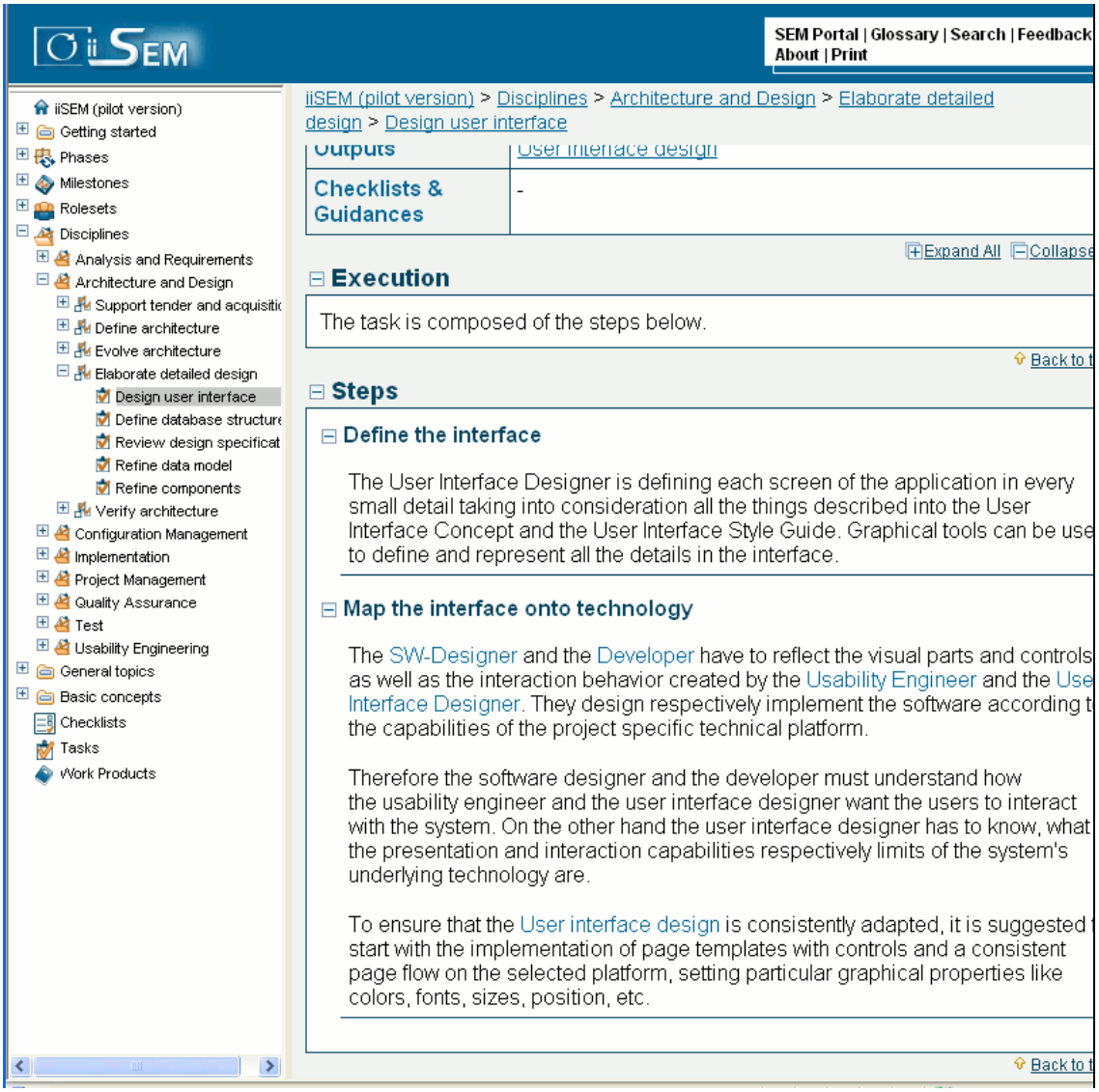


Abbildung 19: iiSEM – Aufgabe „Design user interface“ (2 v. 2)

Mit diesem Einblick in die Rolle des „User Interface Designers“ sei die Vorstellung des Praxisbeispiels „Systementwicklungsmethodik SEM New Generation“ abgeschlossen.

5 Zusammenfassende Bewertung

Prozessorientierung als Grundlage für die Durchführung von Entwicklungsprojekten ist heute eine notwendige, wenn auch nicht hinreichende Voraussetzung für Wettbewerbsfähigkeit. Kein Unternehmen kann darauf verzichten, der Unterschied liegt darin, wie wirksam und erfolgreich dieser Anspruch verfolgt und vor allem praktisch umgesetzt wird. Dieser Bericht hat einen Ansatz der Prozessmodellierung motiviert, vorgestellt und anhand von Beispielen illustriert, nämlich das auf dem Standard SPEM basierende Eclipse Process Framework. Es soll nicht der Eindruck erweckt werden, das sei der einzige oder der optimale Weg. Es ist aber ein Weg, der entscheidende Vorteile gegenüber dem bisher häufig eingeschlagenen hat, auf dem wortreich nicht standardisierte Prozessbeschreibungen in Dokumenten niedergelegt werden, die aber nie konsequent Eingang in die alltägliche Praxis finden, und deren Pflege immense Aufwände verschlingt, sodass die für einen kontinuierlichen Verbesserungsprozess so wichtigen Anpassungen und Aktualisierungen eher unterbleiben.

Mit dem kostenfrei verfügbaren EPF Composer ist es möglich, Prozessmodelle im Sinne von Vorgehensmodellen und Methodiken vollständig, konsistent, modular, anpassbar und änderungsfreundlich zu definieren, dabei auch existierende Ressourcen außerhalb des Modells einzubinden und eine stets aktuelle, zielgruppenorientierte Online-Dokumentation automatisch zu generieren. Modellierungs- und Tool-Expertise wird dabei nur von Prozess-Autoren in überschaubarem Umfang verlangt.

Für die Anwender lässt sich die Prozessdokumentation nahtlos in heute gängige Intranet-Umgebungen einbinden und über Hyperlinks mit anderen Inhalten verknüpfen. Diese Lösung ist hervorragend geeignet für organisationsübergreifende, unternehmensweite oder bereichsspezifische Prozessvorgaben, die wegen der hohen Individualität der praktischen Projektkontexte die operative Umsetzung nicht bis ins Detail festschreiben wollen oder können. Die gezeigten Praxisbeispiele fallen in diesen Anwendungsbereich.

Der EPF-Einsatz stößt heute an seine Grenzen, wo eine werkzeuggestützte Überleitung in die operative Projektausführung angestrebt wird. Hier findet der Medienbruch statt. Es kann dem Projektleiter anhand der Online-Dokumentation noch so komfortabel und umfassend illustriert werden, wie ein Projekt zu gestalten ist und welche Tailoring-Optionen es gibt, den Projektplan muss er weiterhin mit oder ohne Tool-Unterstützung selbst erstellen. EPF Composer bietet zwar eine Funktion, um modellierte Prozesse als Template für ein verbreitetes Projektplanungswerkzeug zu exportieren. Will man diesen Weg verfolgen, kommt ein Projektleiter aber nicht mehr mit der generierten Prozessdokumentation allein aus, sondern muss das Modellierungstool beherrschen. Hier sind benutzerfreundlichere Ansätze denkbar, die neben dem EPF Composer in das Eclipse Process Framework integriert werden könnten, vielleicht eine Projektmanagement-Engine, die vor allem bei der Projektdurchführung alle Prozesselemente auch tatsächlich mit berücksichtigt. Diese Engine sollte dann wiederum offene Schnittstellen zu weiteren Komponenten

wie Projekt-Repositories, Versionsmanagement-Systemen und Software-Entwicklungsumgebungen besitzen.

Wer in diese Richtung heute schon weiter voranschreiten möchte und in seinem Unternehmen geeignete Rahmenbedingungen findet, kann durchaus auf proprietäre, kommerzielle Umgebungen zurückgreifen. Dazu sind aber unbedingt eine gründliche, kompetente Anforderungsanalyse und eine darauf aufbauende Marktstudie notwendig.

6 Literaturverzeichnis und Verweise

[1] ATESST Consortium, *Specification of the EAST-ADL2* (ATESST Deliverable D3.1 Part II), Februar 2007, siehe Projekt-Website atesst.org

[2] Automotive Open System Architecture (AUTOSAR), siehe Website autosar.org

[3] Botella, P.; Franch, X.; Ribó, J.M. (2004). *Software Process Modelling Languages Based on UML*. UPGRADE , V (5) : 34-39. ISSN: 1684-5285

[4] Deming, W. E. *Out of the Crisis*. 2. Aufl., Cambridge/ Mass. / USA: Massachussetts Institute of Technology Press, 1986

[5] Gnatz, M.; Deubler, M.; Meisinger, M.; Rausch, A. *Towards an Integration of Process Modeling and Project Planning*; in Proceedings of the 5th International Workshop on Software Process Simulation and Modeling (ProSim 2004). ICSE 2004, 2004

[6] Haumer, P. *Eclipse Process Framework Composer, Part 1: Key Concepts*. Second Revision, April 2007, eclipse.org/epf/general/EPFComposerOverviewPart1.pdf, abgerufen am 7.9.2009

[7] Haumer, P. *Eclipse Process Framework Composer, Part 2: Authoring method content and processes*. Second Revision, April 2007, eclipse.org/epf/general/EPFComposerOverviewPart2.pdf, abgerufen am 7.9.2009

[8] Humphrey, W. S. *Managing the Software Process*, SEI Series in Software Engineering, Addison Wesley, 1989.

[9] Imai, M. *Kaizen: Der Schlüssel zum Erfolg der Japaner im Wettbewerb*. Ullstein, Frankfurt/M. 1986.

[10] Object Management Group, *Catalog of OMG Business Strategy, Business Rules and Business Process Management Specifications*, siehe omg.org/technology/documents/br_pm_spec_catalog.htm

[11] Object Management Group, *Software & Systems Process Engineering Meta-Model Specification*, Version 2.0, OMG Document Number: formal/2008-04-01, Standard document URL: omg.org/spec/SPEM/2.0/PDF

[12] Osterweil, L. *Software Processes are Software Too*. In Proceedings of the International Conference on Software Engineering (ICSE-9), 1987.

[13] Schmelzer, H. J.; Sesselmann, W. *Geschäftsprozessmanagement in der Praxis. Kunden zufrieden stellen, Produktivität steigern, Wert erhöhen.* Hanser Fachbuch, 6. Auflage, 2007

[14] Siemens AG, *SEM Portalseite* (Intranet), sem.siemens.at/sem/default.htm, abgerufen am 6.10.2009

[15] Siemens AG, *SIS SDE Methodology Portal* (Intranet), editsem.gud.siemens.at/semng/default.htm, Pilotversion, abgerufen am 6.10.2009

[16] TIMMO Consortium, *TIMMO Timing Model – Methodology Version 2* (TIMMO Deliverable D7), August 2009

[17] Verbundprojekt „TIMMO – Timing Model“ (ITEA 2 project 06005), April 2007 bis Dezember 2009, Projekt-Website www.timmo.org

[18] Das V-Modell XT, Bundesrepublik Deutschland. Online verfügbar unter: v-modell-xt.de