Reihe: Telekommunikation @ Mediendienste · Band 11

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. Jens Homann (Hrsg.)

Virtuelle Organisation und Neue Medien 2001

Workshop GeNeMe2001
Gemeinschaften in Neuen Medien

TU Dresden, 27. und 28. September 2001

JOSEF EUL VERLAG
Lohmar · Köln

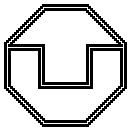
Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Virtuelle Organisation und Neue Medien 2001 / Workshop GeNeMe 2001 – Gemeinschaften in Neuen Medien – TU Dresden, 27. und 28. September 2001. Hrsg.: Martin Engelien ; Jens Homann. – Lohmar ; Köln : Eul, 2001
(Reihe: Telekommunikation und Mediendienste ; Bd. 11)
ISBN 3-89012-891-2

Josef Eul Verlag GmbH
Brandsberg 6
53797 Lohmar
Tel.: 0 22 05 / 90 10 6-6
Fax: 0 22 05 / 90 10 6-88
http://www.eul-verlag.de
info@eul-verlag.de

Printed in Germany
Druck: RSP Köln

Bei der Herstellung unserer Bücher möchten wir die Umwelt schonen. Dieses Buch ist daher auf säurefreiem, 100% chlorfrei gebleichtem, alterungsbeständigem Papier nach DIN 6738 gedruckt.



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

PD Dr.-Ing. habil. Martin Engelien,
Dipl.-Inf. Jens Homann
(Hrsg.)

Dresden, 27./28.09.2001

GeNeMe 2001

Gemeinschaften in Neuen Medien

Workshop zu Organisation, Kooperation und Kommunikation auf der Basis innovativer Technologien

Forum für den Dialog zwischen Wissenschaft und Praxis

an der
Fakultät Informatik der Technischen Universität Dresden

gefördert von der Klaus Tschira Stiftung
gemeinnützige Gesellschaft mit beschränkter Haftung

am 27. und 28. September 2001
in Dresden

http://pdai.inf.tu-dresden.de/geneme
Kontakt: Thomas Müller (tm@pdai.inf.tu-dresden.de)

C.4. Konfigurationsmanagement für Gruppenarbeit

Prof. Dr. Rüdiger Liskowsky,
Dipl.-Inf. Marco Sladek
Fakultät Informatik, Technische Universität Dresden

1. Einleitung

Software-Systeme werden sowohl aus organisatorischen wie auch methodischen Gründen aus Komponenten aufgebaut. Ein sehr wichtiger Grund, der gesamtgesellschaftlich erhebliche Einsparungen mit sich bringt, ist die Wiederverwendung der Softwarebausteine, wie sie bei Hardware-Baugruppen schon lange ausgeprägt ist. Die Aufgabe des Konfigurationsmanagements ist die Verwaltung und Wartung solcher meist großer Software-Systeme, die sich aus Komponenten zusammensetzen. Die Komponenten selbst können in vielfachen Varianten und Versionen existieren, die besonders bezüglich ihrer Schnittstellen und Passfähigkeit zu überwachen sind. Die Aufgabe der Versions- und Konfigurationsverwaltung besteht demnach darin, alle Änderungen zu überwachen und zu jedem Zeitpunkt die Integrität eines Software-Systems sicherzustellen, wobei die Entwicklung der Konfiguration während des gesamten Software-Lebenszyklus verfolgbar sein muss.
Eine derart wichtige Aufgabe sollte im Interesse der ständigen Zugriffsmöglichkeit als Gruppenprozess organisiert werden. Darunter versteht man die Ausführung bzw. Spezifikation von Aktivitäten und Informationen, die von einer Gruppe von Mitgliedern (Rollen) nach definierten Ausführschemata (Gruppenprotokoll) durchgeführt werden. Das Ziel der Gruppenarbeit (CSCW) [1] besteht nun darin, solche dynamischen Systeme zu gestalten, in denen die Gruppenmitglieder flexibel, ohne starre Hierarchie (Lean Management) die Gruppenaufgaben schnell, untergeordnet dem gemeinsamen Ziel einer stets konsistenten Systemkonfiguration, erledigen können.
Der folgende Beitrag ist eine Fortführung der Arbeiten von [2]. Dort wurde ein Konfigurationsmanagement-System vorgestellt, das insbesondere aus Anforderungen der Praxis für die Verwaltung der Softwarebausteine von Notes-Anwendern geschaffen wurde. Mit dieser Ausarbeitung soll stärker auf die Prozessstruktur des Submodells KM aus dem V-Modell des Bundes eingegangen werden [3]. In ähnlichem Umfang setzt sich der „Rational Unified Process (RUP)“ der Firma Rational Software Corporation Cupertino, USA, als firmenorientiertes Vorgehensmodell durch [5]. Das Ziel besteht darin, nicht Lösungen für ein bestimmtes Vorgehensmodell anzubieten. Vielmehr sollen die beiden genannten Modelle analysiert und auf der Basis des detaillierten V-Modells eine Lösung entwickelt werden, die über Tailoring an beliebige praktische

Konfigurationsprozesse anpassbar ist. Dabei ist die Philosophie weiterentwickelter Werkzeuge für das Konfigurationsmanagement, wie z.B. ClearCase als Tool-unterstützung für den RUP mit zu berücksichtigen [6]. Zur Implementierung, die im Rahmen dieser Ausführungen bis zu einem lauffähigen Prototyp zu entwickeln ist, wird wiederum ein Groupware-Produkt der Produktfamilie Lotus gewählt [7]. Unter dem Gesichtspunkt eines von Gruppenmitgliedern einfach auszuführenden Tailoring ist zu entscheiden, ob die jeweilige Entwurfsumgebung von Domino Workflow (Architect) oder Lotus Notes R5 (Designer) eingesetzt wird.

2. Anforderungen an ein gruppenorientiertes Konfigurations-management

2.1 Gruppenarbeit

Als Gruppenarbeit bezeichnet man die Spezifikation bzw. Ausführung von aufgabenbezogenen Tätigkeiten, welche von einer Gruppe von Menschen innerhalb eines festgelegten Rahmens ausgeführt werden. Die Gruppe bedient sich dabei unterstützender Arbeitsmittel und Ressourcen, um seine Gruppenziele zu erreichen [7].
Die bereits mehrfach bewiesenen Vorteile der Gruppenarbeit gegenüber einer technikorientierten, zugeschnittenen Arbeitsgestaltung zeigen sich in:

- Erhöhung der Flexibilität der Gruppenmitglieder
- Qualitative Veränderung der Arbeitszufriedenheit
- Abbau einseitiger Belastungen
- Verminderung der Entwicklungszeiten
- Verbesserung der Produktivität und der Produktqualität.

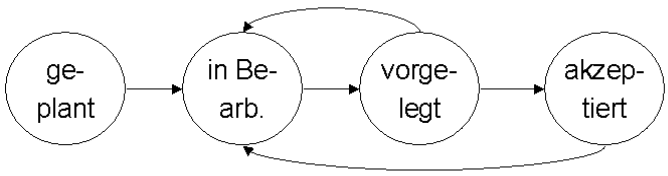
Diese Effekte sind bisher am eindeutigsten bei der Gruppenarbeit in der Autoindustrie wahrgenommen worden [8], lassen sich mittels der flexiblen Komponenten in Gruppenprozessen ebenso auf Softwareentwicklungsprozesse und das Konfigurationsmanagement übertragen [9], [2].
Abgeleitet von den essentiellen Prozesselementen der Softwareentwicklung [10] lassen sich speziell für das Konfigurationsmanagement (KM), wie es im V-Modell unter dem Submodell KM beschrieben ist, folgende Komponenten für die Gruppenarbeit ableiten [11]:

- Statische Komponenten:
  - Gruppenziel: Ständige Konsistenz von Softwaresystemen zu jedem beliebigen Zeitpunkt; globales Ziel, allen persönlichen Zielen übergeordnet
  - Gruppenorganisation: Beschreibung der notwendigen Rollen, verfügbare Mitarbeiterprofile und Ressourcen in der Gruppe

- technisches Gruppenprofil: Informationsaustausch der Gruppenmitglieder, hier vor allem gegeben durch die GUI-Gestaltung des KM-Systems
- soziales Gruppenprotokoll: sozial begründete Verhaltensregeln zwischen den Rollen repräsentierenden Mitarbeitern
- Gruppenumgebung: Räume, Klima, aber vor allem die speziellen für das Konfigurationsmanagement benötigten Werkzeuge und methodischen Hilfsmittel.

♦ Dynamische Komponenten:

- Gruppendokumente sind die einzeln oder gemeinsam bearbeiteten Informationsträger des KM, im V-Modell als Produkte bezeichnet.
- Gruppenaktivitäten stellen die eigentlich im Konfigurationsmanagement auszuführenden Tätigkeiten dar, sowohl mit als auch ohne Werkzeugstützung.
- Gruppensitzungen finden bei asynchroner oder synchroner Ausführung von Aktivitäten statt.
- Gruppenzustand stellt den momentanen Status des Gruppenprozesses dar. Für das KM ist der Zustand der Softwarekomponenten bzw. –dokumente wesentlich signifikanter als der Zustand der Aktivität. In Abhängigkeit von bestimmten Ereignissen werden Zustandswechsel ausgelöst. Im V-Modell festgelegt sind dies die in Abb. 1 gezeigten Zustandsübergänge.



Zustandsbeschreibung:

Geplant:	Eingangszustand für alle Produkte
in Bearb.:	Produkt in „privater“ oder „gruppenorientierter Entwicklungsbibliothek
vorgelegt:	QS übergeben, ohne Mängel --> „akzeptiert“ sonst retour
akzeptiert:	von QS freigegeben --> Abschluß der Version

Abbildung 1: Produkt-Zustandsübergänge im V-Modell

2.2 Analyse des Konfigurationsmanagements auf Basis V-Modell und RUP

Alle im V-Modell [3],[4] enthaltenen Gruppenkomponenten sind summarisch mit dem Produktfluss in Abb. 2 dargestellt.

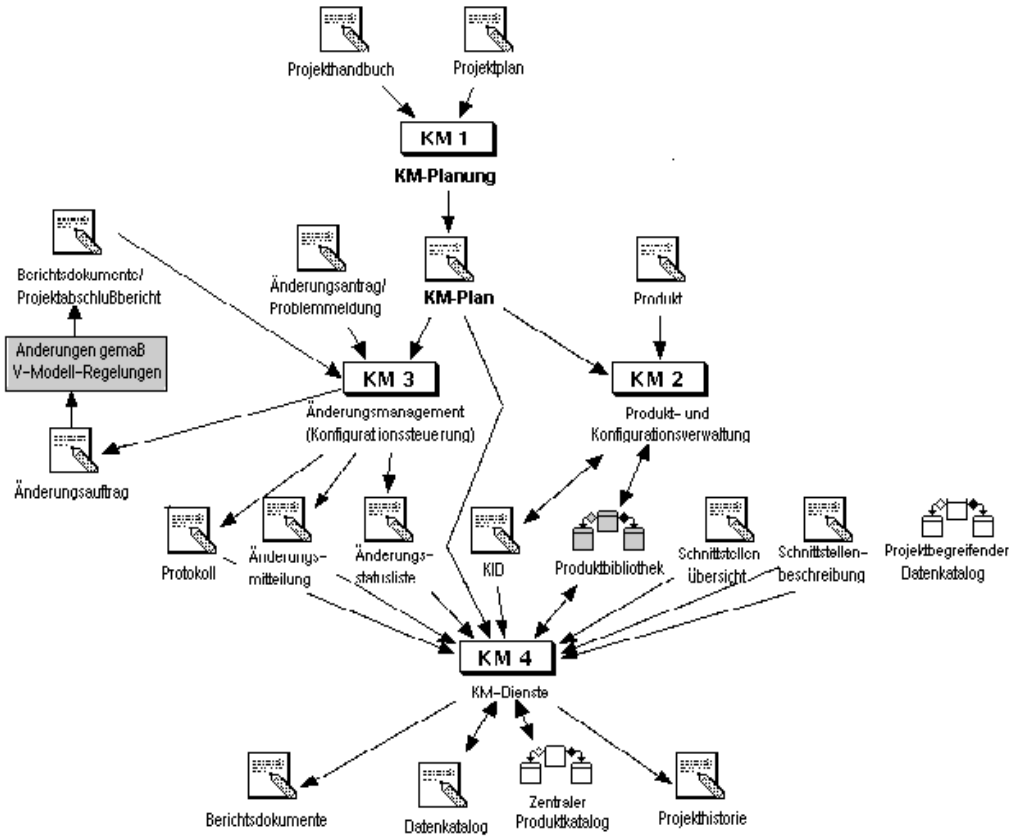


Abbildung 2: KM-Produktfluss basierend auf dem V-Modell nach [4]

Unterschieden werden die vier Hauptaktivitäten:

KM1 - KM-Planung,
KM2 - Produkt- und Konfigurationsverwaltung,
KM3 - Änderungsmanagement,
KM4 - KM-Dienste.

Die wichtigsten benötigten Produkte bzw. Dokumente sind in den Kästchen mit Stift- oder Diagramm-Symbol gezeigt, wobei die der Ergebnisverwaltung dienenden Protokolle und Berichte der Hauptaktivität KM4 innerhalb dieser Betrachtungen zunächst unberücksichtigt bleiben. Jede Hauptaktivität spaltet sich in untergeordnete Aktivitäten mit internem Produktfluss auf, von denen beispielhaft die KM-Planung in Abb. 3 gezeigt ist (hier entsprechen Ellipsen den Produkten).

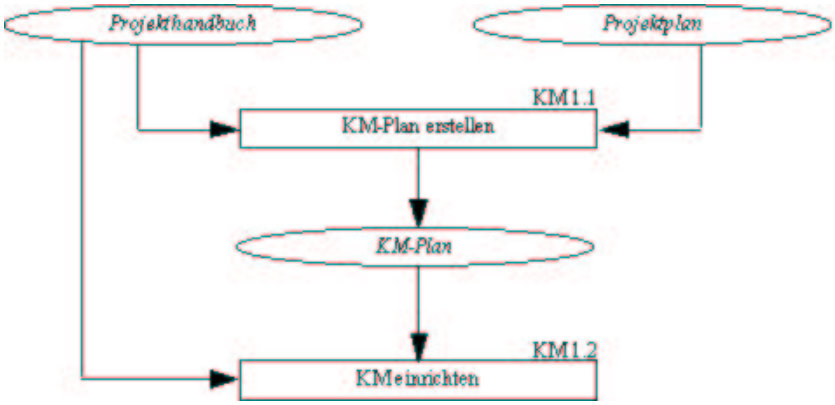


Abbildung 3: Aktivitäten der KM-Planung nach [3]

Für alle weiteren Hauptaktivitäten existieren analoge Untergliederungen einschließlich ausführlichen Beschreibungen für Subaktivitäten, Produkten und ausführenden Rollen [3],[4]. Generell stellen die Elemente des V-Modells eine sehr gute Grundlage für die Gestaltung von Softwareprozessen dar. Der an vielen Stellen fehlende inhaltliche und methodische Tiefgang lässt auf der anderen Seite viele Freiheitsgrade für die Realisierung der Lösung offen.

Den prinzipiellen Arbeitsablauf bzw. Workflow im RUP für das Teilmodell KM zeigt Abb. 4. Dieses Vorgehensmodell ist nicht vollkommen werkzeugneutral [12] sondern lehnt sich stark an die von der Firma Rational Software Corporation gelieferten Tools an, im Falle von KM an ClearCase [6].

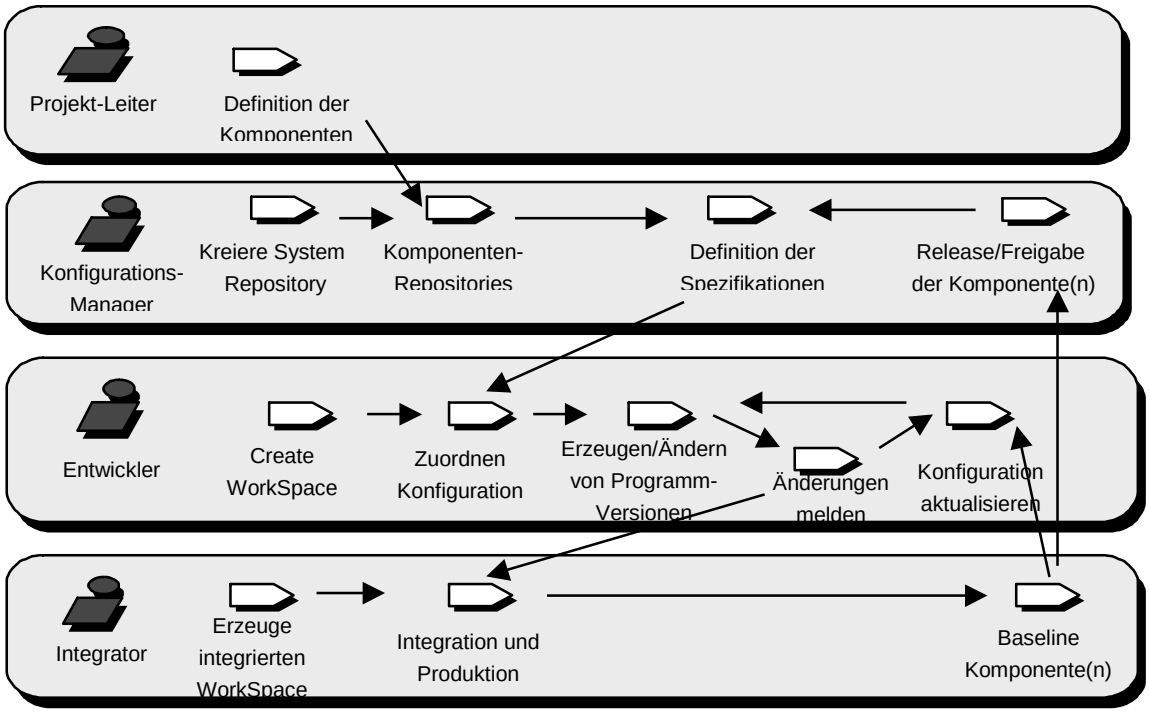


Abbildung 4 KM-Prozess als Workflow im RUP

Auch dieses Vorgehensmodell besteht aus Rollen, die einzelne Aktivitäten ausführen und einem zugehörigen Workflow. Eine direkte Nomenklatur für Produkte, die zwischen den Aktivitäten ausgetauscht und als Ergebnisdokumente abgespeichert werden, kann man nicht erkennen. Beide Modelle enthalten eindeutige Begriffsdefinitionen für das KM, die hier gewissermaßen als Lexikon für die wichtigsten Anforderungen miteinander abgestimmt wiedergegeben werden sollen (Tabelle 1):

Begriff	Semantik
Konfiguration	benannte und formal freigegebene Menge von versionierten Objekt-Basen (VOB) für eine widerspruchsfreie Abarbeitung
Version	Instanz bzw. Entwicklungsstand einer Komponente, je Editierung nach dem Zustand „akzeptiert“ erhöht sich Versions-Nr. um 1
Komponente	Softwarebaustein oder VOB in der Mini-Welt von Anwendungsprojekten (im V-Modell auch Softwareeinheit)
Baseline	spezifizierte Menge an Komponenten, VOB und Produkten, die einen gesicherten Arbeitsstand für die Weiterarbeit bezeichnen
Variante	Modifikation einer Version ohne Änderung des Versionsstandes
Release	freigegebene konsistente Menge von Softwareeinheiten
Projekt	realisiertes (Software-)Vorhaben, resultierend aus Anforderungen/Zielen und objektspezifischen Bedingungen
Spezifikation	auf konkretem Projekt beruhendes spezifisches Anwendungsprofil für einen Auftraggeber
System	Einheitliches Ganzes (Hardware, Software, Prozesse), das die Fähigkeit besitzt, Forderungen zu erfüllen, z. B. Komponentenbibliothek

Tabelle 1: Begriffslexikon zum Konfigurationsmanagement

2.3 Anforderungen an die zu schaffende KM-Groupwarelösung

Aus der vorangegangenen Untersuchung der Funktionen des KM anhand der beiden abstrakten Vorgehensmodelle sollen nun stichpunktartig die Anforderungen an die KM-Lösung für Gruppenarbeit aufgelistet werden.

Zunächst soll sowohl die Konfigurationsverwaltung als auch die Versionsverwaltung von mehreren Gruppenmitgliedern in einem Gruppenprozess ausgeführt werden können. Damit soll sich auch die Gruppenorganisation während des Prozesses entsprechend dem Gruppenziel ändern können und auch die dynamischen Komponenten können Änderungen in der Zuordnung der Rollen unterliegen.
In der Versionsverwaltung geht es darum:

- Versionen von Komponenten anzulegen, zu identifizieren und alle zugehörigen Bestandteile zu verwalten,
- Namenskonventionen und Relationen zwischen den VOB zu finden,
- Bilden und Kontrollieren der Baseline (Versions-Hauptlinie),
- Dokumentieren von Änderungen,
- Verwalten der System- bzw. Komponenten-Repositories,
- Festlegen und Überwachen von Zugriffsrechten.

Hauptaufgabe der Konfigurationsverwaltung ist die Wahrung der Systemintegrität, d.h. der Zustand eines Systems als Gesamtheit der benötigten Unterlagen muss zu jedem Zeitpunkt eindeutig beschreibbar sein. Dazu gehören auch die Beschreibbarkeit von Spezifikationen bis hin zu Dokumentations- und Werbematerial. Im einzelnen bedeutet dies:

- Definition und Identifizierung der Komponenten einer Konfiguration,
- Sicherung der Vollständigkeit und Korrektheit aller Konfigurationskomponenten,
- Kontrolle der Freigabe und aller Änderungen der Konfigurationskomponenten während des gesamten Lebenszyklus,
- Protokollierung und Erstellung von Berichten zum Status der Komponenten und Änderungsanforderungen.

Die beiden letzten Anstriche unterstreichen die enge Verbindung zum Änderungsmanagement KM3. In Abwandlung zum V-Modell soll das Fehler- und Änderungswesen entsprechend Forderungen der Praxis, wie in Abb. 5 dargestellt, realisiert werden. Um den Eigenschaften des Gruppenprozesses entgegenzukommen, werden den Aktivitäten keine (festen) Rollen zugeordnet.

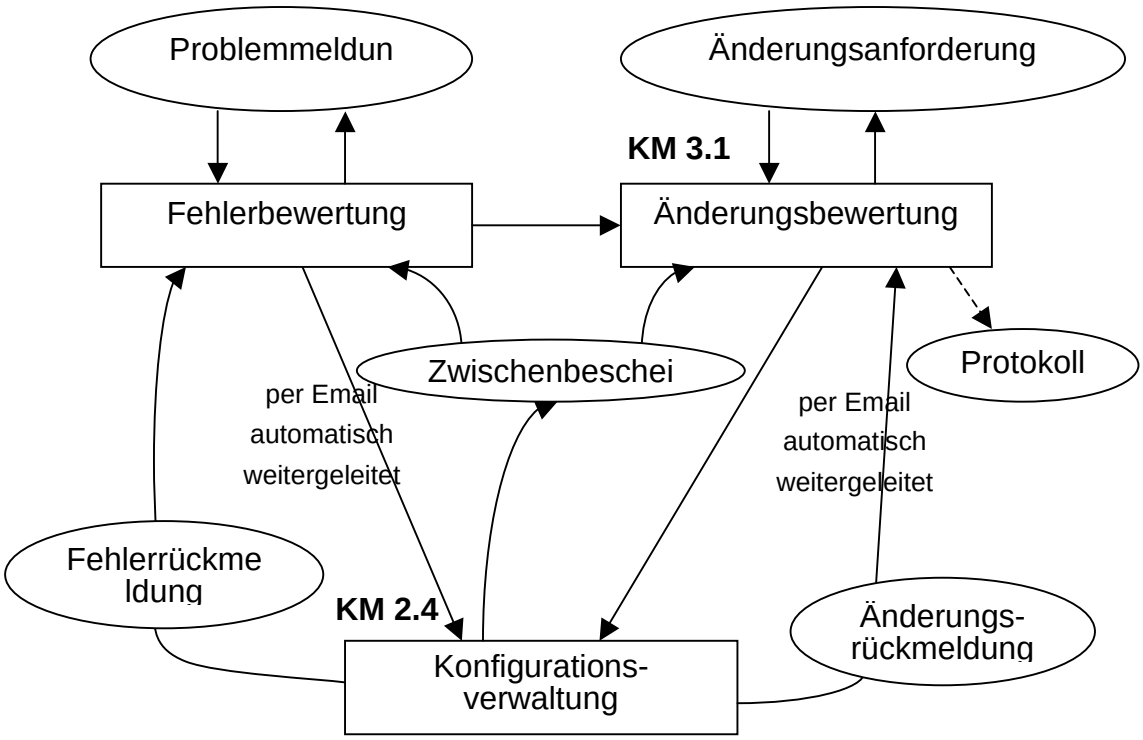


Abbildung 5: Fehler- und Änderungswesen in Anlehnung an das V-Modell

Abschließend seien alle Anforderungen für die gruppenorientierte KM-Lösung in einer Tabelle der zu realisierenden Gruppenprodukte im Sinne des V-Modells zusammengestellt (Tabelle 2):

Gruppenprodukt	**Funktion**
Projektblatt	allgemeine Beschreibung des zu realisierenden bzw. zu wartenden (Software-)Vorhabens
Versionsblatt	eindeutige Versionskennzeichnung mit enthaltenen Komponenten und Zuordnung zum Vorhaben
Spezifikationsblatt	an Kundenwünsche angepasster Auslieferungszustand eines (Software-)Vorhabens
Informationsmaterial	Informations- und Werbematerial zugeordnet zu Vorhaben, Spezifikation und/oder Version
Komponentenblatt	Beschreibung einer Software-Komponente mit Status und Zuordnung enthaltener Komponenten
Problemmeldung	Meldung eines Fehlers mit ID und geschätzter Dringlichkeit der Behebung
Änderungsanforderung	Anforderung einer Fehlerbehebung und Dringlichkeit der Änderung
Änderungsrückmeldung	Rückmeldung über erfolgreiche Lösung des Änderungs-

Gruppenprodukt	Funktion
	antrages und seine Beschreibung
Fehlerrückmeldung	Rückmeldung zur weiteren Bearbeitung und Behebung des Fehlers
Zwischenbescheid	sofortiges Feedback über registrierten Eingang einer Fehlermeldung oder einen Änderungsantrag über Mail
Protokoll	schriftl. Bericht über Verlauf und Resultate der Änderungsbearbeitung

Tabelle 2: Produkte/Dokumente der KM-Lösung

Auf eine Darstellung der erzeugenden bzw. verbrauchenden Aktivität (Produktfluss) muss an dieser Stelle verzichtet werden. In den nachfolgenden Abschnitten wird auszugsweise darauf eingegangen.

3. Objektorientierter Entwurf des Konfigurationsmanagements (KM)

War für den Entwurf eines gruppenorientierten Systems in [2] noch das datenorientierte Vorgehen mittels Entity-Relationship-Modell dominant, so wird im folgenden konsequent objektorientiert in Anlehnung an [13] vorgegangen. Eine zusätzliche Prämisse ist die Verwendung von Notes-Elementen[14], die aber in keiner Weise eine Beeinträchtigung der Allgemeingültigkeit darstellt. Trotz der konsequent Datenbank-orientierten Struktur des späteren Notes-Anwendersystems und den nicht immer vollständig objektorientierten Programmierelementen wird an dieser Stelle zunächst ein objektorientierter Entwurf zum KM vorgestellt. Mit der Definition der benötigten Klassen fördert er eindeutig die Umsetzung der eingeführten Begriffe sowie der zugehörigen Gruppenkomponenten. Die aufgefundenen Beziehungen zwischen den Klassen stellen eine Fortschreibung der Relationen des KM Entity-Relationship-Modells [2] dar. Die Mittel der UML [13], insbesondere die Assoziationsspezifikation unter Angabe der Rollen beteiligter Klassen, sind für die beabsichtigte Modellierung sehr wertvoll.

Abgeleitet aus dem V-Modell und dem RUP-Prozess-Modell [12] werden im praxisnah angepaßten (tailorisierten) KM-Workflow folgende **Rollen** berücksichtigt:

- Projekt-Leiter,
- KM-Administrator,
- KM-Verantwortlicher,
- Entwickler,

- Prüfer und
- Anwender (Meldender).

Der Anwender als Meldender wurde der Vollständigkeit und Anschaulichkeit halber eingeführt, im Workflow aber nicht aktiv realisiert, weil er nur von außen auf den Prozess einwirkt. Alle anderen Team-Mitglieder-Rollen werden als Verantwortung tragend im KM-Prozess behandelt. Die ausführliche Rollenbeschreibung kann zum Beispiel [3] entnommen werden. Von der Gruppe ist zu überlegen, wie diese Rollen von den einzelnen Team-Mitgliedern ausgefüllt werden. Das System gestattet die Möglichkeit zur Einrichtung dieser Rollen, ihre konkrete Besetzung mit Gruppen-Mitgliedern, ihre Interaktion untereinander sowie die Zuweisung aller gewünschten Zugriffsrechte zu den Dokumenten. Die Rollen der Team-Mitglieder werden über die Klasse **Person** und ihre Assoziationen modelliert.

Die restlichen Klassen stellen die notwendigen zu bearbeitenden **Produkte/Dokumente** im KM-Gruppen-Prozess ausgehend vom V-Modells dar (s. Tabelle 2):

- Fehler (in der Eigenschaft als **Problemmeldungs-Formular**)
- Änderung (in der Eigenschaft als **Änderungsanforderungs-Formular**),
- Werbematerial (allgemein **Informationsmaterial**),
- Komponente (zu bearbeitendes System-Dokument/**Komponentenblatt**),
- Version (als **Versionsblatt**),
- Spezifikation (als **Spezifikationsblatt**),
- Projekt (als **Projektblatt**) und
- Anhang (erklärende beizufügende Dokumente für Annotationen).

Die weiteren, in Tabelle 2 enthaltenen Dokumente, werden in der Phase des Grob-Entwurfs erstmal nicht modelliert. Sie werden später nach gleichen Grundsätzen wie die übrigen Formulare in den Prototyp übernommen.

In dem folgenden Klassen-Diagramm (Abb. 6) wurde speziell auf die Ausgestaltung der Assoziationen zum Verständnis des Rollenverhaltens Wert gelegt. Damit steht die Klasse Person im Zentrum des Diagramms. Über die Rollen-Funktionen repräsentiert sie, wie die Gruppenmitglieder mit den anderen Objekten agieren. Von untergeordneter Bedeutung für den Gruppenprozess sind die Rollen bzw. die assoziierten Klassen, die nur für von Gruppenmitgliedern bearbeitete Produkte/Dokumente stehen. In Abbildung 6 sind nur die Hauptbeziehungen zwischen den tragenden Klassen dargestellt, die gegenseitige Referenzierungen benötigen.

Die augenscheinlich nötigen Attribute und Methoden-Namen sind bereits eingetragen. Das Attribut "Rollen_Liste" der Klasse Person sei besonders erwähnt, da über dieses alle zu bekleidenden Rollen eingetragen werden. Durch Einstellungen des KM-

Administrators sind sie jederzeit modifizierbar. Damit wird die notwendige Flexibilität und freie Ausgestaltung des Gruppen-Prozesses gesichert.

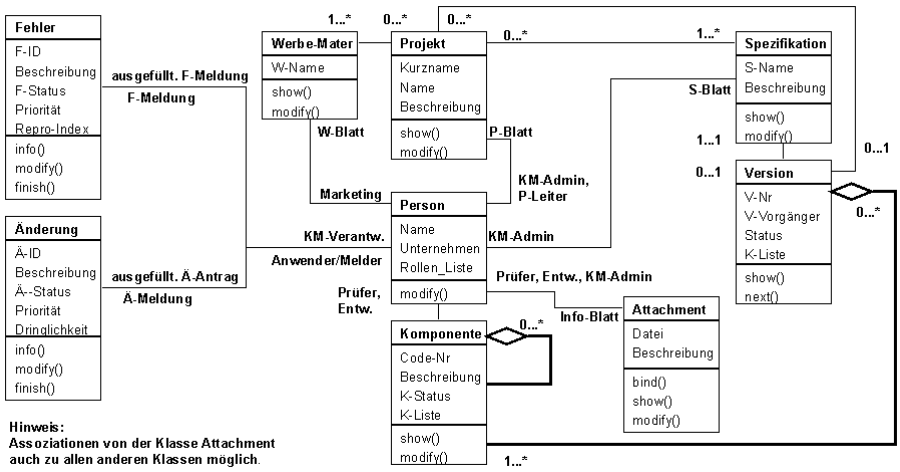


Abbildung 6: Klassendiagramm des Fehler- und Änderungsmanagements mit Rollenangaben

Die Ausgestaltung der eigentlichen Änderungsaktivitäten aufgrund der Problem-meldungen und Änderungsanforderungen in bezug auf die (Software-) Komponenten sowie die sich daraus ergebenden Auswirkungen auf das Software-Produkt sind hauptsächlich Inhalt der Methoden ***modify()***, weshalb sie in Abb. 6 nicht darstellbar sind. Dazu wurden hauptsächlich dynamische Zustandsübergangsdiagramme bezogen auf die KM-Produkte genutzt.

4. Prototyp-Implementierung mit Domino Workflow (DWF)

Basierend auf [2] soll hier ein Prototyp unter Einbeziehung der besonderen, von Domino Workflow [14] bereitgestellten Entwurfsobjekte und Funktionalität vorgestellt werden. Dazu war zunächst die KM-Lösung von [2] in Hinblick auf die Verwertbarkeit einzelner Implementierungsbestandteile bezüglich der Notwendigkeit einer Erweiterung im Sinne des neuen objektorientierten Entwurfs gegenüber dem alten rein datenmodellierten ERD zu prüfen.

4.1 Grundstruktur des Produktes Domino Workflow

DWF basiert auf der Lotus Notes Architektur und dient aufgrund seines Arrangements von Datenbanken spezieller Struktur und dazu bereitgestellter Funktionalität der Modellierung, Installation, Verwaltung und Veranschaulichung von Arbeitsabläufen sowie der Gewährleistung von deren Betrieb zur Laufzeit. Für den **Anwender** erscheint dies als Anwendung (weitgehend Notes gestützt) aus Dokumenten und darüber

agierenden Aktivitäten mit zugeordneten Team-Mitglieder-Rollen, getriggert durch die Nutzung des Notes eigenen Mail-Systems sowie durch Anwender-Aktionen zur Dokumentenbearbeitung via Notes- oder Web-Client (Browser).

Dem **Prozessverantwortlichen** stellt sich DWF grundlegend als Workflow gestaltende Anwendung dar, mit dem Prozess-Designer (DWF Architect) als zentralem Element mit grafischem Editor für die Prozess-Abläufe, samt mitgelieferter zugehöriger Notes-Schablonen für den Prozess-Entwurf. Weiterhin steht ihm ein Prozess-Betrachter (DWF Viewer) für die statische und dynamische Anzeige von initiierten Vorgängen (basierend auf vorher entworfenen, geprüften und aktivierten Prozessen) zur Verfügung.

Die Windows-Anwendungen DWF-Architect und DWF-Viewer arbeiten eng mit den weiteren wesentlichen DWF-Komponenten zusammen, etwa spezielle miteinander wechselwirkende Notes-Datenbanken.

Vertreter der enthaltenen Datenbanken können über spezielle DWF-Profile gesondert verwaltet werden.

Die geschilderte DWF-Tool-Architektur ist in Abb. 7 grafisch veranschaulicht (Datenbanken und Zugriffsmöglicheiten, eingefaßt in die DWF-Komponentenstruktur, s.a. [16]).

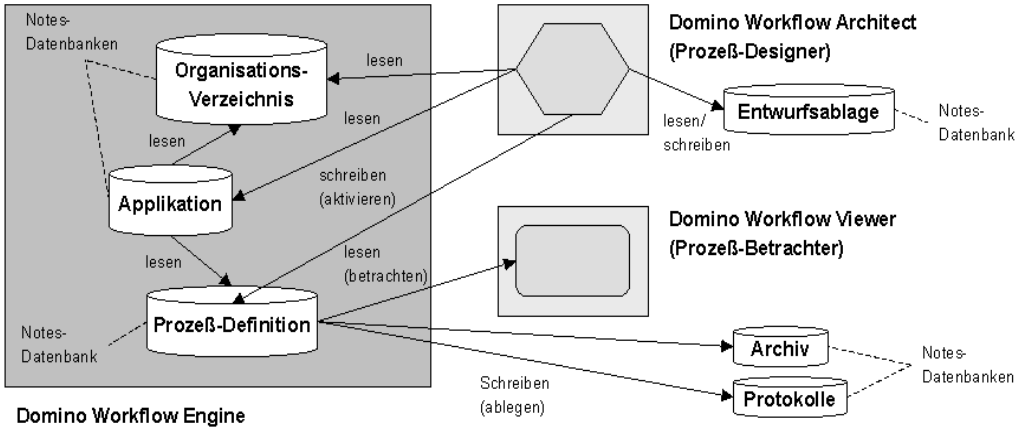


Abbildung 7: Domino Workflow - Tool-Architektur

4.2 Prototyping mit dem DWF

Die durch DWF praktizierte Art und Weise des Prototyping ist am Umgang mit dem jeweils zu gestaltenden Workflow zu erkennen. Eine zu gestaltende Anwendung auf Lotus Notes Basis kann unter DWF nur im Zusammenhang mit der Gestaltung eines zugehörigen Workflows gesehen werden. Die beiden ersten nötigen Datenbanken Organisationsverzeichnis und Applikations-DB (Abb. 7) werden über von DWF angebotene Schablonen erzeugt und mit den gewünschten Dokumenten-Inhalten gefüllt, wobei wesentliche Daten für erstere z.B. aus dem öffentlichen Namens- und

Adressbuch übernommen werden können. Die Ausgestaltung der im Team vorliegenden Organisationsstruktur geschieht über die Zuordnung der beteiligten Rollen zu den Verwaltungseinheiten. In der **Applikationsdatenbank** werden alle für die Anwendung zu benutzenden Masken, Ansichten und zugehörigen Formeln untergebracht, bei Beachtung der Besonderheiten der konkurrierenden WF-Steuerung durch DWF.
Die letztlich nötige explizit zu füllende Prozess-Definitions-Datenbank beherbergt und verwaltet die Prozess-Definitionen nach ihrer erfolgreichen grafischen Gestaltung im Prozess-Designer (DWF Architect). Dieser legt alle noch nicht fertig gestalteten und positiv geprüften Prozesse zunächst in der Entwurfsablage ab. Über ihn sind die syntaktisch korrekten Prozesse aktivierbar, um dann durch ein Team-Mitglied via Applikationsdatenbank als Instanz (Vorgang) initiiert werden zu können. Auch im Prozess-Designer sind beim Workflow-Entwurf die Besonderheiten der zukünftigen Applikation zu berücksichtigen.
Die eigentliche **prototypische Entwicklung mit DWF** kann nur stattfinden, wenn die nötigen Datenbanken grundlegend initialisiert und mit dem Minimal-Anwendungs-datenbestand versehen wurden. So sollten die Anwendungsmasken auf Seite der Applikationsdatenbank und mindestens eine syntaktisch korrekte Prozess-Definition in der entsprechenden Datenbank vorhanden sein, d.h., auf letzterer basierender Prozess muss aktiviert und zugehöriger Vorgang initiiert werden. Ab diesem Entwurfsstadium kann durch Weiterentwicklung schrittweise erneut ein solcher konsistenter Zustand (etwa nur ein neuer oder korrigierter Workflow) erreicht werden, was die bisherigen installierten Workflows und darüber agierende Anwendungsdaten nur ansatzweise beeinflussen braucht. Ausschließliche Änderungen an existierenden Workflows (etwa via "Prozess-Designer"-Rolle, d.h. KM-Verantwortlicher bzw. -Manager) stellen zudem eine Art des Betriebes einer DWF-Anwendung dar.
D.h., zu jedem solchen Stadium ist die jeweils entstandene DWF-Anwendung zugleich einer neuen zugrunde liegenden Nutzer-Anforderung gerecht geworden und ebenso ein lauffähiger Prototyp zur Testung, ggf. ausreichend für den andauernden laufenden Betrieb oder aber Ausgangspunkt zur Weiterentwicklung. Diese Bedingung traf i.a. für Lotus Notes Anwendungen schon zu und muss daher für darauf aufsetzende An-wendungen zumindest bedingt ebenso gelten.

4.3 Prämissen der bestehenden KM-Lösung

Die in [2] aufgezählten, genutzten Besonderheiten von Lotus Notes/Domino als Basis der KM-Implementierung werden hier nicht noch einmal aufgezählt. Genauso werden die grundlegenden Anwendungsstrukturmerkmale, schon um die Prototyp-Lösung dem

Vorgängersystem anzupassen, im wesentlichen übernommen. Das betrifft vor allem die grundlegende Datenbank-Struktur mit den Einzeldatenbanken:

- Konfigurationseinheiten (Komponenten-DB),
- Konfigurationsverwaltung-DB und
- Fehler- und Änderungsmanagement-DB.

Die aufgeführten DB sind Notes-Datenbanken, d.h. sie können unterschiedlich strukturierte Dokumente beinhalten, weshalb sie aber nicht in Widerspruch zu dem Klassenentwurf von Abb. 6 geraten. Lediglich die Werkzeuganbindung und die Funktionen zur Dokumentation sowie die eigentliche Versionsverwaltung der Softwarekomponenten sind in einem neuen Zusammenhang zu sehen. Die in [2] aufgeführte Begrifflichkeit Produkt für das Top-Objekt einer Entwicklung wird im weiteren als **Projekt** bezeichnet. Während in [2] mehr auf die reine konsistente Verwaltbarkeit der beteiligten Komponenten und Dokumente Wert gelegt wurde, stehen bei der Weiterentwicklung Aspekte einer flexiblen Gruppenarbeit und deren Steuerbarkeit im Mittelpunkt. So sind in [2] zwar Bearbeitungszustände und Versionen von Produkten/Dokumenten für das KM implementiert, aber die Voraussetzungen für gleichzeitige Arbeit und variable Rollenverteilung nicht. Die Ausgestaltung des Änderungswesens berücksichtigt Elemente des zugehörigen Workflows als fest installierte Abläufe, ohne diese explizit zu nennen. Dies gestattet im Ergebnis nur eine grundlegende, auf der speziellen Ablauf-Charakteristik des Prozesses aufsetzende passive Arbeitsteilung. Die Modellierung des Gruppenprozesses an sich, als auch die Betrachtung und Flexibilität der beteiligten Rollen war in [2] nicht von ausschlaggebender Bedeutung. Hier soll die neue, auszugestaltende Prototyp-Lösung aufsetzen. Der Schwerpunkt liegt auf der asynchronen Arbeit der Gruppenmitglieder, die für reale KM-Prozesse charakteristisch ist. Deshalb wird die synchrone, echt gleichzeitige Arbeit der Team-Mitglieder nicht realisiert, ebensowenig wie die eigentliche werkzeuggestützte Änderungsarbeit an den Software-Komponenten, die von CASE-Tools durchgeführt werden sollte. Die Datensicherung ist mittels des Notes-Datenbanken-Managements ausreichend abgedeckt [14]. Konsistenz-Bedingungen sind via Formelkonstruktionen gut prüfbar und ggf. auch automatisch herstellbar. Der Informationsaustausch der Team-Mitglieder ist unter Nutzung von Notes-Mail leicht installierbar.

Daher sind also im Ergebnis:

- die jeweils vorhandene Organisationsstruktur in die Rollenstruktur zu integrieren,
- die Wechsel der Rollen-Zugriffsrechte in Abhängigkeit von den Zustandswechseln der zu bearbeitenden Produkte/Dokumente auszugestalten (s. Tabelle 2),

- eine Informationsmöglichkeit für die beteiligten Rollen über die Art und Stelle ihrer Einbindung im Gesamtprozess zu schaffen, einschließlich der jeweils rechtzeitigen Mitteilung über die ihnen zugeordneten folgend nötigen aktuellen Aktivitäten samt zugehörigen Dokumente,
- die Gestaltbarkeit und Installation des Gruppenprozesses (etwa Werkzeug DWF-Prozess-Designer nutzbar, durch Rolle des KM-Verantwortlichen, als eine Art „Super“-Administrator durchzuführen) mit seinen Elementen zu sichern, etwa via der Modellierung der Äktivitäten-Abfolgen(Workflow) mit den jeweils zu nutzenden Dokumenten,
- der Start, die Abarbeitung und ggf. zwischenzeitliche Stoppung des Workflows samt sinnvoller Triggerung (z.B. via Notes-Mail und Formelkonstrukten) abzusichern, etwa in einer extra Verwaltung, als extra Datenbank-Anwendung,
- die beteiligten Dokumente geeignet zu verwalten für Gruppenprozess Modellierungs- Zwecke.

Allerdings sind die DWF-eigenen Besonderheiten, etwa Einschränkungen bzgl. der dann jeweils installierten Workflows gegen die im Ergebnis noch zu erfüllenden Anforderungen abzuwägen, bzw. ein wirksames und simples "Work around" aufzubauen.

4.4 Konstruktion des KM-Prototypen für Gruppenarbeit

Die grundlegende DB-Architektur des KM, aufbauend auf der Arbeit von [2] und [15], ist in der folgenden Abb. 8 zu sehen.

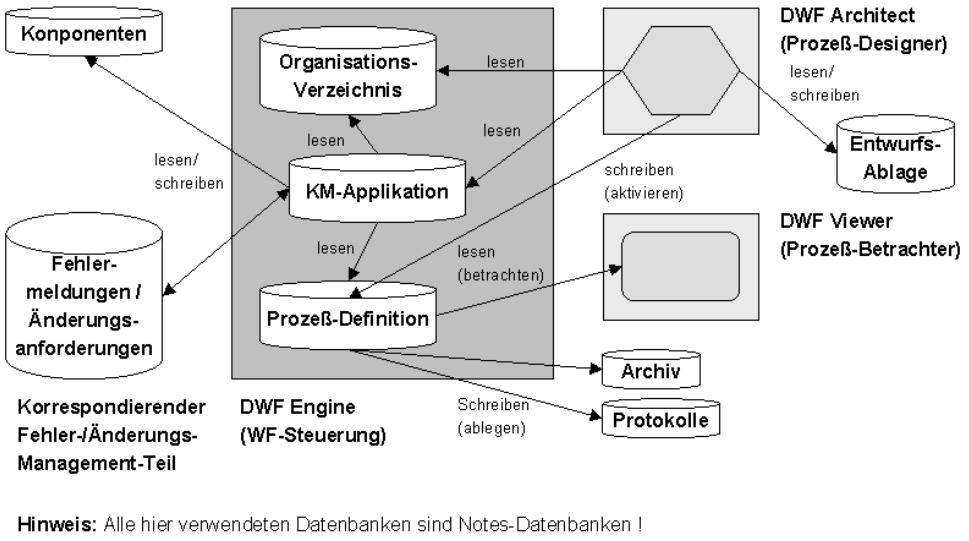


Abbildung 8: Prototyp-DB-Architektur zur KM-Lösung für Gruppenarbeit

Sehr deutlich treten zunächst in Abb. 8 die DWF-Komponenten hervor. Eine spezielle Anpassung in Hinblick auf das KM ist mit der Gestaltung der Applikationsdatenbank getätigt worden. Alle wesentlichen Datenbank-Besonderheiten aus der KM-Datenbank von [2] finden dort Platz. Daher steht im Applikationsteil in der Grafik auch KM-Applikation. Die Spezifik des Fehler- und Änderungsmanagements in der Ziel-Anwendung wird in Abb. 8 durch eine separate Datenbank verdeutlicht. Sie interagiert über ihre anwendungsspezifische Funktionalität mit dem meldenden Anwender und realisiert die Verwaltung der dabei gewonnenen Daten. Die Schnittstelle zur eigentlichen KM-Applikation stellt ihr die zur Verwaltung im Rahmen des KM nötigen Daten in der gewünschten Form zur Verfügung und nimmt die aus den Änderungsvorgängen resultierenden Rückmeldungen entgegen.
Die Herausstellung der Komponenten-Datenbank verdeutlicht die separate Verwaltung der Komponenten-Daten und den ändernden oder lesenden Zugriff von der Konfigurationsverwaltung aus (im Rahmen von Änderungen).

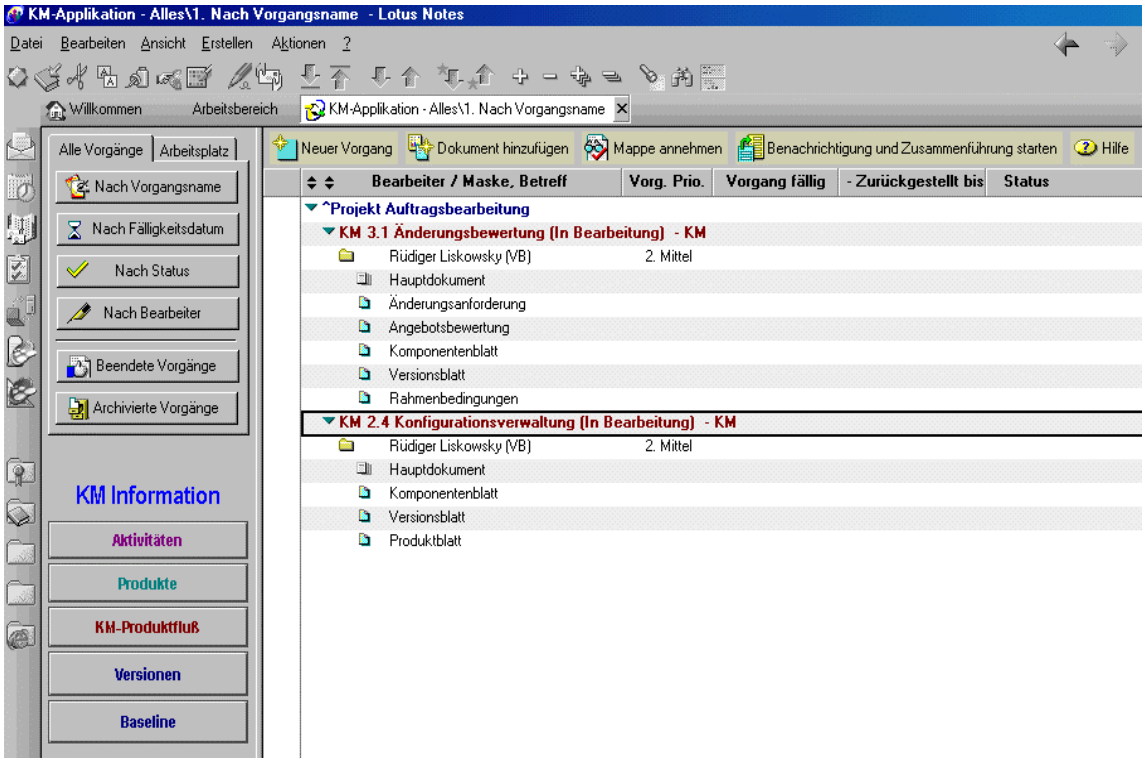


Abbildung 9: Ansicht des Prototypen zu Aktivitäten, Personen und Dokumenten

Zur Veranschaulichung soll an dieser Stelle exemplarisch auf eine realisierte Ansicht des Prototypen eingegangen werden (Abb. 9), welche einen Einblick in die prototypisch modellierten KM-Vorgänge KM 2.4 und KM 3.1 gewährt. Der linke Bildteil beinhaltet die Möglichkeiten zur Navigation (Ansichten: Aktivitäten, Produkte, Versionen, Baseline mittels Lotus Notes Client sowie Anzeige: aktueller Stand des KM-Produktflusses/-

Workflows via DWF). Der Anzeigebereich im mittleren Teil listet die beteiligten Produkte/Dokumente, geordnet nach Projekt, Aktivität und Verantwortlichem.
In jeder Ansicht lässt sich das zugehörige aktuelle Dokument aufrufen und je nach Berechtigung können die zugeordneten Versionsdaten geändert werden. Quasi lässt sich jede Klasse nach Abb. 6 in ein entsprechendes Notes-Dokument umsetzen. Die Eigenschaften der Klasse Person finden sich in den Personendokumenten bzw. Zugriffskontroll-Listen wieder.
Sollten sich alle Anforderungen aus Abschn. 2 nicht zweckdienlich mit DWF erfüllen lassen, ist die gewählte Modellierungsvariante zunächst weiter mit Mitteln von Lotus Notes zu ergänzen, ehe anders geartete Implementierungen eingesetzt werden. Dabei hat eine weitgehende Anlehnung an die doch sehr innovative Lösung von DWF Priorität.
Im Prozess der schrittweisen Vervollkommnung des Prototypen wird sich die "beste" Lösung für das Konfigurationsmanagement herauskristallisieren.

5. Zusammenfassung und Ausblick

Ausgehend von einer datenorientierten Betrachtungsweise des Konfigurations-managements [2] wurde in dem Beitrag zur objektorientierten Modellierung aller am Prozess beteiligten Personen, Aktivitäten und Dokumente übergegangen. Die Vorlage dafür lieferten das V-Modell des Bundes und der Rational Unified Process (RUP). Wie gezeigt wurde, liegt ein Vorteil der abstrakten Modellierung darin, dass all-gemeingültige Gesetzmäßigkeiten erfasst werden können, ohne sich im Detail festlegen zu müssen. Dieser Umstand wurde besonders für die Erfassung und Flexibilität von Gruppenprozessen genutzt. Insbesondere die dynamischen Komponenten der Gruppen-arbeit sind so nachzubilden, dass alle beteiligten, verteilten Gruppenmitglieder das Ziel einer zu jeder Zeit konsistenten Projektkonfiguration erreichen können. Damit fallen auch die Unterschiede zwischen Vorgehensmodellen der Softwareentwicklung nicht mehr so stark ins Gewicht, in dem die EDV-Lösung eine Anpassung an die praktisch günstigsten Arbeitsabläufe zulässt.
Zur flexiblen Implementierung des Klassenmodells hat sich Domino Workflow be-währt. Hier konnten auch Erfahrungen früherer Arbeiten [15] nutzbringend verwertet werden. Außerdem sind die Verbindungen interessant, die zu dem dort implementierten Modell der Softwareentwicklung existieren. Zunächst wurde ein lauffähiger Prototyp für die Gruppenarbeit im Konfigurationsmanagement geschaffen, der KM-Aktivitäten mit zugeordneten Verantwortlichen (Rollen), Aufgaben, Dokumenten und nach Wunsch mit einer Zeitplanung steuert. Damit sind die Voraussetzungen für einen praktischen Einsatz der EDV-Lösung gegeben. Aufgrund der Verwendung von DWF lassen sich Änderungen im laufenden Betrieb und notwendige Erweiterungen leicht realisieren.

Neben der Vervollkommnung der Lösung in der praktischen Erprobung werden Weiterentwicklungsmöglichkeiten im Zusammenspiel mit Tools gesehen, die für spezielle Aktivitäten des KM zum Einsatz kommen. Beispiele dafür sind die Überwachung von Verknüpfungs-(Make-)Files für Versionen als auch die Versionenablage auf der Basis von Differenzen, um nur zwei Beispiele zu nennen. Hier gewinnt die Unterstützung standardisierter Austauschformate zur Ermöglichung des effektiven Zusammenspiels mit speziellen CASE-Tools für Konfigurationsmanagement an Bedeutung. Eine Möglichkeit besteht in der Unterstützung des XMI-Formates, das in Form des allgemeineren XML auch in Lotus Notes verarbeitbar ist.
Generell stellt die Weiterentwicklung des Domino Workflow basierten Prototypen eine gute Basis für die Einführung der Gruppenarbeit auf diesem verwaltungstechnisch geprägten Subsystem der Softwareentwicklung dar.

6. Literatur

[1] Rüdebusch, T.: CSCW-Generische Unterstützung von Teamarbeit in verteilten DV-Systemen; Deutscher Universitäts-Verlag Wiesbaden 1993

[2] Hanzelmann, D., Liskowsky, R., Löscher, S.: Gruppenorientiertes Konfigurationsmanagement auf Basis von Lotus Notes; Technischer Bericht TU Dresden, Fakultät Informatik TUD/F/97/04 v. April 1997
URL: http://www.inf.tu-dresden.de/pup/bedrichte/tud97-04.ps.gz

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

[4] 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; URL: http://www.informatik.uni-bremen.de/gdpa

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

[6] Rational Software: Unified Change Management from Rational Software: An Activity – Based Process for Managing Change; Rational Software White Paper; URL: http://www.rational.com/products/clearcase/whitepapers.jsp

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

[8] Ulich, E.: Gruppenarbeit – arbeitspsychologische Konzepte und Beispiele; in Friedrich, J., Rödiger, K.H. (Hrsg.): Computergestützte Gruppenarbeit (CSCW); Teubner-Verlag 1991, S. 57-78

[9] Altmann, J.: Kooperative Softwareentwicklung – Rechnerunterstützte Koordination und Kooperation in Softwareprojekten; Diss. Universität Linz; Universitätsverlag Rudolf Trauner 1999

[10] Liskowsky, R., Pjater, R.: Modellierung gruppenorientierter Prozesse mit Notes/ Domino; in Engelien, M., Neumann, D.(Hrsg.): Virtuelle Organisation und neue Medien 2000; Josef Eul Verlag 2000

[11] Leblang, D.: The CM challenge: Configurationmanagement that works; Configuration Management, Tichy, W., F.(Hrsg.) vol.2 of Trends in Software, Verlag Wiley 1994

[12] Reinhold, M.: Rational Unified Process 2000 und V-Modell'97: Synergie oder Widerspruch, OBJEKTspektrum, 3/2000 H. 3;
URL: http://www.cocoo.de/V-Modell/V-Modell/cocoo_vmlinks.html

[13] Bruegge, B., Dutoit, A., H.: Object-Oriented Software Engineering; Prentice Hall 2000;
URL: http://vig.prenhall.com/academic/product/1,4096,0134897250,00.html

[14] Axt, H., Hertel, M., Wagner, M.: Lotus Domino & Notes. Markt & Technik Verlag München 1999

[15] Ichim I.: Schaffung einer Lösung für eine gruppenorientierte Softwareentwicklungsumgebung gemäß dem V-Modell unter Notes/ Domino 5, Dip-lomarbeit an der Fakultät Informatik der TU Dresden, 2000;

[16] Lotus Developement Corporation: Quickstart - Leitfaden, Lotus Domino Workflow, Architect und Engine; Domino Workflow Dokumentation in PDF-File; 1999