Herausgegeben von Norbert Szyperski, Udo Winand, Dietrich Seibt, Rainer Kuhlen, Rudolf Pospischil und Claudia Löbbecke

Martin Engelien/Detlef Neumann (Hrsg.)

Virtuelle Organisation und Neue Medien 2000

Workshop GeNeMe2000 Gemeinschaften in Neuen Medien

TU Dresden, 5. und 6. Oktober 2000

JOSEF EUL VERLAG
Lohmar · Köln

Reihe: Telekommunikation @ Mediendienste · Band 10

Herausgegeben von Prof. Dr. Dr. h. c. Norbert Szyperski, Köln, Prof. Dr. Udo Winand, Kassel, Prof. Dr. Dietrich Seibt, Köln, Prof. Dr. Rainer Kuhlen, Konstanz, Dr. Rudolf Pospischil, Brüssel, und Prof. Dr. Claudia Löbbecke, Köln

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inf. Detlef Neumann (Hrsg.)

Virtuelle Organisation und Neue Medien 2000

Workshop GeNeMe2000
Gemeinschaften in Neuen Medien

TU Dresden, 5. und 6. Oktober 2000

JOSEF EUL VERLAG
Lohmar · Köln

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

GeNeMe <2000 Dresden>:
GeNeMe 2000 : Gemeinschaften in neuen Medien ; Dresden, 5. und 6. Oktober 2000, an der Fakultät Informatik an der Technischen Universität Dresden / Technische Universität Dresden, Fakultät Informatik, Institut für Angewandte Informatik, Privat-Dozentur „Angewandte Informatik". Martin Engelien ; Detlef Neumann (Hrsg.). – Lohmar ; Köln : Eul, 2000
(Reihe: Telekommunikation und Mediendienste ; Bd. 10)
ISBN 3-89012-786-X

Josef Eul Verlag GmbH
Brandsberg 6
53797 Lohmar
Tel.: 0 22 05 / 91 08 91
Fax: 0 22 05 / 91 08 92
http://www.eul-verlag.de
info@eul-verlag.de

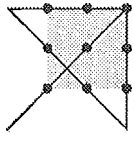
Printed in Germany
Druck: Rosch-Buch, Scheßlitz

Technische Universität Dresden
Fakultät Informatik • Institut für Angewandte Informatik
Privat-Dozentur „Angewandte Informatik“

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inf. Detlef Neumann
(Hrsg.)

an der
Fakultät Informatik der Technischen Universität Dresden

gefördert von der Klaus Tschira Stiftung,
gemeinnützige Gesellschaft mit beschränkter Haftung,
unter Mitwirkung der Gesellschaft für Informatik e.V., Regionalgruppe Dresden



am 5. und 6. Oktober 2000
in Dresden

http://www-emw.inf.tu-dresden.de/geneme
Kontakt: Detlef Neumann (dn3@inf.tu-dresden.de)

E.2. Modellierung gruppenorientierter Software-Entwicklungsprozesse mit Notes/Domino

Prof. Dr. R. Liskowsky
R. Pjater
Fakultät Informatik der Technischen Universität Dresden

1. Einleitung

Mit der ständigen Zunahme der Entwicklung und Produktion von Software, die sich gegenwärtig in einer fehlenden Zahl von Spezialisten ausdrückt, wächst der Zwang, die damit verbundenen Prozesse weiter zu rationalisieren. Diese unbedingte Notwendigkeit bezieht sich nicht nur auf die Verbesserung der technischen Hilfsmittel, also der Werkzeuge, sondern auch auf den Prozess der Aufteilung von Entwicklungsaufgaben innerhalb eines Prozessteams. Darin eingeschlossen ist der Aufbau notwendiger Kommunikations- und Koordinationsbeziehungen als Grundlage für das optimale Zusammenwirken der Teammitglieder in einer verteilten Community.

Das Projektteam selbst bildet eine virtuelle Gemeinschaft, d.h. in einer räumlichen und zeitlichen Aufteilung an verschiedenen Orten und in der Regel asynchron zu unterschiedlichen Zeitpunkten wirken die Teammitglieder zusammen. Zu gewissen Zeiten kann auch eine Synchronisation bestimmter Mitarbeiter, z. B. zum zeitgleichen Editieren von Entwurfsdokumenten wichtig sein.

Der Beitrag geht vom bisherigen Stand der Modellierung von Software-Entwicklungsprozessen aus. Obwohl es bereits zahlreiche Forschungsansätze gibt, muss man feststellen, dass gruppenunterstützende Aspekte in den heute eingesetzten Entwicklungsumgebungen unterrepräsentiert sind [1], [2, S. 591ff]. Zwar gibt es eine Reihe von Werkzeugen zur Unterstützung typischer Gruppenaktivitäten [3], doch diese sind noch nicht durchgängig in Werkzeugumgebungen speziell zur Unterstützung von Software-Entwicklungsprozessen integriert.

Auf der Basis einer gründlichen Analyse soll versucht werden, typische Merkmale gruppenorientierter Software-Entwicklungsprozesse zu beschreiben. Im Ergebnis werden essentielle Objekte der Prozessmodellierung herausgearbeitet, die in jedem Fall Elemente einer rechnergestützten Groupwarelösung zur Steuerung der Softwareentwicklung sein sollten. Außerdem soll der gegenwärtige Stand der Unterstützung von Vorgehensmodellen aufgezeigt werden.

Zu diesem Zweck werden anhand einiger typischer Beispiele Vor- und Nachteile genutzter Vorgehensmodelle gegenübergestellt, die in unserem Fall zur Auswahl des V-Modells als Entwicklungsstandard für IT-Systeme des Bundes (ESdIT) geführt haben. An diesem relativ allgemeingültigen Vorgehensmodell, das die Grundlage der weiteren

Ausführungen bildet, wird gezeigt, welche Werkzeugunterstützungsstufen theoretisch denkbar und welche Stufe mit dem folgenden Ansatz erreichbar ist.

Der Hauptteil des Beitrages zeigt, welche Elemente und Mechanismen das Groupwaresystem Lotus Notes unter Wahrung einer hohen Flexibilität zur Verfügung stellt, um die einzelnen Objekte von Softwareentwicklungsprozessen zu modellieren. Mit den verfügbaren Notes-Objekten und Funktionen wie Datenbankzugriff, Notes-Mail und der differentiellen Zugriffsberechtigung können komplexe Software-Entwicklungsprozesse nach dem V-Modell praktisch umgesetzt werden, wobei die Rechnerstützung sich vom beschreibenden „Electronic Process Guide (EPG)" [4] zu einem Workflow-System mit Ablage aller Softwaredokumente in einer Notes-Datenbank entwickelt. Die Richtigkeit dieses Weges geht bereits aus der Tatsache hervor, dass zunehmend Einzelprozesse des Software Engineering Lifecycle sowohl phasenspezifische als auch phasenneutrale Verwaltungsprozesse unter Nutzung der Gruppenaspekte mit Lotus Notes implementiert werden. Die Kopplung mit anderen an der Softwareentwicklung beteiligten CASE-Werkzeugen ist zunehmend über die beiderseitig unterstützten Datenaustauschformate möglich.

Zum Schluss der Ausführungen wird ein mit Lotus Notes/Domino implementierter Prototyp für nach dem V-Modell aufbereitete Software-Entwicklungsprozesse vorgestellt. Aus den Erprobungsergebnissen werden Schlussfolgerungen für die weitere Modellierung von Softwareprozessen mit Notes/Domino gezogen.

2. Charakteristik gruppenorientierter Software-Entwicklungsprozesse

2.1 Merkmale gruppenorientierter Software-Entwicklungsprozesse

Zu Prozessmodellen der Softwareentwicklung gibt es eine Vielzahl von Veröffentlichungen, u. a. von CHROUST [5] und BALZERT [2, S.98]. Ihnen ist eigen, dass die Gruppenorientierung, ausgedrückt in den Wechselwirkungen zwischen den Teammitgliedern und den Aktivitäten an Dokumenten örtlich und zeitlich unterschiedlich verteilt, nur ungenügend modelliert werden.

Ein gruppenorientierter Software-Entwicklungsprozess ist zunächst dadurch gekennzeichnet, dass die Teammitglieder zum Zweck der Zusammenarbeit auf folgenden Ebenen Informationen austauschen [6, S. 26]:

- Kommunikation zur Vermittlung von Informationen ohne notwendige Bekanntschaft der Partner
- Koordination unter Steuerung wechselseitiger Abhängigkeiten von Aktivitäten

- Kooperation mit Vereinbarung gemeinsamer Ziele zum Erreichen eines aufeinander abgestimmten Arbeitsergebnisses (Softwareprodukt).

Auf Basis der Handlungsspielräume beteiligter Gruppenmitglieder unterscheidet man zwischen „teamartiger" und „gefügeartiger" Kooperation [1]. Im ersten Fall spricht man auch von unstrukturierter Kooperation mit gleichgestellten Gruppenpartnern, während für den zweiten Fall vordefinierte strukturierte Arbeitsprozesse typisch sind. Modelle von Software-Entwicklungsprozessen sollten möglichst beide Arten von Kooperation erfassen. Dabei spielt natürlich auch der Zeitpunkt der Modellierung eine wesentliche Rolle, denn teamartige Kooperation mit einer unstrukturierten, im vorhinein nicht planbaren Aufgabenentwicklung wird spätestens mit Beginn oder während der Aufgabenbearbeitung auch zu einem strukturierten Prozess. Daraus folgt, dass die Modellimplementierung sich flexibel wandelnden Anforderungen anpassen soll.

Gruppenorientierte Softwareentwicklungssysteme sind sehr komplex und müssen schon wegen des Vorhandenseins mehrerer Bearbeiter oftmals in Teilsysteme aufgespalten werden. Damit einher geht eine aufgabenbezogene, räumliche und zeitliche Verteilung des kooperativen Software-Entwicklungsprozesses mit folgenden Merkmalen [1]:

- Verteilung der Teilaufgaben auf die Teammitglieder nach ihren Kenntnissen und Fähigkeiten entsprechend einer geeigneten Koordinationsstrategie.
- Räumliche Verteilung von Aktivitäten und Teammitgliedern mit Bereitstellung einer geeigneten Kommunikationsinfrastruktur (Intranet oder Internet).
- Zeitliche Verteilung mit einer Koordination auf Basis asynchron bzw. synchron/parallel durchführbarer Arbeitsvorgänge.

Als kooperative Softwareentwicklung bezeichnet man demzufolge die Planung, Durchführung und Kontrolle aller aufgabenbezogenen, räumlichen und zeitlich verteilten Aktivitäten, die zur Abdeckung des Kommunikations- und Kooperationsbedarfes eines komplexen Softwareproduktes benötigt werden [1]. Zur Realisierung dieser Anforderungen wird neben organisatorischer Unterstützung eine flexible Werkzeugumgebung benötigt.

2.2 Objekte des Software-Entwicklungsprozesses

Notwendig für den Aufbau automatisierter gruppenorientierter Prozessmodelle für die Softwareentwicklung ist die Einführung formaler Beschreibungsmittel. Sie sind die Voraussetzung für die einheitliche, arbeitsteilige und wiederholbare Spezifikation realer Prozessabläufe, die damit eine konkrete Instanz eines abstrakten Softwareprozessmodells darstellen. Aus der Charakterisierung der gruppenorientierten Software-Entwick-

lungsprozesse lassen sich folgende essentielle Elemente von Prozessmodellen ableiten, die im folgenden auch als Objekte des SW-Entwicklungsprozesses bezeichnet werden (Abbildung 1).

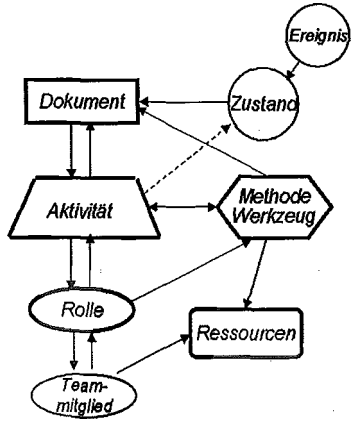


Abbildung 1: Objekte des SW-Entwicklungsprozesses

Die Darstellung wurde erstmalig in [7] benutzt, ist aber noch um einige wichtige Objekte der Gruppenarbeit ergänzt worden.

Im Mittelpunkt steht das *Dokument* als Teamobjekt, das sowohl Eingangsprodukt von *Aktivitäten* sein kann, als auch wieder ausgangsseitig deren Ergebnisse enthält. Wichtige Instanzen des Objektes *Dokument* sind die Bestandteile des Softwareproduktes selbst, wie lauffähige Programme und Dokumentationen. Die *Aktivitäten* beinhalten manuelle oder *werkzeuggestützte* Tätigkeiten zum Erzeugen oder Verarbeiten von *Dokumenten*. Die *Rollen* beschreiben das Können und die Fertigkeiten zur qualifizierten Durchführung von *Aktivitäten*. Und letztendlich sind die *Teammitglieder* die personifizierten Mitarbeiter, die bei Übereinstimmung ihrer Eigenschaften mit den Qualifikationsanforderungen die *Rollen* ausführen. Sie können sich dabei softwaretechnologischer *Entwicklungs-Methoden* oder in einer automatisierten Form auch (CASE-)*Werkzeugen* bedienen. Um die Begrenztheit personeller und finanzieller Quellen sowie von Sachmitteln aufzuzeigen, wurde zunächst abstrakt das Objekt *Ressourcen* eingeführt. Zur Steuerung des dynamischen Prozessablaufes werden schließlich die beiden Objekte *Zustand* und *Ereignis* verwendet. Der *Zustand* bezeichnet den Bearbeitungsstatus von *Dokumenten* (geplant, in Bearbeitung, vorgelegt, akzeptiert). Gelegentlich können auch *Aktivitäten* mit

Zuständen belegt werden. Die *Ereignisse* lösen den Zustandsübergang evtl. verbunden mit einer Aktivität/Aktion auf dem *Dokument* aus.
Die aufgezählten Objekte stellen die atomaren Bausteine jeder Modellierung von gruppenorientierten Softwareentwicklungsprozessen dar.

3. Stand der Modellierung von Software-Entwicklungsprozessen

Bemühungen zur Modellierung gruppenorientierter Software-Entwicklungsprozesse stammen aus mehreren Richtungen, sie lassen sich aber alle nach [8] in ein Spektrum vier unterschiedlicher Koordinationsmodelle einordnen. Zur ersten Gruppe der *Formularorientierten Kommunikationsmodelle* gehören meist im Bürosektor anzutreffende Lösungen, die auf strukturierten Formularen basieren, und die selbst Abarbeitungsregeln enthalten können [9]. Auf dem Ansatz der Sprechakttheorie beruhen *Konversationsorientierte Koordinationsmodelle* [10].
Die genannten beiden Modellarten wurden bisher im Kontext der Softwareentwicklung so gut wie nicht eingesetzt. Anders sieht es mit den beiden folgenden Modelltypen aus. *Kommunikationsorientierte Koordinatiosmodelle* basieren auf einer rollenorientierten Beschreibung des Verhaltens von Teamobjekten, die über Nachrichten und den Ablauf symbolisierende Regeldefinitionen miteinander kommunizieren. Ein in diese Gruppe einzuordnendes Projekt ist MERLIN [11], das zur Prozess- und Kooperationsbeschreibung eine Wissensbasis auf Grundlage der Sprache P r o l o g benutzt. Die bedeutendste Gruppe, speziell auch für die Softwareprozessmodellierung, sind die *Vorgangsorientierten Koordinationsmodelle.* Im Mittelpunkt steht hier die Koordination vorstrukturierter Einzelaktivitäten (gefügeartige Kooperation) zu einem Vorgang, wobei eine zentrale Kontrollinstanz den Aktionsfluß zwischen den Aktivitäten überwacht. Diese Modellart bildet die Basis für alle Workflow-Management-Systeme [12]. Von den zahlreichen Produkten, die schwerpunktmäßig für das Management von Geschäftsprozessen eingesetzt werden, sei hier ein spezielles für die Softwareprozessmodellierung erwähnt [13]. Es beruht auf einer erweiterten Form von Petrinetzen, den sogenannten FUNSOFT-Netzen, in denen transaktionsgesteuert Aktivitäten Software-Dokument-typen bearbeiten.
Bei praktischen Softwareprozessen können die FUNSOFT-Netze sehr komplex und damit wenig flexibel handhabbar werden, so dass sie praktisch nicht genutzt wurden.

Die im folgenden vorgeschlagene Modellierung auf der Grundlage von Lotus Notes gehört ebenfalls zur Gruppe der *Vorgangsorientierten Koordinationsmodelle.* Durch die Verwendung von Modellelementen (*Formulare*) aus den anderen Modellarten kann die

Beschreibung wesentlich variabler spezifiziert werden, was eine gute Anpassung an die praktisch unterschiedlichsten Problemstellungen ermöglicht.

3.1 Stand der Vorgehensmodellierung

Speziell auf dem Gebiet der Softwaretechnik werden Prozessmodelle in der Form von Phasenmodellen und Vorgehensmodellen abgebildet [2, S. 98]. Gegenüber den einfacheren Phasenmodellen, die hauptsächlich den zeitlichen Verlauf der *Aktivitäten* (Phasen) beschreiben, werden in Vorgehensmodellen als eine höhere Qualitätsstufe die Ergebnisse der *Aktivitäten*, die *Dokumente* sowohl in ihrem prinzipiellen Aufbau als auch in ihrem Bearbeitungsfluss definiert. Je nach dem Grad der Konkretisierung von Vorgehensmodellen kommen die weiteren Prozess-Objekte (Abb. 1) hinzu.

Zunächst steht die Frage, welches Vorgehensmodell soll für die spätere Modellierung ausgewählt werden. Hier sei ein Vergleich objektorientierter Vorgehensmodelle der industriellen Praxis herangezogen [15], die zunehmend auch mit Rechnerstützung angeboten werden. Wir beschränken uns bei der Charakteristik auf die allgemeine Nutzbarkeit, Offenheit und Allgemeingültigkeit der beschriebenen Vorgehensmodelle.

Ein speziell bei Hewlett-Packard und für OO-Projekte eingesetztes Vorgehensmodell nennt sich Fusion [16]. Es umfaßt bis auf die *Einführung* alle Hauptphasen, kennt die Prozess-Objekte außer der *Rolle*, schließt einen Projektmanagementprozess ein und arbeitet mit einem zentralen Data Dictionary als gemeinsame Ablage aller Entwicklungsdokumente. Mit Worldwide Solution Design and Delivery Method (WSDDM) wird ein umfassendes Methodenpaket von IBM zur objektorientierten Vorgehensmodellierung angeboten [17]. Unterstützt werden alle Entwicklungsphasen einschließlich des *Tests*, eines umfassenden Repositories und des Projektmanagements. Die Modellpräsentation existiert maschinenlesbar als Hypertextsystem. Von führenden CASE-Tool Herstellern werden die Vorgehensmodelle RUP (Rational Unified Process) und SELECT Perspective angeboten. RUP basiert vordergründig auf einer auf der Unified Modeling Language (UML)[14] basierenden Software-Engineering Methode der Firma Rational [18]. SELECT Perspectiv ist eine Methode der Firma SELECT Software Tools und versteht sich als Sammlung in der Industrie bewährter Techniken (*best practices*) unter besonderer Berücksichtigung des Bereitstellungsprozesses von Komponenten (*component process*) [19]. Beide unterstützen sämtliche Phasen des Entwicklungsprozesses, wobei RUP zusätzlich *Workflows* zur Widerspiegelung der Prozesssicht definiert. Aus einer Konsolidierung verschiedener objektorientierter Methoden durch Mitglieder des OPEN Konsortiums entstand das Vorgehensmodell OPEN [20]. Es beinhaltet ein vertragsgetriebenes Vorgehensmodell, die Modellierungssprache OML (Object Modeling Language) und zeichnet sich durch

zahlreiche Aktivitäten zum Akzeptanztest aus. Schließlich sei noch das nationale Anwendungsentwicklungsmodell AE-Modell des Informatikzentrums der Sparkassenorganisation (SIZ) genannt [21]. Es liegt jetzt in einer objektorientierten Version vor und deckt alle Entwicklungsphasen in Modellvarianten für die normale, die inkrementelle und die komponentenbasierte Entwicklung sowie für das evolutionäre Prototyping ab.

Alle aufgeführten Vorgehensmodelle besitzen nach der Steuerung der durchzuführenden *Aktivitäten* Mechanismen zur Prozesssteuerung. Sie lassen sich nach [15] klassifizieren in *aktivitätenorientiert* (nach Fertigstellung der Aufgabe), *ergebnisorientiert* (nach Vorliegen des *Dokuments*) und *vertragsorientiert* (nach vertragsdefinierten Entscheidungen). In der Praxis können auch Mischformen auftreten. Die Werkzeugunterstützung der Vorgehensmodelle ist unterschiedlich ausgeprägt. Bis auf *Fusion* und OPEN existieren hypertext- oder browserbasierte Online Handbücher bzw. Präsentationen. Die Methodenstützung erfolgt über vorgehensmodell-spezifische (*RUP, SELECT-Perspective*) Werkzeuge oder über beliebige marktgängige Tools. Eine durchgängige Werkzeugstützung des gesamten Prozessmodells der Softwareentwicklung ist uns bei keinem der genannten Vorgehensmodelle bekannt.

3.2 Auswahl des V-Modells als Entwicklungsstandard für IT-Systeme

Die aufgeführten Vorgehensmodelle werden durch eine im militärischen Bereich und für die gesamte Bundesverwaltung in Deutschland verbindliche Vorschrift, das *V-Modell*, ergänzt. Sein Name stammt von den V-förmig angeordneten *Aktivitäten*, wobei Konstruktions- und Validationsaufgaben direkt gegenüber angeordnet sind [22]. Das herausragendste Merkmal des V-Modells als EsdIT sind seine Allgemeingültigkeit und die Möglichkeit der (werkzeuggestützten) Anpassung an konkrete Projekte aus den verschiedensten Problembereichen (Tailoring) [23]. Das V-Modell setzt sich aus vier Submodellen zusammen, aus dem Hauptprozess *Systementwicklung* (SE) und den für die projektbegleitenden Tätigkeiten *Qualitätssicherung* (QS), *Konfigurationsmanagement* (KM) und *Projektmanagement* (PM), die miteinander eng verzahnt arbeiten (Abb. 2).

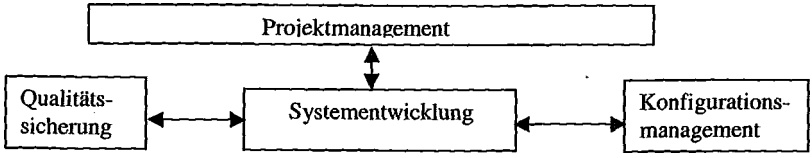


Abbildung 2: Submodelle des V-Modells

Im Rahmen des Beitrages liegt das Hauptaugenmerk auf dem Submodell SE, wobei Querverbindungen zu den projektbegleitenden Modellen durchaus zu beachten sind. Nach unserem Kenntnisstand gibt es zum Projektmanagement [24] und zum Konfigurationsmanagement [25] bereits rechnergestützte Produktsteuerungs- und –verwaltungssysteme auf der Grundlage von Lotus Notes.

Im V-Modell werden als Objekte der Softwareentwicklung (Abbildung 1) beschrieben:

- Aktivitäten (hierarchisch gegliedert)
- Dokumente/Produkte mit einem grundsätzlichen Gestaltungs- oder Produktmuster
- Zustände, die die Dokumente im SE-Prozess durchlaufen
- Ereignisse, die Dokumentweitergaben an andere Aktivitäten auslösen (Transitionen)
- Rollen zugeordnet den Aktivitäten.

Weiterhin legt das V-Modell eine *Methodenzuordnung* und *funktionelle Werkzeuganforderungen* fest, mit denen Hinweise zur Durchführung der Aktivitäten gegeben werden. Einen möglichen Dokumentenfluß zwischen den Aktivitäten der 1. Ebene im Submodell SE zeigt Abbildung 3.

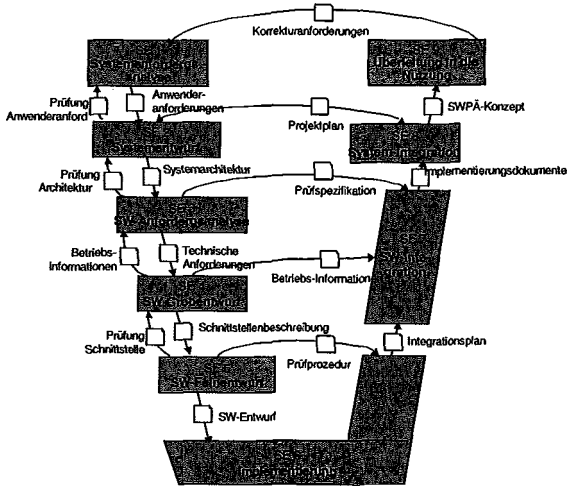


Abbildung 3: Möglicher Dokumentenfluß der SE im V-Modell

Für die Auswahl des V-Modells sprachen neben seiner Allgemeingültigkeit und zunehmenden Verbreitung in öffentlichen und industriellen Einrichtungen folgende weitere Gründe:

- Unabhängigkeit von Organisationen und Fachgebieten.
- Es ist ein breitgefächerter Methodeneinsatz möglich.
- Auf Basis der formulierten funktionellen Werkzeuganforderungen bleibt das Spek-
- trum einsetzbarer Werkzeuge offen.
- Für das Tailoring werden bereits Werkzeuge erfolgreich eingesetzt [26],[27].
- Bei der Anwendung des V-Modells wurde ein reicher Erfahrungsschatz gesam-
- melt, der beispielsweise zur Gründung der Interessengemeinschaft ANSSTAND
- führte [23, S.247].

Die angeführten Gründe zusammen mit der Offenheit und ausführlichen Dokumentation, die jedermann zugänglich ist, gaben den Ausschlag, das V-Modell als Grundlage einer rechnergestützten Modellierung von Software-Entwicklungsprozessen auszuwählen.

3.3 Stufen zur rechentechnischen Stützung von Vorgehensmodellen

Der Rechnereinsatz zur Unterstützung des Einsatzes von Vorgehensmodellen ist breit gefächert. Um die zu entwickelnde Lösung einordnen zu können, soll in Tabelle 1 ein stufenweise zunehmender Einsatz von Werkzeugen angegeben werden, beginnend bei der Stufe 0, wo das Management hinter der Nutzung des Vorgehensmodells steht, aber noch kein Werkzeug benutzt wird.

Stufe	Werkzeugunterstützung	Tool-Beispiel
0	Stützung durch höheres Management, kein Werkzeugeinsatz	-
1	Papierdokumentation des Vorgehensmodells, mit Werkzeug (Textverarbeitung) erzeugt	Word, FrameMaker
2	Werkzeuggestütztes Online-Handbuch mit Hyperlinks zunehmend webbasiert (Electronic Process Guide) zur Vorgehensmodell-Präsentation	[4],[28],[29]
3	Werkzeuggestütztes Anpassen von Vorgehensmodellen an spezielle Prozesse	[26],[27]
4	Einbindung ausgewählter Software-Entwicklungswerkzeuge (CASE) zur Methodenstützung	s. [23]
5	Einbindung phasenneutraler Verwaltungswerkzeuge für Projektmanagement, Qualitätsicherung und Konfigurationsmanagment	[24],[25]
6	Rechnergestützte Prozesssteuerung zur Dokumenten-/Produktverwaltung und Integration von Einzelwerkzeugen	[1]
7	Vollständige Integration eines Vorgehensmodells in eine Software-Entwicklungsumgebung (CASE)	[19],[30]
8	Software-Produktionsumgebung (Factory) für beliebige Vorgehensmodelle	

Tabelle 1: Unterstützungsstufen von Vorgehensmodellen durch Werkzeuge

Für die höchste Stufe 8 sind keine Beispiele bekannt. In Stufe 7 ließe sich mit einigen Abstrichen das CASE SELECT Enterprise einordnen. In dieser Stufe befindet sich auch das CASE-Tool INNOVATOR 6.2 [30], das den Produkten des V-Modells direkt die zugehörigen Werkzeuge zur Dokumentbearbeitung zuweist. Im Rahmen dieses Beitrages soll eine Modelllösung für Stufe 6 entwickelt werden. Auf dem gleichen Niveau ist die Lösung von ALTMANN [1] angesiedelt, allerdings steuert er nur über in C++ und einem Framework implementierte Fenster parallel Prozess- und

Produktsichten. Ein zustandsorientiertes Dokumentenmanagement mit einer Workflowsteuerung findet mit dem *Cooperation Assistent* [1] nicht statt.

4. Modellierung des V-Modells mit Lotus Notes

Die zu modellierenden Objekte eines SW-Entwicklungsprozesses, die sich 1:1 abgebildet im V-Modell wiederfinden sollen, zeigt Abbildung 1. Zur Umsetzung in eine Groupware-Lösung stellt Notes folgende Gestaltungelemente zur Verfügung (Abbildung 4). Sie sind geeignet, um schon mit Programmierung per Menü die V-Modell-Objekte zu entwerfen.

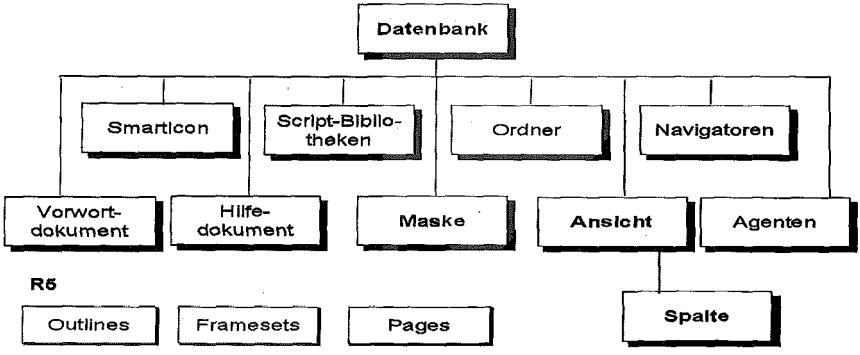


Abbildung 4: Gestaltungselemente von Notes-Anwendungen

Die Einzelelemente lassen sich wie folgt realisieren.

4.1 Produkt/Software-Dokumente

Das Notes Element *Maske* ist das zentrale Mittel, um Dokumente strukturieren und mit Informationen im Laufe der Bearbeitung füllen zu können. Die Informationen können vom unterschiedlichsten Typ sein, angefangen von Texten über Grafiken, Tabellen bis zu Animationen. Zur Maske gehören als Unterelement *Felder*, die die Datenobjekte aufnehmen. Für jedes Notes-Gestaltungselement gelten erprobte Sicherheitsmechanismen. So kann die Feldsicherheit durch Verschlüsselungsalgorithmen (des amerikanischen Verteidigungsministeriums) und die Dokumenten- und Maskensicherheit über den kontrollierten Lese- bzw. Schreibzugriff der Autoren, Leser bzw. anderer Rollen gewährleistet werden [31].

Wie Abb. 4 zeigt, werden alle Notes-Elemente, auch die Dokumente (entspricht ausgefüllte Maske) in der (Notes-)Datenbank abgelegt. Zum Überblick über alle Dokumente der Datenbank gibt es das Element *Ansicht*, das zeilenförmig (s. Abbildung

11) alle Dokumente auflistet. Jede Ansicht besteht aus *Spalten*, in denen zu Ordnungszwecken wichtige Dokumentenfelder angeordnet sind, die das Dokument übersichtlich charakterisieren.

Die Bereitstellung der aufgeführten vier Noteselemente gestattet es relativ einfach, Softwaredokumente zu modellieren, sie in Notes-Datenbanken zu erfassen und weiter zu verarbeiten sowie über das Verschicken als Mail-Dokument gezielt anderen Mitarbeitern der Gruppe zur weiteren Bearbeitung zu übergeben.

4.2 Rollen / präsentierende Mitarbeiter

Das fortgeschriebene V-Modell unterscheidet über alle Submodelle noch ca. 23 Rollen [23]. Ihre Rechte in den Submodellen oder auch in einem Submodell werden beschrieben durch eine für die Gruppe gültige Access Control List (ACL), die ebenfalls Bestandteil der Notes-Datenbank ist. Notes unterscheidet die folgenden in Abbildung 5 wiedergegebenen personifizierten Zugriffsebenen.

Manager:	kann alle Operationen auf einer DB ausführen: z.B. Masken, Ansichten erstellen, löschen; alle Dokumente lesen, erstellen, bearbeiten
Entwickler:	kann alle Operationen des Managers ausführen ausser Zugriffskontrollliste verwalten u. Datenbanken löschen
Editor:	kann alle Dokumente lesen, gestalten und bearbeiten
Autor:	kann Dokumente lesen, neue erstellen, aber nur die selbst erstellten bearbeiten
Leser:	kann existierende Dokumente lesen
Archivar:	kann neue Dokumente erstellen, aber existierende nicht lesen auch nicht die selbst erstellten
kein Zugriff:	man kann die Datenbank nicht öffnen

Abbildung 5: Personifizierte Zugriffsebenen in Notes

Mittels Einstellungen in der Zugriffskontrollliste (ACL) ist es möglich, ganz dezidierte Einzelrechte für jede Rolle und die sie präsentierenden Mitarbeiter zu vergeben. Das

geschieht über einen differenzierten Zugriff zu Masken, ausgewählten Abschnitten und Feldern der Maske oder zu ihren Ansichten.

4.3 Zustände von Dokumenten

Laut V-Modell können (Teil-)Produkte im Laufe ihrer Entwicklung die in Abbildung 6 gezeigten *Zustände* annehmen [23].

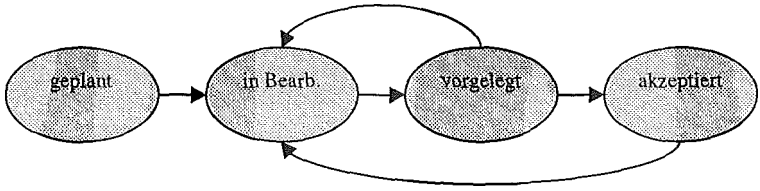


Abbildung 6: Dokumenten-Zustandsübergänge nach dem V-Modell

Dabei gilt die Bedingung, dass der Übergang von einem Zustand in einen anderen grundsätzlich durch eine *Aktivität* auszulösen ist, d. h. am Ende der Produktbearbeitung ist der *Zustand* des *Dokumentes* zu setzen.

Von diesem Grundsatz geht auch die Modellierung in Notes aus, in dem für die Beschreibung des Zustands eines Dokuments einfach ein zusätzliches Schlüsselwort in die Maske eingeführt wird. Die Schlüsselwörter können textlicher Natur sein. Es lassen sich aber auch Radiobutton für jeden Zustand im Dokument einführen. Das Markieren des entsprechenden Zustandsfeldes kann *statisch* durch die berechtigten *Rollen* der *Aktivität* geschehen. Es ist aber auch möglich, den Zustandsübergang *dynamisch* über einen *Dokumenten*-Workflow zu realisieren.

4.4 Aktivitäten und Workflows

Unter *Aktivitäten* versteht man die im Vorgehensmodell geregelten oder enthaltenen Tätigkeiten, die durch ihre Abwicklung und ihre Ergebnisse (Produkte/*Dokumente*) eindeutig beschrieben sind.
Sie können aus der Sicht der Notes-Gestaltungselemente (Abbildung 4) wiederum als Masken (Menge von Feldern) erfasst und damit als Abwicklungsvorschrift hierarchisch gegliedert in einer Notes-Datenbank gespeichert werden. In der Maske legt man einen Teil der Funktionalität der *Aktivität* (Anwendung) fest und kann ihr visuelles Aussehen entsprechend gestalten. Mit der Ansicht, die charakteristische, die *Aktivität* beschreibende Spalten enthält, werden alle enthaltenen Tätigkeiten aufgelistet, bei

Bedarf können sie nach festzulegenden Kriterien sortiert, ausgewählt und kategorisiert werden. Auch die hierarchische Ebenenarchitektur der *Aktivitäten* kann angezeigt und bei Bedarf für die untergeordneten Elemente geöffnet werden.
Zur Modellierung von Workflows unterstützt Notes zwei grundsätzliche Prinzipien (Abbildung 7).

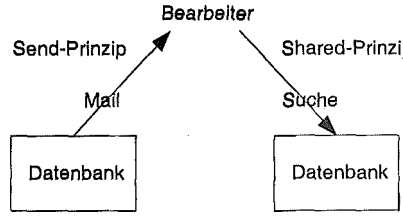


Abbildung 7: Grundprinzipien zur Realisierung von Workflow

Entweder die Bearbeiter werden durch E-Mail (Send-Prinzip) auf jedes neue Dokument hingewiesen oder sie müssen es z. B. nach akustischer Benachrichtigung in einem Arbeitsbereich der Datenbank selbst aufsuchen (Shared-Prinzip). Seit kurzer Zeit steht die kommerziell angebotene Software Domino Workflow 2.0 zur Verfügung. Sie besitzt als zusätzliche Komponente die Möglichkeit, Workflows grafisch zu definieren und anschließend in speziellen Notes-Datenbanken abzuspeichern. Die grafische Workflow-Definition kann auch innerhalb eines Web-Browsers verwendet werden.
Die Workflows des V-Modells sind zwischen *Aktivitäten* zu realisieren, die von zugeordneten *Rollen* durchgeführt werden, indem sie *Dokumente* erzeugen, bearbeiten oder ablegen und sie dabei von einem *Zustand* in einen anderen versetzen.

4.5 Ereignisse zur Weitergabe von Produkten

Das *Ereignis* kann Auslöser des Übergangs eines Produkts/*Dokuments* vom *Zustand* „geplant" in den *Zustand* „in Bearb." sein. Neben diesen auf das *Dokument* ausgerichteten *Ereignissen* gibt es solche bezogen auf die *Aktivität*, z. B. bei Eintreffen oder Verlassen eines Produktes, die ebenso auf die Personen und damit auf *Rollen* bezogen werden können. Natürlich spielen auch zeitliche Ereignisse eine Rolle, z. B. durch den Kalender gesteuert oder bei Erreichen eines bestimmten Termins.
In der Regel werden durch *Ereignisse Aktivitäten* ausgelöst. Ihre Nachbildung mit Notes gestaltet sich völlig problemlos, indem die vielfältigen Bedienereignisse mit Maus oder Tastatur sowie auch interne Ereignisse als Initiator von Rechenoperationen genutzt werden.

4.6 Abarbeitung des Modells im WWW und Koppelbarkeit mit CASE

Bei der heutigen Dominanz des Internet sollte die Modellierung des V-Modells nicht nur zwischen Notes-Clients und zugehörigen Notes-Servern möglich sein, sondern es sollten normale Web-Nutzer ohne Vorhandensein einer Notes-Software mit der Lösung arbeiten können. Das heißt, mit einem handelsüblichen Web-Browser sollten sie Zugriff und auch abgegrenzten Einfluss auf die Produktentwicklung nach dem Vorgehen des V-Modells haben. Da der Domino-Server als Internet-Server fungiert, müssen lediglich in der gemeinsamen Notes-Datenbank die Zugriffsrechte entsprechend gesetzt werden. Spezifische internettypische Layoutanforderungen an die Masken sind schon bei der Entwicklung der Notes-Datenbank zu beachten.

Falls das Notes Prozesstool keine die Methoden unterstützende Werkzeuge enthält wie [32], ist eine Datenaustauschmöglichkeit bzw. Kopplung mit CASE vorzusehen. Diese Schnittstelle kann zum einen zur Übergabe von V-Modell-Produkten, zur Weiterverarbeitung in CASE, aber auch umgekehrt zur Einbettung von CASE-Ergebnissen in V-Modell-Dokumente genutzt werden.

Das Problem dieses Datenaustausches besteht im Vorhandensein offener passfähiger Schnittstellen. Eine sehr einfache Variante ist z. B. die Nutzung der Zwischenablage, über die UML-Diagramme in Notes-Dokumente eingebunden werden können. Eine weitere Möglichkeit bietet die Einbettung von OLE-Daten in das Dokument bzw. die Verknüpfung in ihm. Der dritte eigentliche normale Fall besteht in der Bereitstellung standardisierter Repository-Austauschformate wie XMI. Leider unterstützen momentan weder Notes noch die meisten CASE-Tools dieses Format, so dass auf die Werkzeuge zugeschnittene Sonderwege gegangen werden müssen [32].

5. Stand der Implementierung des Prototypen zur Unterstützung des V-Modells

Der Prototyp besteht aus einer Projektdatenbank, in der alle für das aktuelle Projekt benötigten Objektinstanzen wie Produkte, Aktivitätsbeschreibungen, Rollen usw. abgelegt sind. [33] Zu Beginn der Arbeit sind durch den Administrator bzw. innerhalb zugelassener eigener Berechtigungen den konkreten Personen ihre Rollen und Verantwortlichkeiten zuzuordnen. Dazu werden Funktionen in der Zugriffskontrollliste definiert und entsprechend zugewiesen, wie in Abb. 8 beispielhaft gezeigt ist.

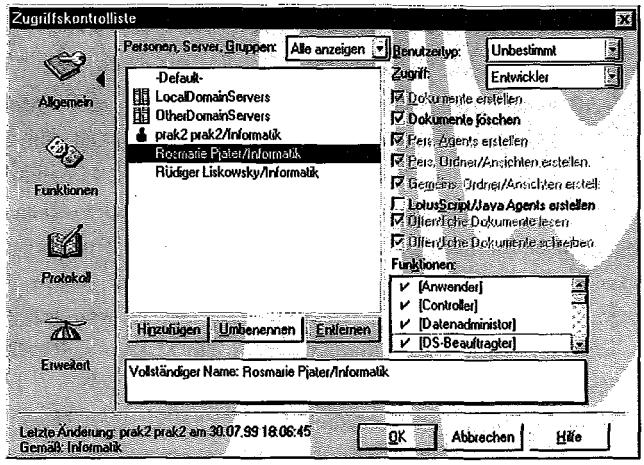


Abbildung 8: Zugriffskontrollliste für Rollenzuordnung

Die zu bearbeitende Aktivität innerhalb eines Submodells bzw. einer übergeordneten Hauptaktivität wird über eine Auswahlansicht festgelegt (Abb. 9).

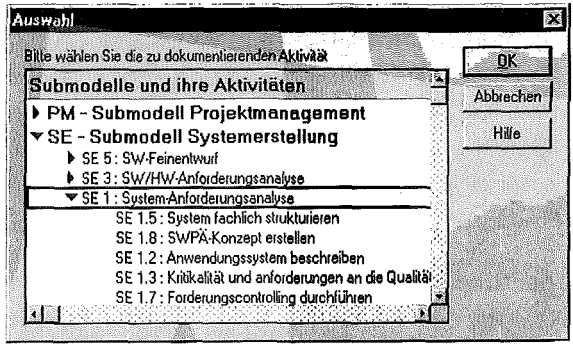


Abbildung 9: Auswahl aus Aktivitäts-Ansichten

Die Aktivität selbst ist in einer Aktivitätsmaske beschrieben und (als Notes-Dokument) in der Datenbank gespeichert. Zu dieser Aktivität können danach die relevanten Produkte als Antwortdokument über die zugehörige Dokumentmaske bearbeitet und durch die Kennung des Zustandes im Produktfluss zu einer folgenden Aktivität aufgefunden werden (Abbildung 10).

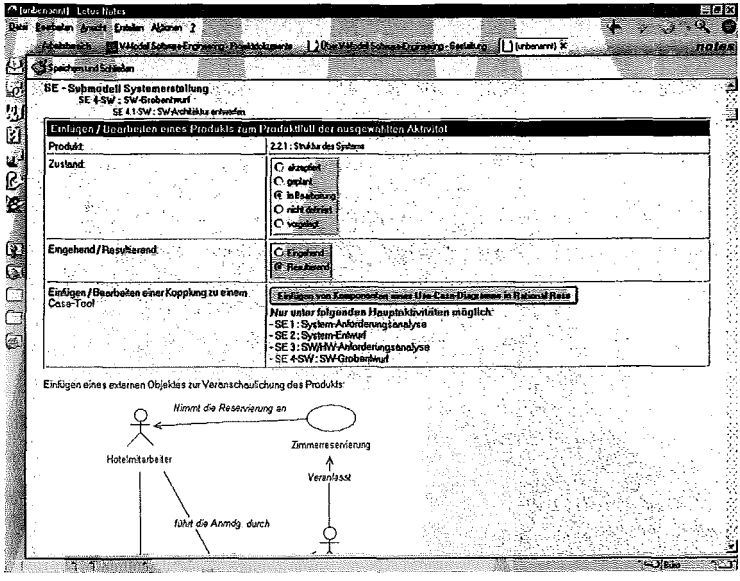


Abbildung 10: Maske zur Dokumentation des Produktflusses (mit Use Case von CASE)

Mittels Navigatoren auf der linken Fensterseite wird der Nutzer durch die Anwendung geführt. So werden z. B. die entsprechenden Ansichten aufgerufen, über die die vorhandenen Dokumente des V-Modells nach unterschiedlichen Gesichtspunkten ausgewählt und kategorisiert angezeigt werden können (Abb. 11).

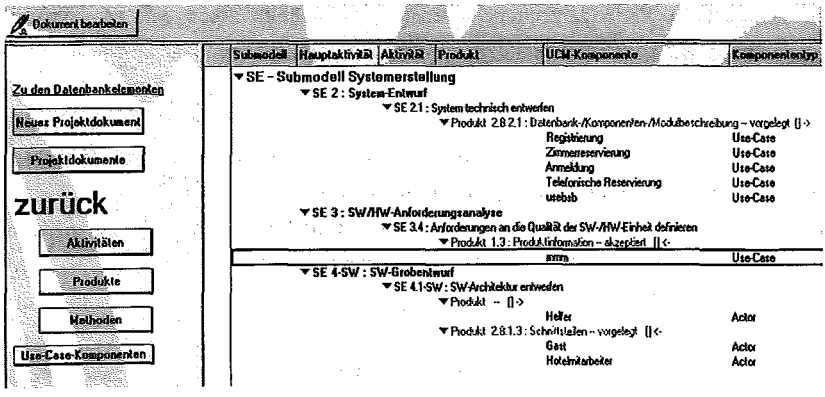


Abbildung 11: Navigator mit Ansichten der den Aktivitäten zugeordneten Produkte

Eine weitere Stufe der Prototypimplementierung ist die Realisierung eines flexiblen Workflows über die speziellen grafischen Fähigkeiten der Workflowdefinition mit Domino Workflow 2.0. Einen möglichen Ablauf, initiiert durch den Projektleiter, ausgehend von den Anwenderanforderungen ist in Abbildung 12 für die Hauptaktivität SE2 Systementwurf dargestellt.

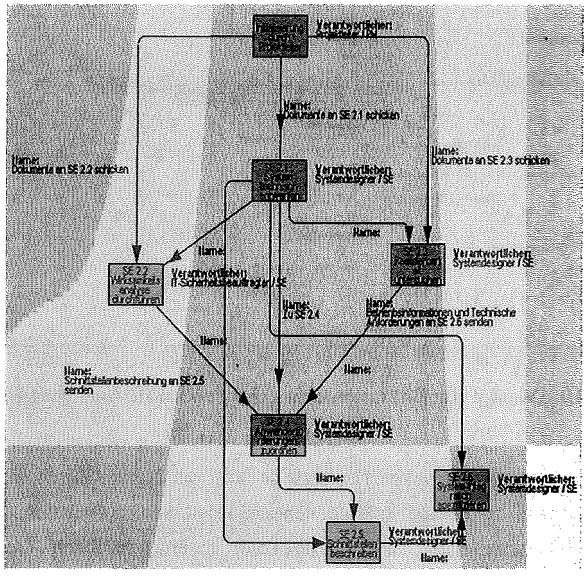


Abbildung 12: Modellierung von SE2 mit dem Domino Workflow Architect

Die angegebenen Verbindungspfeile werden ausschließlich durch den Produktfluss instanziiert, wobei an ihnen neben dem zu versendenden Dokument auch die zuständige Rolle notiert werden kann.

Auf Basis der geschilderten Eckpunkte wurde ein Prototyp zur Modellierung des V-Modells mit variabler Auswahl von Aktivitäten und Dokumenten bei ihrer gleichzeitigen Speicherung in einer Notes-Datenbank geschaffen [33].

Eine Tailorisierung des V-Modells nach Nutzeranforderungen ohne Verletzung von Vorrangbeziehungen [26],[27] ist momentan nicht Bestandteil des Projekts. Es wird davon ausgegangen, dass das Zuschneiden vorher erfolgt und danach das resultierende Modell in dem Notes-System weiterverarbeitet wird. Einen großen Vorteil der Lösung sehen wir darin, dass das Vorgehensmodell nicht nur im Sinne eines EPG beschrieben wird, sondern für alle mitarbeitenden Personen zugleich Handlungsanleitung wird. Es werden Schablonen für die Dokumente bereitgestellt, sie werden ereignisgesteuert

bearbeitet und weitergegeben sowie als Bestandteile der Produktdokumentation gespeichert. Damit wird ein weiterer Beitrag zur Rationalisierung der gruppenorientierten Softwareentwicklung geleistet.

6. Schlußfolgerungen und Ausblick

In dem Beitrag wurden zunächst gruppenorientierte Software-Entwicklungsprozesse charakterisiert und die typischen enthaltenen Objekte herausgearbeitet. Die Analyse beschränkte sich aber nicht nur auf den Prozess selbst, sondern wollte den gegenwärtigen Stand der Nutzung von CSCW-Werkzeugen bzw. von Groupware verdeutlichen. Da die Verwendung von Werkzeugen eine kritische Untersuchung der für den Software-Entwicklungsprozess benutzbaren Modelldarstellungen voraussetzt, wurde dieses Gebiet beginnend von allgemeinen Koordinationsmodellen bis hin zur Umsetzung konkreter Vorgehensmodelle ansatzweise beleuchtet. Es ging dabei auch darum, den Stand der rechentechnischen Unterstützung von Vorgehensmodellen nach 8 Unterstützungsstufen zu klassifizieren.
An der TU Dresden wurde sich wegen der Universalität, der leichten Anpassbarkeit und der weiten Verbreitung für das V-Modell als konkretes Vorgehensmodell entschieden. Als rechentechnische Groupwarebasis wurde Notes/Domino gewählt, insbesondere weil die Notes-Gestaltungselemente eine flexible Umsetzung der essentiellen Objekte von Softwareprozessen bis hin zur Definition von Workflows gestatten.
Die vorgestellte Lösung ist ein erster Ansatz zur Modellierung des kompletten Submodells Systementwurf (SE) mit den Mitteln von Lotus Notes. Ein unbedingtes Ziel ist die Erprobung im Rahmen eines lohnenden Projektes der Praxis. Deshalb soll die Vorstellung der Lösung auf dem Workshop dazu beitragen, einen interessierten Partner aus der Industrie oder Verwaltung zu finden.
Unabhängig davon wird eine Weiterentwicklung auf mehreren Gebieten stattfinden. Zunächst soll die Integration der Komponente Domino-Workflow 2.0 vollständig abgeschlossen werden, um bei Änderungen des Produktflusses schneller reagieren zu können. Die Tailorisierung des V-Modells ist zum gegenwärtigen Zeitpunkt noch kein Bestandteil unserer Lösung. Notes stellt alle Mittel bereit, dies auch selbst innerhalb einer Anpassungskomponente zu tun. Bisher fehlte in der Hochschule einfach die Zeit, die Tailorisierungsalgorithmen mit den verfügbaren Notes Programmiermitteln umzusetzen.

Ein wesentlich attraktiveres Gebiet stellt die weitere Verbreitung der Lösung auf die Submodelle KM, PM und QS dar. Auf der Grundlage des Vorhandenseins von Notes-Lösungen für einige Gebiete ergeben sich unterschiedliche Realisierungsmöglichkeiten.

Entweder die Notes-Datenbanken lassen sich ohne tiefgreifende Anpassungen in einer gemeinsamen Anwendung koppeln oder man muss auf die einzelnen Gestaltungselemente zurückgehen und sie neu in einem Notes-Projekt vereinigen. Schließlich sind weitere CASE-Tools hinsichtlich des möglichen Datenaustausches zu analysieren. Es wird die Erwartung geäußert, dass mit einem Vorankommen standardisierter Austauschformate auf Basis von OLE-Objekten bzw. XMI die Kopplung mit CASE als Methodenwerkzeuge sich verbessern wird. Die Nutzung verschiedenster Werkzeuge in einem ganzheitlichen Prozess ist bereits jetzt Normalität, so dass ein derartiges System vergleichbar mit Unterstützungsstufe 7 wäre. Bei Nachweis der Unabhängigkeit von einem speziellen Vorgehensmodell in der Praxis könnten auch andere firmenneutrale Modelle unterstützt und damit die Idealstufe 8 erreicht werden.

7. Literatur

[1] Altmann, J., Pomberger, G.: Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge; in Scheer, A.-W., Nüttgens, M.(Hrsg.): 4. Internationale Tagung Wirtschaftsinformatik, Physica-Verlag 1999

[2] Balzert, H.: Lehrbuch Software-Technik; Softwaremanagement, Software-Qualitätssicherung, Unternehmensmodellierung; Spektrum-Verlag 1998

[3] Schwabe, G., Krcmar, H.: CSCW-Werkzeuge; Wirtschaftsinformatik 38 (1996) H.2, S. 209-225

[4] Becker-Kornsaedt, U.: Der V-Modell Guide: Web-basierte Unterstützung eines Prozess-Standards, IESE-Bericht Nr. 023.99/D (www.iese.fhg.de/pdf_files_iese_023_99.pdf)

[5] Chroust, G.: Modelle der Software-Entwicklung; Oldenbourg Verlag 1992

[6] Teufel, S., Sauter, Ch., Mühlherr, T., Bauknecht, K.: Computerunterstützung für die Gruppenarbeit; Addison-Wesley 1995

[7] Deiters, W.: Systematisches Management von Geschäftsprozessen; Fraunhofer Institut für Software und Systemtechnik, Dortmund 1993

[8] Dittrich, J.: Koordinationsmodelle für computerunterstützte Gruppenarbeit;

in Friedrich, J., Rödiger, K.-H.(Hrsg.): Computergestützte Gruppenarbeit (CSCW); BG Teubner Verlag 1991

[9] Malone, T., Crowston, K.: What is coordination theary an how can it help design cooperative work systems; Proceedings of the Third Conference on Computer-Supported Work, S. 357, Los Angeles 1990

[10] Flores, F., Graves, M., Hartfield, T., Winograd, T.: Computersystems and the design of organisational interaction; ACT-Transactions on Office Information Systems 6(1988)H.2, S. 153-172

[11] Junkermann, G., Peuschel,W., Schäfer, W., Wolf, S.; Merlin: Supporting Cooperation in Software Development through a Knowledge-based Environment; in Finkelstein (Hrsg.): Research Studies Press; John, Wiley & Sons 1994

[12] Jablonski, S.: Workflow-Management-Systeme: Motivation, Modellierung und Architektur; Informatik-Spektrum 18(1995)H.1, S. 13-24

[13] Deiters, W., Gruhn, V.: Vorgangsmanagement mit FUNSOFT-Netzen und CorMan; Fraunhofer Einrichtung für Software- und Systemtechnik, Dortmund 1993

[14] Rational Software Corporation (ed.): Unified Modeling Language, Dokumentation Set Version 1.3, Juni 1999 (http://www.rational.com/uml/index.jtmpl)

[15] Noack, J., Schienemann, B.: Objektorientierte Vorgehensmodelle im Vergleich; Informatik-Spektrum 22(1999)H.3, S. 166-180

[16] Malan, R., Letsinger, R., Colemann, R.: Object-Oriented Development At Work: Fusion in the Real World; Prentice Hall 1995

[17] IBM Object Oriented Technology Center: Developing Object Oriented Software; Prentice Hall 1997

[18] Kruchten, P.: The Rational Unified Process: An Introduction; Addison-Wesley 1998 (http://www.rational.com/products/rup)

[19] Allan, P., Forst, S.: Component-Based Development for Enterprise Systems. Applying the SELECT Perspective; Cambridge University Press 1998

[20] Graham, I., Henderson-Sellers, B., Younessi, H.: The OPEN Process Specification; Addison-Wesley 1997

[21] Noack, J.: AE-Modell: Prozessorientiertes Rahmenwerk für die Anwendungsentwicklung in der Sparkassenorganisation; Information Management H.4(1997), S. 20-26

[22] Boehm, B.W.: Verifying and Validating Software Requirements and Design Specifications; in: IEEE Software, Jan. 1984, S. 75-88

[23] Dröschel, W., Heuser, W., Midderhoff, R.: Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97; Oldenbourg Verlag 1998

[24] Haberstock, P., Nastanski, L.: Der Einsatz groupwarebasierter Multiprojektmanagement-Systeme im Controlling; Zeitschrift für erfolgsorientierte Unternehmenssteuerung 11(1999)H.10

[25] Hanzelmann, Liskowsky, R., Löscher, S.: Gruppenorientiertes Konfigurationsmanagement auf der Basis von Lotus Notes; Technischer Bericht der TUD, Fakultät Informatik Nr. 04 v. April 1997

[26] Gesellschaft für Prozessrechner Programmierung (GPP): VM-Tailor 2.0; http://www.gppm.de/s_vmtail.htm)

[27] Micro TOOL GmbH Berlin: Whitepaper in-Step. Die Version zum Kennenlernen; 1998 (http://www.microtool.de/inStep/de/index.htm)

[28] Schmidt, W.: Prädikative Spezifikation und Analyse des Vorgehensmodells; GMD-Studie Sankt Augustin, Nov. 1995

[29] Purper, C.B.: GDPA-A Process Web-Centre for the V-Model; Vortrag der GI-FG 5.11 „Vorgehensmodell für die betriebliche Anwendung"; Bonn, März 2000

[30] INNOVATOR unterstützt das KBSt V-Modell Version 97; Whitepaper der MID GmbH Nürnberg (http://www.mid.de)

[31] Axt, H., Hertel, M., Wagner, M.: Lotus Domino & Notes 5; Markt und Technik Verlag München 1999

[32] Liskowsky, R., Pjater, R., Stelter, H.: Gruppenorientiertes Requirement Engineering auf der Basis von Lotus Notes; in Engelien, M., Homann, J.(Hrsg.): Virtuelle Organisationen und Neue Medien; Josef Eul Verlag 1999

[33] Ichim, Iounut: Schaffung einer Notes/Domino-Lösung zur gruppenorientierten Software-Entwicklung; Belegarbeit an der Fakultät Informatik der TU Dresden