Eine Softwareproduktionsumgebung für kleine und mittelständische Unternehmen

Bericht zum 2. Meilenstein

E. Ulrich Kriegel, Dirk Kurzmann[1], Joachim Altenhein[2], Michael Löw[3], Paul Thierse[4]

email. ulrich.kriegel@isst.fhg.de

1 Fraunhofer ISST Berlin
2 System Consult GmbH
3 Tembit Software GmbH
4 AUCOTEAM GmbH

Anhang

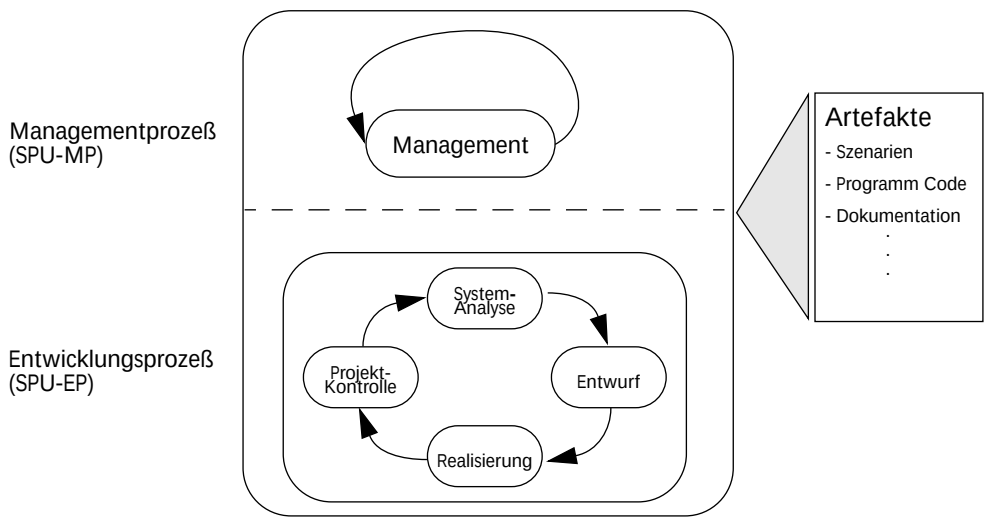
1 Einleitung

Im Rahmen des SPU-Projektes wird eine auf die Bedürfnisse und besonderen Eigenheiten kleiner und mittelständischer Unternehmen (kmU) zugeschnittene Software-Produktionsumgebung (SPU) zur Entwicklung objektorientierter Software konzipiert und prototypisch implementiert. Die Umgebung unterstützt die Entwicklung flexibler, anpassungsfähiger und langfristig weiterentwickelbarer Softwareprodukte von der Projekt-Planung bis zur Wartung des fertigen Produkts. Sie wird damit die Voraussetzung schaffen, innovative Softwareprodukte herzustellen und am Markt zu etablieren.

In der ersten Phase des Projektes wurde ein Organisations- und Vorgehensmodell zur objektorientierten Software-Entwicklung definiert[Kri+96]. Auf der Grundlage des Vorgehensmodells und der darin spezifizierten Anforderungen wurden in der darauf folgenden Projektphase konkrete Werkzeuge[1] ausgewählt, die dann zu einer prototypischen Werkzeugumgebung kombiniert wurden. Ziel dieser Publikation ist die Beschreibung der prototypischen Werkzeugumgebung.

Bild 1 zeigt das zu unterstützende Vorgehensmodell.

Bild 1
Schematische Darstellung des SPU-Prozesses



1 Bei der Auswahl von Werkzeugen wurden bevorzugt »Standard-Werkzeuge« berücksichtigt, die in vielen kmU Verwendung finden.

Es besteht aus zwei gekoppelten Teilprozessen, dem Managementprozeß und dem eigentlichen Entwicklungsprozeß. Der SPU-Managementprozeß (SPU-MP) beinhaltet alle Aktionen, die im weitesten Sinne mit der Organisation und der betriebswirtschaftlichen Überwachung des Projektes zusammenhängen. Hauptaufgabe dieses Teilprozesses ist die Planung der benötigten Ressourcen für die Durchführung des Projektes. Der SPU-Entwicklungsprozeß (SPU-EP) ist ein inkrementeller, iterativer Prozeß, der eine Folge von Prototypen liefert, an denen der Projektfortschritt gemessen werden kann. Der SPU-EP umfaßt die Phasen Analyse, Entwurf, Realisierung und Kontrolle. In beiden Teilprozessen werden entweder neue Artefakte[2] erzeugt oder bereits vorhandene verfeinert.

Für die grafische Darstellung von Sachverhalten werden, wenn möglich, Unified Method Language-konforme Diagramme [UML97] verwendet. Da die ausgewählten Werkzeuge zur Unterstützung des Entwicklungsprozesses zum Zeitpunkt der Realisierung des Prototypen nicht in einer Version vorlagen, die eine Integration über Protokolle wie Tooltalk[Too93] oder OLE[Bro94] ermöglichten, wird eine getrennte Beschreibung der Realisierung von Management- und Entwicklungsprozeß erfolgen.

Zunächst geht Kapitel 2 auf die Komponente zur Unterstützung des Prozeßmanagements ein. Hier wird beschrieben, wie durch ein Workflow- und Dokumentenverwaltungssystem die Ausführung der im Vorgehensmodell beschriebenen Aktivitäten gesteuert werden kann und die dabei entstehenden Artefakte adäquat verwaltet werden können.

Die Komponente zur Unterstützung des SPU-Managementprozesses wird in Kapitel 3 erläutert. Dazu wird zum einen die in der SPU realisierte Projektplanung allgemein dargestellt, und zum anderen werden das verwendete Projektplanungswerkzeug und die notwendigen Anpassungen an die Erfordernisse des Vorgehensmodells beschrieben.

In Kapitel 4 wird schließlich auf die im Rahmen des SPU-Entwicklungsprozesses benutzten Werkzeuge, auf die zusätzlich notwendigen Ergänzungen und auf die Arbeit mit der Entwicklungsumgebung eingegangen.

2 Als Artefakte bezeichnen wir alle Entitäten, die im Rahmen des Software-Entwicklungsprozesses erzeugt und modifiziert werden. Artefakte können z.B. Projektpläne, Dokumente, Entwürfe, Implementationsdateien oder Prototypen sein.

2 Basistechnologien: Workflow- und Dokumenten- und Versions/ Konfigurationsmanagement

Die SPU-Workflow-Komponente wurde mit dem Groupware Produkt Lotus Notes 4.5 entwickelt. Lotus Notes benutzt zur strukturierten Verwaltung von Artefakten spezielle Datenbanken. Das grundlegende Element einer solchen Datenbank ist ein individuelles Dokument, dessen Struktur durch ein sogenanntes »Formular« definiert wird. Formulare können Felder mit unterschiedlichen Inhalten enthalten, z.B. Freitext, OLE-Elemente, Dateien als Attachments oder als »Buttons« bezeichnete Schaltflächen, die bei Aktivierung definierte Aktionen initiieren. Dokumente können direkt, über Buttons oder über »Agenten« genannte Hintergrundprogramme verändert werden. Für die Darstellung der Inhalte von Datenbanken können spezifische Ansichten definiert werden.

Ein wesentliches Kennzeichen von Notes ist die Möglichkeit zur Replikation, also dem Abgleich der Dokumentänderungen zwischen auf den verschiedenen Systemen verteilten Kopien einer Datenbank. Dabei vollziehen entfernte Systeme die lokal durchgeführten Änderungen und umgekehrt. Durch diesen Replikationsmechanismus können Benutzer an verschiedenen Orten unabhängig voneinander arbeiten und trotzdem leicht auf die Arbeitsergebnisse der anderen Benutzer zugreifen. Neben dem Dokumentenaustausch werden auch E-Mail, News und Gruppenkalender zur Unterstützung der Kommunikation und Koordination zwischen den Benutzern angeboten.

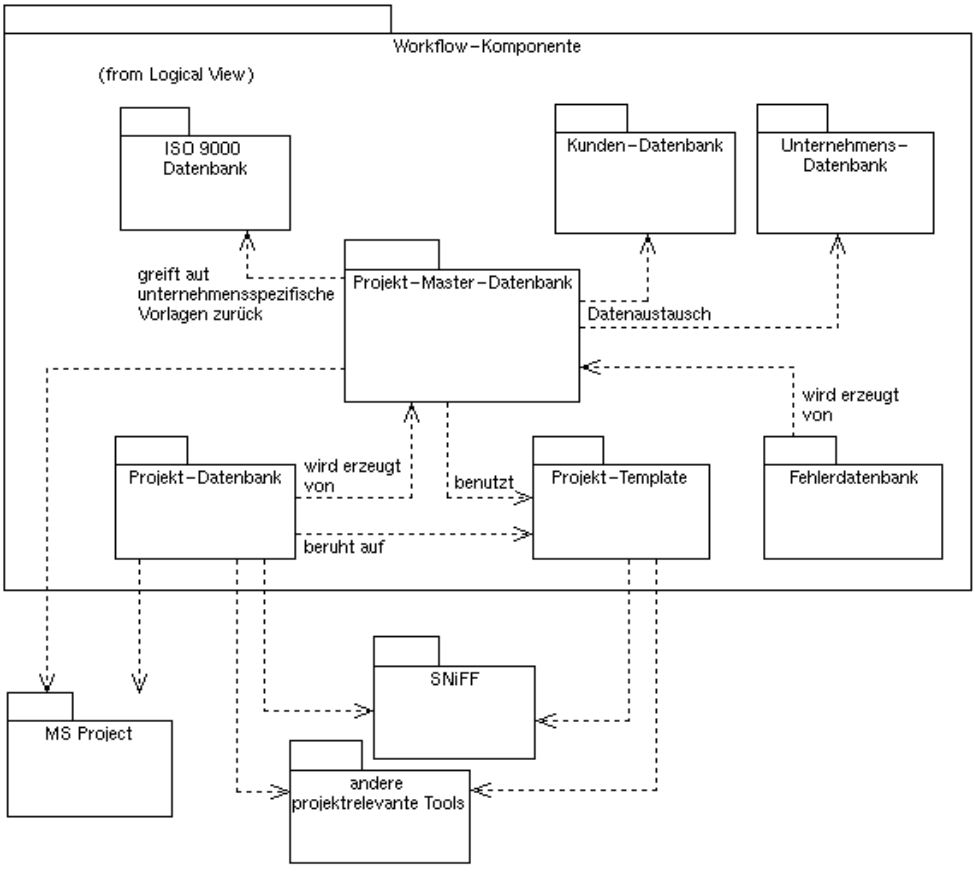
Für anwendungsspezifische Anpassungen steht eine integrierte Entwicklungsumgebung zur Verfügung. Während einfache Operationen per Mausklick in Form von sogenannten »Smart Actions« konfiguriert werden können, bietet sich für umfangreiche Entwicklungsaufgaben die Basic-ähnliche Programmiersprache Lotus-Script an. Letztere wurde für die Realisierung der SPU-Workflow-Komponente intensiv genutzt, um die im ersten Meilenstein aufgestellten Anforderungen hinsichtlich Triggermechanismen und flexibler Vorgangssteuerung zu erfüllen.

2.1 Architektur der Workflow-Komponente

Zentrale Elemente der Architektur sind die Projekt-Master-Datenbank, über die sämtliche Projekte verwaltet werden, und die spezifischen Projekt-Datenbanken, in denen alle Aufgaben und Ergebnisse eines Projektes enthalten sind. Zur Erstellung einer neuen Projekt-Datenbank wird auf eine bestehende Projektschablone zurückgegriffen. Die Projekt-Master-Datenbank bedient sich verschiedener Hilfsdatenbanken wie der Kunden-Datenbank zur Verwaltung von Kunden-

daten, der Unternehmens-Datenbank zur Verwaltung von Mitarbeiterprofilen und einer (optionalen) ISO 9000-Datenbank zur Unterstützung des Qualitätsmanagements. Die in der Entwicklungsphase eines Projektes aufgetretenen Fehler können in einer Fehlerdatenbank festgehalten werden. Es bestehen direkte Schnittstellen zwischen der Projekt-Master-Datenbank und den Projekt-Datenbanken einerseits und dem Werkzeug zur Unterstützung des Management-Prozesses (siehe Kapitel 3) andererseits, über die ein Austausch von Projektplanungsdaten erfolgen kann. Das in Bild 2 gezeigte Klassendiagramm gibt einen Überblick über die Architektur der Workflow-Komponente. Im folgenden wird die Benutzerschnittstelle dieser Komponente beschrieben, für die Darstellung ihrer Struktur sei auf den Anhang A verwiesen.

Bild 2
Architektur der Workflow-Komponente



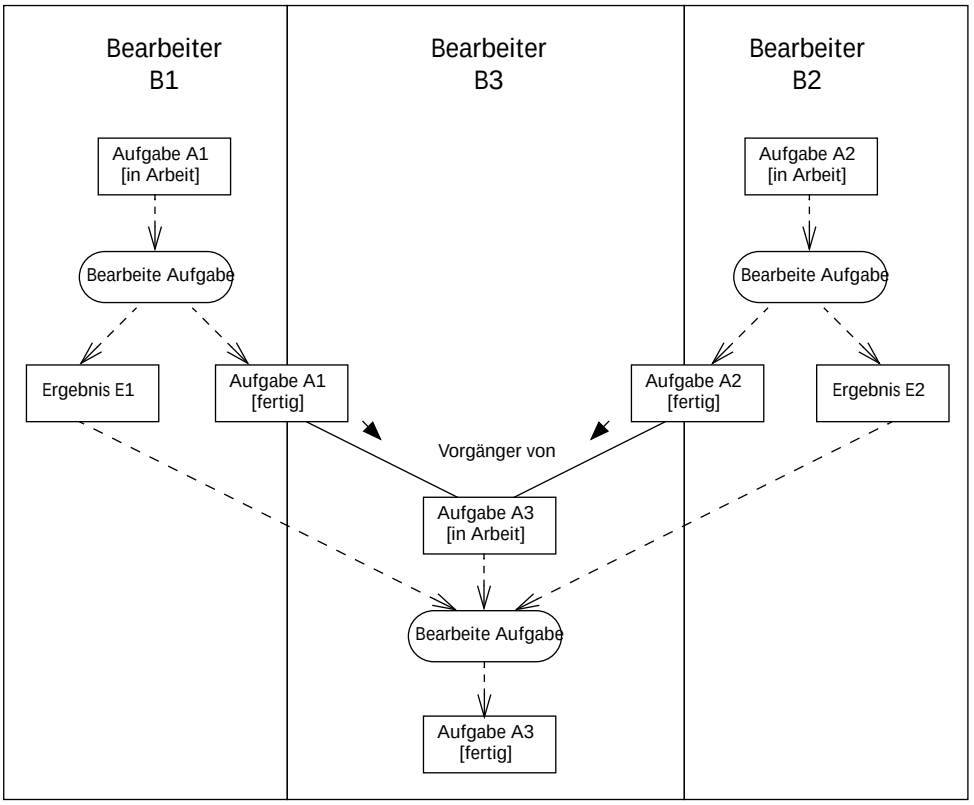
2.2 Workflow

Im wesentlichen lassen sich in der SPU zwei Workflows identifizieren: der Aufgaben-Workflow und der Ergebnis-Workflow.

2.2.1 Aufgaben-Workflow

Das activity-Diagramm (Bild 3) für den Aufgaben-Workflow zeigt die Zuordnung der Aufgaben zu den Bearbeitern. Die Aufgaben werden mittels Agenten in der Projekt-Datenbank verschickt. Sobald ein Bearbeiter seine Aufgabe als erledigt markiert hat, wird geprüft, ob die Nachfolge-Aufgabe bearbeitet werden kann. Ist dies der Fall, wird der Status der neuen Aufgabe auf »in Arbeit« gesetzt und per Mail an den Bearbeiter geschickt. Weiterhin werden alle Output-Ergebnisse der Vorgängeraufgaben als Input-Ergebnisse der neuen Aufgabe zugewiesen.

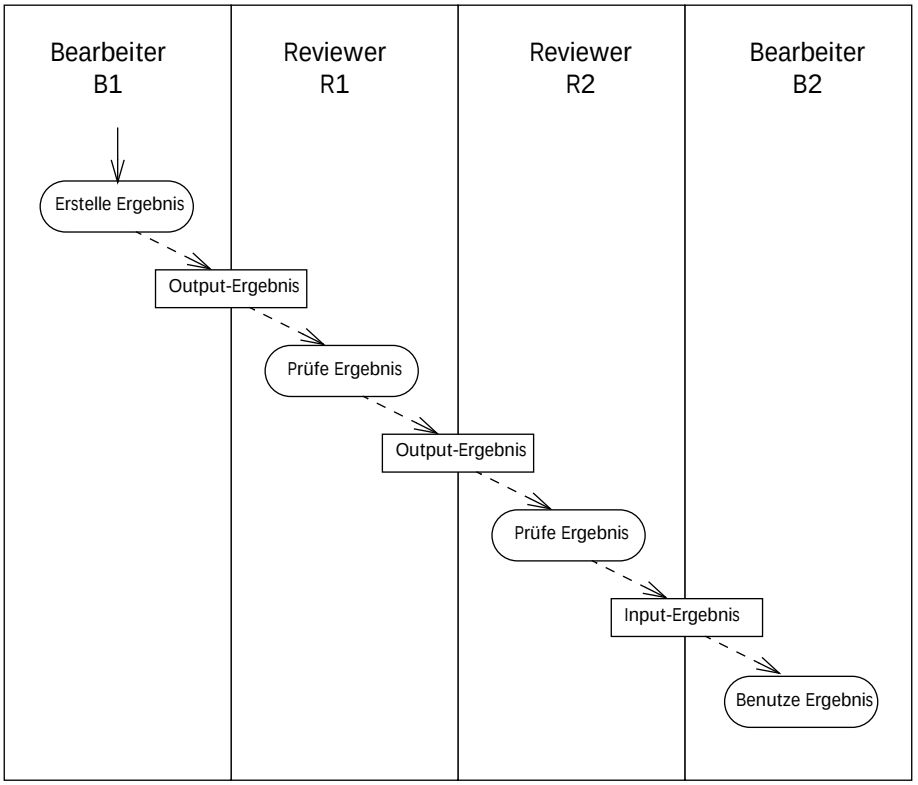
Bild 3
Aufgaben-Workflow



2.2.2 Ergebnis-Workflow

Der Ergebnis-Workflow bezieht sich im wesentlichen auf die Reviews. Der Bearbeiter eines Ergebnisses bestimmt die Reviewer und startet das Review. Bild 4 zeigt die Aktivitäten bei einem seriellen Review. Es besteht auch die Möglichkeit von parallelen Reviews. Die Ergebnisse werden nacheinander an die Reviewer verschickt.

Bild 4
Activity-Diagramm
Ergebnis-Workflow



2.3 Beschreibung der einzelnen Komponenten

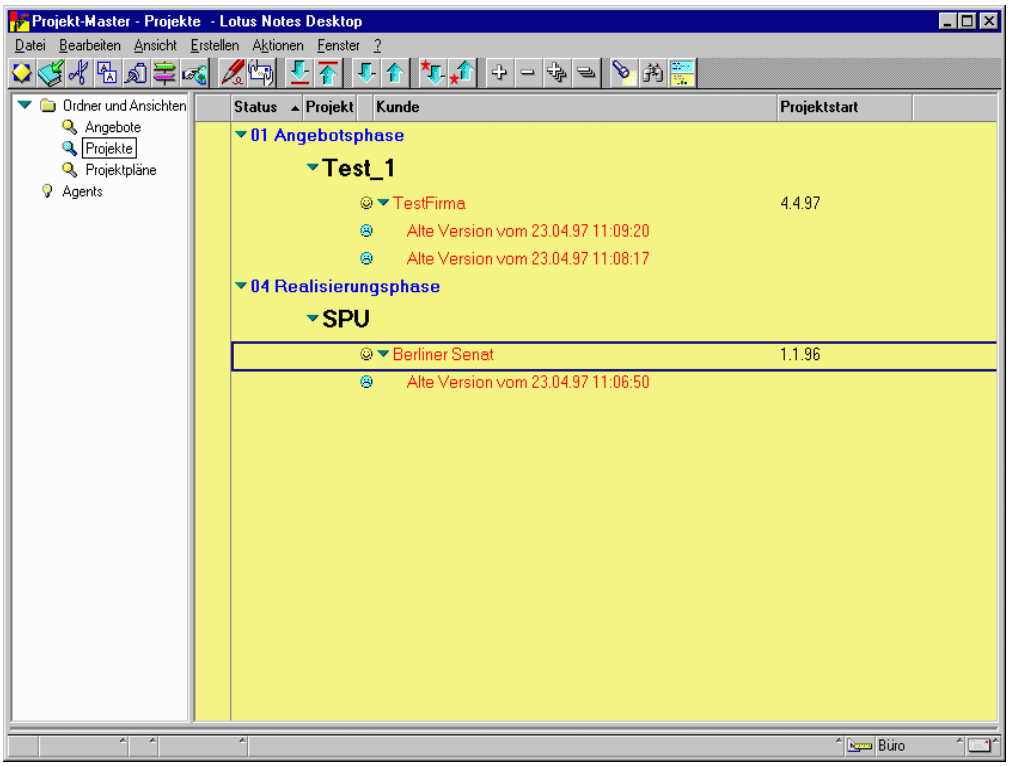
Im folgenden sollen die einzelnen Komponenten hinsichtlich ihres Aufbaus, der in ihnen abgelegten Attributstrukturen und der implementierten Ansichten, Navigatoren und Aktionen näher erläutert werden.

2.3.1 Projekt-Master-Datenbank

Die Projekt-Master-Datenbank dient zur Verwaltung aller Projekte. Für jedes Projekt wird ein Dokument mit den wichtigsten Daten angelegt. Aus dem Projektdokument können die eigentliche Projekt-Datenbank und eine Fehlerdatenbank erstellt werden. Zusätzlich werden in der Projekt-Master-Datenbank der Projektplan und das Angebot verwaltet. Die Vorlagen werden aus der ISO 9000-Datenbank (siehe Kapitel 2.3.7) geholt, womit gewährleistet ist, daß immer die aktuellen Vorlagen benutzt werden.

Der nachfolgende Bildschirmausschnitt zeigt die Projektansicht.

Bild 5
Projekt-Master

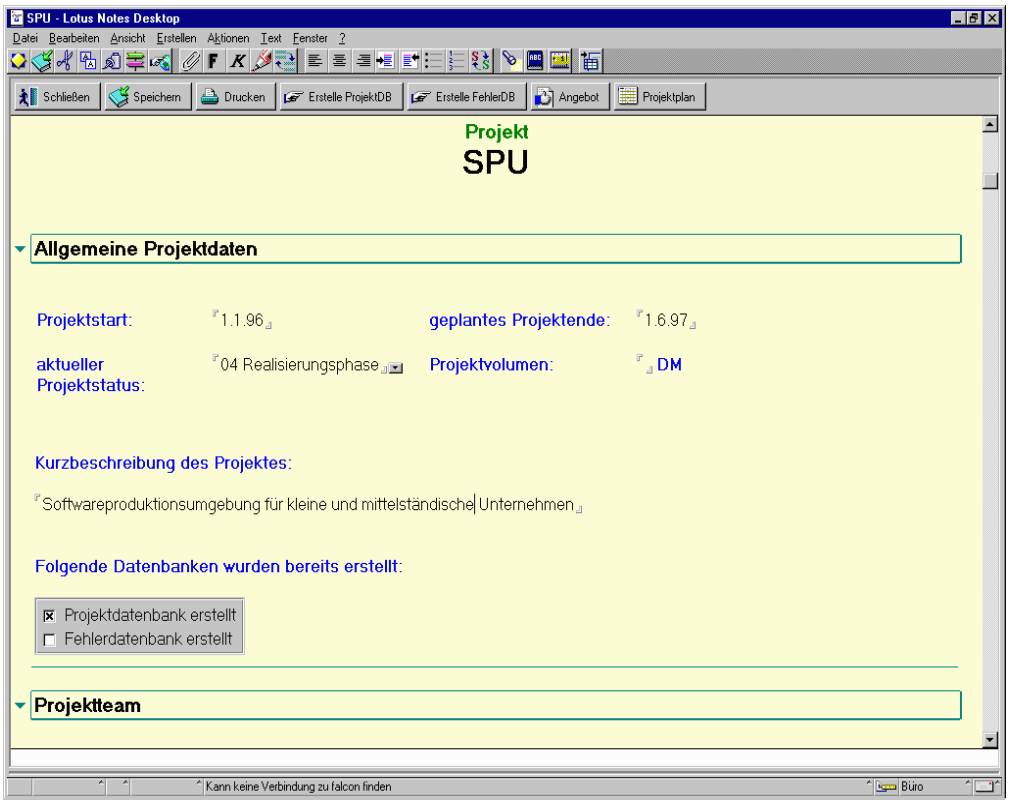


Aus der Projekt-Master-Datenbank heraus können über verschiedene Masken Angebote gepflegt, Projekte angelegt und die dazugehörigen Projektpläne verwaltet werden.

Projekt

Der nachfolgende Bildschirmausschnitt zeigt die Eingabemaske »Projekt«.

Bild 6
Projekt-Maske



Über Schaltflächen und das Aktionsmenü kann direkt aus der Projekt-Maske heraus u.a.

- eine Projektgruppe im Namens- und Adreßbuch erstellt und mittels der Ansicht »Mitarbeiter« aus der Unternehmensdatenbank angezeigt werden,
- ein neues Kundenprofil in der Datenbank »Kunden« erstellt und durch eine entsprechende Ansicht aus der Kundendatenbank dargestellt werden,
- die Projekt-Datenbank mit Hilfe der Projekt-Datenbankschablone, des Projektplanes und Formularen aus der ISO 9000-Datenbank erzeugt werden,
- die Fehlerdatenbank unter Rückgriff auf eine entsprechende Fehlerdatenbankschablone generiert werden,
- der Projektplan angezeigt werden,
- zu einem eventuell zugrundeliegenden Angebot verzweigt werden.

Die konfigurierte Projekt-Ansicht zeigt alle Projektdokumente nach Projektstatus sortiert. Weiterhin werden auch alle Versionen eines Projektdokumentes angezeigt.

Alle Projektpläne können mittels eines speziellen Navigators nach Projekten geordnet dargestellt werden.

Durch die Aktivierung des Buttons »Projektplan« wird das Projektplanungswerkzeug gestartet und der aktuelle Projektplan angezeigt. Über das Aktionsmenü kann nach Änderungen die neue Version des Plans gespeichert werden. Des weiteren stehen Aktionen zur Sperrung und Entsperrung des Plans und zur Durchführung eines Reviewzyklus wie Initialisierung, Start, Ende und Löschung zur Verfügung.

Reviewoptionen

Die Parametrisierung eines Reviewzyklus erfolgt anhand einer speziellen Review-Optionsmaske, in der folgende Felder festgelegt werden:

- Review Typ: seriell »nur Ausgangsdokument«, seriell »mit allen Versionen«, parallel »mit Dokumentensperre«, parallel »ohne Dokumentsperre«,
- Zur Verfügung stehende Zeit: Kein Zeitlimit, Senden zum nächsten Reviewer nach ..., Schicke Erinnerung nach ... Tagen,
- Benachrichtigung: Benachrichtigung nach jedem Review, Benachrichtigung nach dem letzten Review,
- Einstellungen speichern.

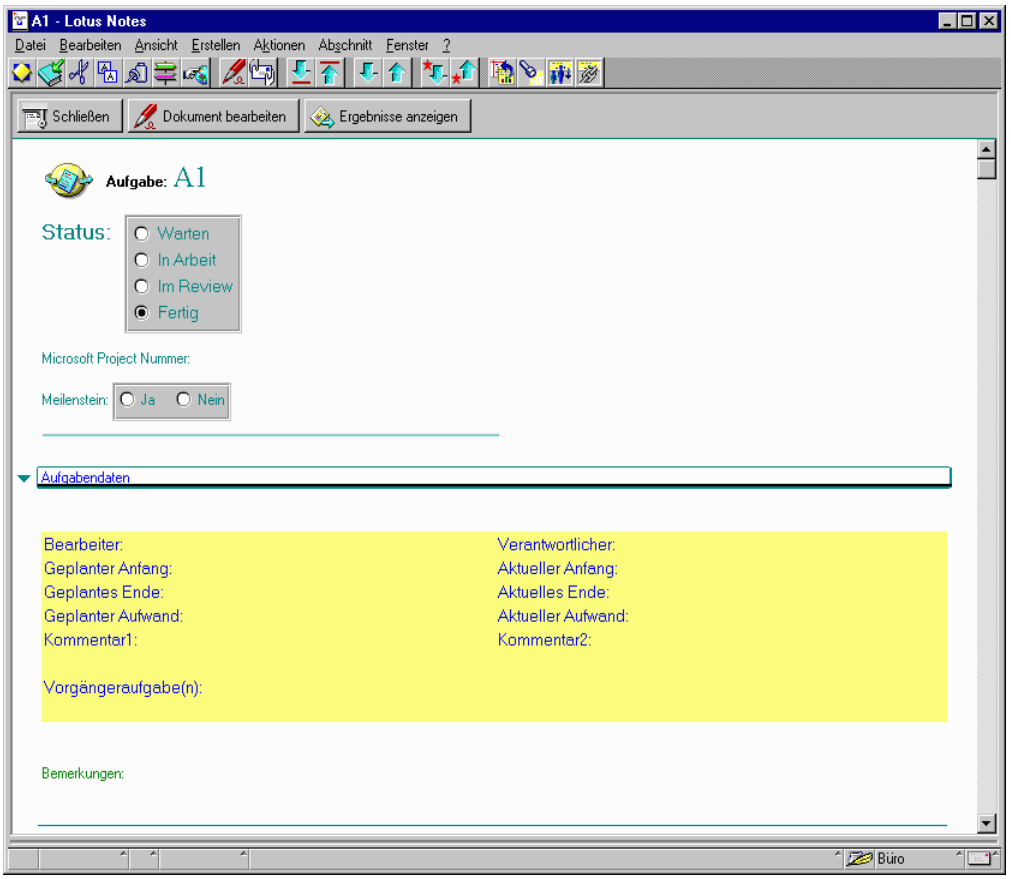
Ein spezieller Button »Angebot« startet ein Textverarbeitungswerkzeug, im speziellen Falle »MS Word«, zur Anzeige des aktuellen Angebots. Auch für Angebote stehen die schon für die Projektpläne erläuterten Aktionen zur Dokumentensperrung und Reviewdurchführung zur Verfügung. Ein Navigator listet alle Angebote nach Projekten sortiert auf.

2.3.2 Projekt-Datenbankschablone

Diese Datenbankschablone enthält alle Masken, Ansichten und Agenten, die für die Projekt-Datenbank benötigt werden. Aus dem Projektdokument der Projekt-Master-Datenbank wird die Projekt-Datenbank mittels dieser Schablone erstellt. Dabei werden die Struktur und die Daten aus dem Projektplan mittels einer Dateischnittstelle übernommen, d.h. die Datenbank wird automatisch mit Aufgaben und Ergebnisvorlagen (aus der ISO 9000-Datenbank) gefüllt.

Der nachfolgede Bildschirmausschnitt zeigt die Eingabemaske »Aufgabe«.

Bild 7
Maske „Aufgabe“



In der Datenbank sind Masken für Aufgaben und Ergebnisse vorgesehen.

Zu einer Aufgabe können alle Ergebnisse, die dieser Aufgabe zugeordnet sind, angezeigt werden.

Mit Hilfe des Buttons »Ergebnisdokument einfügen« kann eine Datei in das Ergebnisdokument eingebettet werden, die mit einer anderen Anwendung erstellt wurde. Darüber hinaus werden Funktionen zur Transformation eines Input-Dokumentes in ein Output-Dokument, zur Dokumentsperrung und zur Durchführung von Reviews (s.o.) angeboten.

Ansichten

Zur übersichtlichen Darstellung der unterschiedlichen, in der Projekt-Datenbank hinterlegten Informationen können verschiedene Ansichten gewählt werden:

– Aufgaben & Ergebnisse: Diese Ansicht zeigt alle Aufgaben mit Ergebnissen nach Aufgaben sortiert.
– Aufgaben: Die Ansichten »nach Bearbeiter«, »nach Status« und »meine Aufgaben« bieten weitere Möglichkeiten der Informationsdarstellung.

- Ergebnisse: Für die Ergebnisdarstellung gibt es drei weitere Ansichten: »nach Bearbeiter«, »nach Status« und »meine Ergebnisse«.
- Kalender: In dieser Ansicht werden alle Anfangs- und Endtermine angezeigt.
- Papierkorb: Hier werden alle gelöschten Dokumente angezeigt.

Agenten

Zur effektiven Arbeit mit der Projektdatenbank und zur Realisierung des Workflows wurden die beiden folgenden Agenten implementiert:

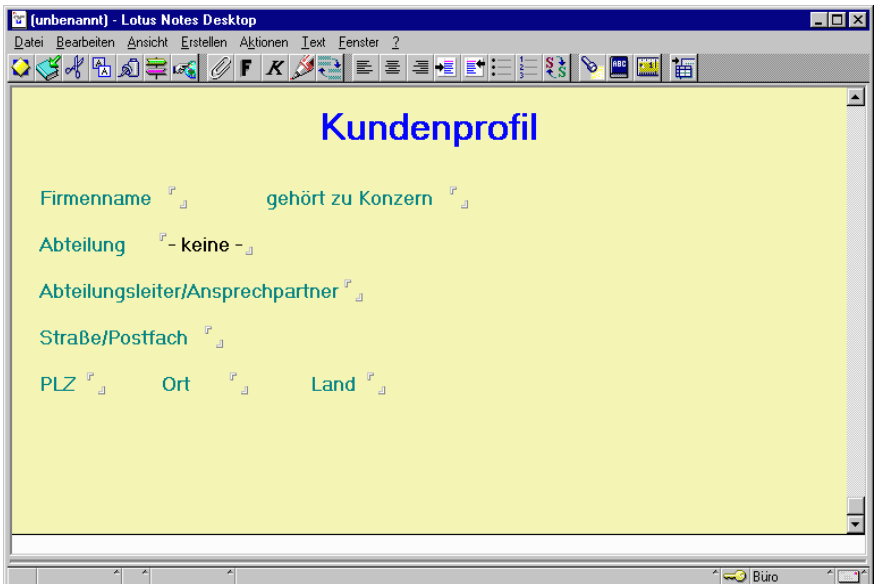
- »Suche fällige Reviews«: Durchsucht die Datenbank nach Ergebnissen, bei denen das Review fällig ist, d.h. es werden Erinnerungs- bzw. Zeitüberschreitungsmails an die Reviewer gesendet.
- »Überprüfe Aufgabenstatus«: Überprüft, ob alle Vorgänger einer Aufgabe den Status fertig haben. Ist dies der Fall, wird der Status auf »in Arbeit« gesetzt, die Output-Ergebnisdokumente werden der neuen Aufgabe als Input-Ergebnisdokumente zugewiesen und der Bearbeiter wird per Mail benachrichtigt.

2.3.3 Kundendatenbank

In der Kundendatenbank werden alle Kundendaten verwaltet. Diese Datenbank kann auch dazu verwendet werden, die gesamte nicht-projektbezogene Korrespondenz zu verwalten, Telefonnotizen festzuhalten usw. Momentan dienen die Ansichten dieser Datenbank zur Auswahl in der Projekt-Master-Datenbank.

Das Bild 8 zeigt die Eingabemaske »Kundenprofil«.

Bild 8
Kundenprofil



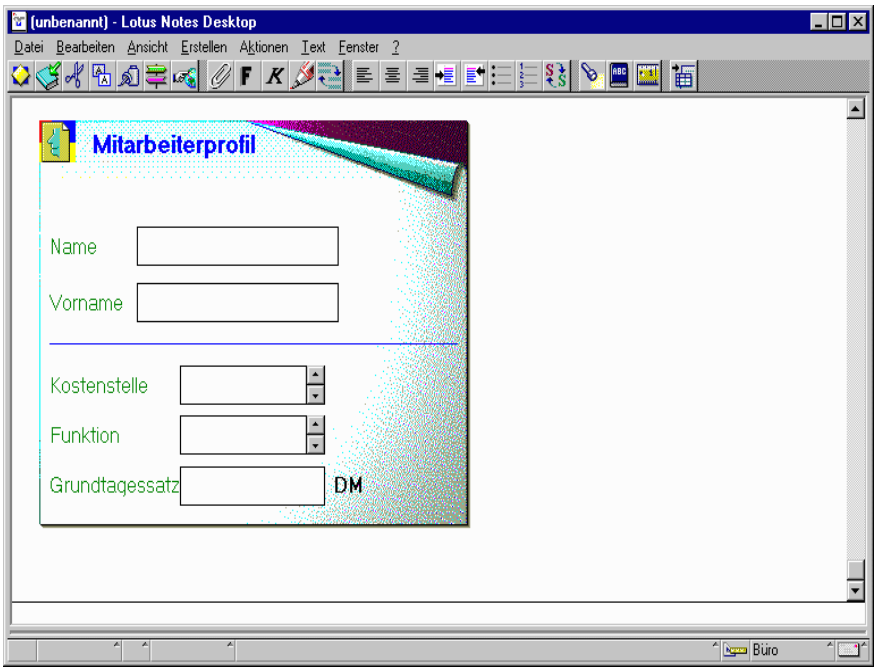
Alle Kunden können in einer Ansicht alphabetisch aufgelistet werden. Des weiteren ist in dieser Datenbank die Kundenansicht definiert, die aus der Projekt-Master-Datenbank heraus angesteuert werden kann.

2.3.4 Unternehmensdatenbank

Die Unternehmensdatenbank enthält momentan nur die Mitarbeiterprofile. Sie kann aber um verschiedene administrative Komponenten erweitert werden, z.B. Urlaubslisten, Reisekosten, Korrespondenz.

Der nachfolgende Bildschirmausschnitt zeigt die Eingabemaske »Mitarbeiterprofil«.

Bild 9
Eingabemaske »Mitarbeiterprofil«



Eine Mitarbeiter-Ansicht listet alle Mitarbeiter alphabetisch auf.

2.3.5 Fehlerdatenbank

Die Schablone für Fehlerdatenbanken dient zur Erstellung einer projektspezifischen Fehlerdatenbank. Diese kann aus dem Projektdokument der Projekt-Master-Datenbank erstellt werden. In dieser Datenbank werden alle Fehler, die bei Tests bzw. während der Entwicklung gefunden werden, registriert und dem jeweiligen Entwickler zugeordnet, der den Fehler behebt.

Der nachfolgende Bildschirmausschnitt zeigt die Eingabemaske »Fehlerbeschreibung«.

Bild 10
Eingabemaske »Fehlerbeschreibung«

Die Datenbank enthält Masken für Fehlerbeschreibungen und Module.

Über die Schaltfläche »Bearbeiter übergeben« kann der Bearbeiter des Fehlers festgelegt werden.

Zu einem Modul soll zunächst nur das Feld »Modulbezeichnung« festgehalten werden.

Es werden zwei Ansichten angeboten:

Übersicht: Es kann nach »Bearbeiter«, nach »Priorität«, nach »Schweregrad«, nach »Status« und nach »offenen Fehlern« sortiert werden.

Optionen: Unter diesem Ansichtspunkt wird eine Liste der Module aufgeführt.

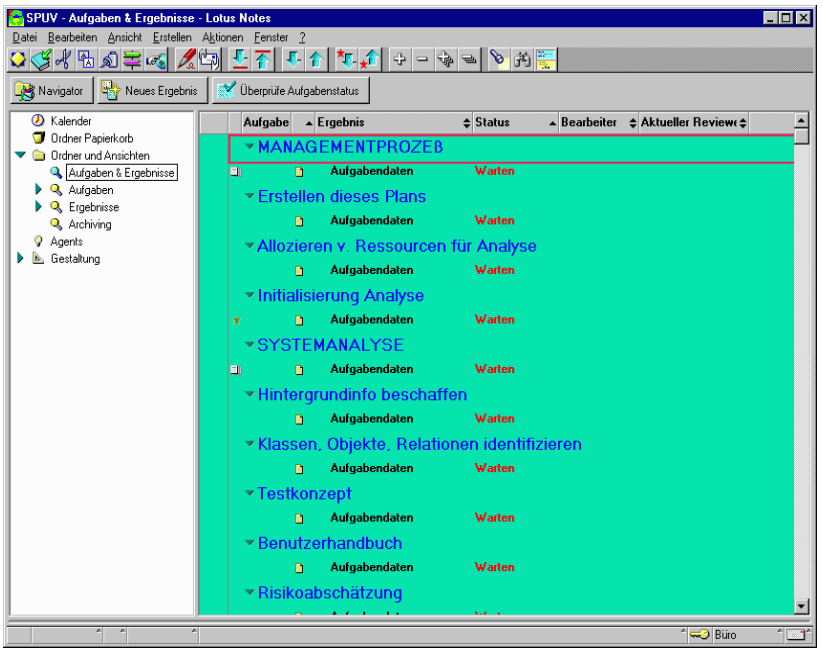
2.3.6 Projektspezifische Datenbank

Die projektspezifische Datenbank gibt einen Überblick über den Ablauf des Projektes. Sie wird aus dem Projektdokument der Projekt-Master-Datenbank

erstellt. Die Aufgaben- und Ergebnisstruktur erhält sie aus dem Projektplan, der sich ebenfalls in der Projekt-Master-Datenbank befindet.

Der nachfolgende Bildschirmausschnitt zeigt die Ansicht »Aufgaben & Ergebnisse«.

Bild 11
Ansicht »Aufgaben & Ergebnisse«



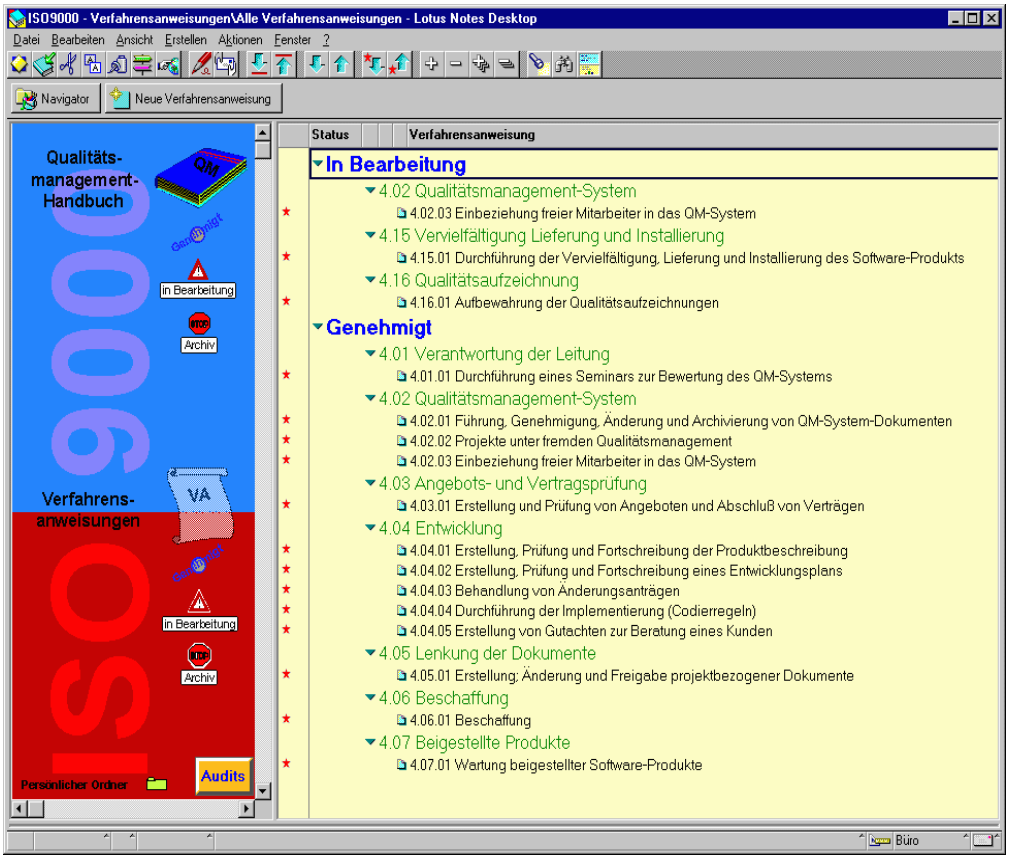
Die vorhandenen Masken, Ansichten und Agenten werden bei der Erzeugung der Projekt-Datenbank aus der Projekt-Datenbankschablone (siehe Abschnitt 2.3.2) übernommen.

2.3.7 ISO 9000 Datenbank

Die ISO 9000-Datenbank ist eine eigenständige Komponente, d.h. sie kann auch abgekoppelt von den SPU-Workflows benutzt werden. Diese Datenbank hat die Aufgabe, das Qualitätsmanagement-Handbuch, die Verfahrensanweisungen mit den darin enthaltenen Formularen, wie Angebotsvorlage, Projektplanschablone usw., zentral zu verwalten und jedem Mitarbeiter zur Verfügung zu stellen. Weiterhin ist das Genehmigungsverfahren der Dokumente als Workflow in dieser Datenbank realisiert. Diese Datenbank und das Genehmigungsverfahren entsprechen den ISO 9001 Richtlinien und wurden von der DQS Deutsche Gesellschaft zur Zertifizierung von Qualittsmanagementsystemen mbH zertifiziert.

Der nachfolgende Bildschirmausschnitt zeigt die Ansicht aller Verfahrensanweisungen.

Bild 12
Ansicht »Verfahrensanweisungen«



Die ISO 9000-Datenbank enthält verschiedene Masken für QM-Elemente, Verfahrensanweisungen, Abweichungsmeldungen und Korrekturmaßnahmen und für den Auditplan und den Auditbericht. Zu jeder Maske werden im folgenden die definierenden Attribute, eventuell vorhandene Schaltflächen und implementierte Aktionen aufgelistet.

Ansichten

Es werden verschiedene Ansichten für die QM-Elemente und die Verfahrensanweisungen angeboten:

- QM-Elemente: Es gibt vier verschiedene Ansichten der QM-Elemente. Die Ansicht »Alle QM-Elemente« zeigt alle Dokumente nach Status (genehmigt, in Bearbeitung, archiviert) sortiert. Die anderen Ansichten sind nach Status getrennt und nach Numerierung sortiert.
- Verfahrensanweisungen: Es gibt vier verschiedene Ansichten der Verfahrensanweisungen. Die Ansicht »Alle Verfahrensanweisungen« zeigt alle Doku-

mente nach Status (genehmigt, in Bearbeitung, archiviert) sortiert. Die anderen Ansichten sind nach Status getrennt und nach Numerierung sortiert.

Agent »Drucken des QM-Handbuches«

Dieser Agent ermittelt alle genehmigten Elemente der ISO 9000-Datenbank, erstellt ein aktuelles Deckblatt und druckt das QM-Handbuch.

3 Werkzeugunterstützung für den SPU-Managementprozeß

3.1 Einführung

Die Qualität der Projektplanung ist entscheidend für den technischen und wirtschaftlichen Erfolg eines jeden Projektes. Fehler in diesem Bereich können zu erheblichen finanziellen Schäden für das ausführende Unternehmen führen. Aus diesem Grund erfolgt eine besondere Konzentration auf die Projektplanung.

Die Projektplanung, wie sie im SPU-Prototypen abgebildet wird, erfolgt in mehreren Phasen:

- Festlegen des Workflows auf der Basis des im Meilensteinpapier [Kri+96] beschriebenen Workflows,
- Planen der Aufwände und Termine für die einzelnen Phasen des Projektes,
- Planen der Ressourcen,
- Projektverfolgung.

Das Arbeitsdokument der Projektplanung ist der Projektplan. Er enthält Aufgaben, Termine, Abhängigkeiten zwischen Aufgaben, Verantwortliche sowie zusätzliche Informationen für den Workflow. Er wird aus Gründen, die weiter unten beschrieben sind, in MS Project erstellt und bearbeitet. Er ist ein »lebendes« Dokument, wird also immer an die aktuelle Situation angepaßt. So ist der Projektleiter jederzeit über den Projektstand informiert und kann bei Problemen frühzeitig Maßnahmen einleiten.

3.1.1 Festlegen des Workflows

Die erste Phase der Projektplanung ist die Festlegung des Workflows. Der SPU-Workflow ist im SPU-Prototyp in ein MS Project-Projekt übertragen worden. Dieses Muster wird für jedes neue Projekt kopiert und anschließend vom Projektleiter angepaßt. Er kann dabei sowohl Aufgaben hinzufügen oder löschen, als auch die für den Workflow relevanten Informationen, z.B. Vorgänger/Nachfolgerbeziehungen, modifizieren.

3.1.2 Planen der Aufwände und Termine

Nachdem der Workflow den Bedürfnissen des speziellen Projektes angepaßt worden ist, wird die Terminplanung durchgeführt. Dazu macht der Projektleiter aufgrund seiner Erfahrung Abschätzungen und trägt diese in den Projektplan

ein. Der SPU-Musterprojektplan hilft dabei dem Projektleiter, indem er typische Werte vorgibt.

Neben den Aufwänden gehen auch externe Vorgaben in den Plan ein. Dies sind insbesondere Zusagen über Lieferungen, die dem Kunden gemacht wurden. Diese können als Meilensteine in dem Projektplan aufgenommen werden.

3.1.3 Planen der Ressourcen

Natürlich lassen sich die Termine nicht unabhängig von den Ressourcen planen. Unter Ressourcen werden im Rahmen des SPU-Prototypen Mitarbeiter verstanden. Die Verwaltung von Sachmitteln, wie Rechner, Software-Lizenzen oder Räumlichkeiten, wird in dieser Ausbaustufe des Protoypen nicht behandelt.

Im SPU-Musterprojektplan werden dem Projektleiter die einzusetzenden Mitarbeiter anhand von Rollen vorgeschlagen. Hat der Projektleiter einen Mitarbeiter für eine Rolle ausgesucht, kann er über eine spezielle Funktion des SPU-Prototypen diesen Mitarbeiter allen Aufgaben zuordnen, die von dieser Rolle zu bearbeiten sind.

Im SPU-Musterprojektplan werden folgende Rollen benutzt:

Ana	Analytiker
Arch	Architekt
Dom	Domain Experte
Ew	Entwickler & Programmierer
Int	Integrator
PM	Projekt Manager
Mgt	Management
Tst	Tester

Die Verfügbarkeit der Mitarbeiter beeinflußt in erheblichem Maße die Termine. Der Projektleiter ordnet die Mitarbeiter den Aufgaben zu. Damit ist der erste Teil der Projektplanung abgeschlossen. Der SPU-Workflow kann jetzt gestartet werden.

3.1.4 Projektverfolgung

Nach Start des Workflows wird der Projektplan ständig aktuell gehalten. Die Fertigstellung von Aufgaben wird im Projektplan vermerkt und die Termine werden

neu berechnet. Alle Änderungen in den Aufwandschätzungen, bei der Verfügbarkeit von Mitarbeitern sowie bei den externen Terminen werden vom Projektleiter im Projektplan eingetragen.

Er wird dabei von der Workflow-Komponente des SPU-Prototypen unterstützt. Die Fertigstellung von Aufgaben wird ihm durch E-Mail mitgeteilt. Mitarbeiter, die nachfolgende Aufgaben zu bearbeiten haben, werden ebenfalls über E-Mail automatisch informiert. Letzteres erfolgt sofort bei Beendigung der Vorgängeraufgabe. Der Workflow blockiert also nicht, bis der Projektleiter die E-Mail liest.

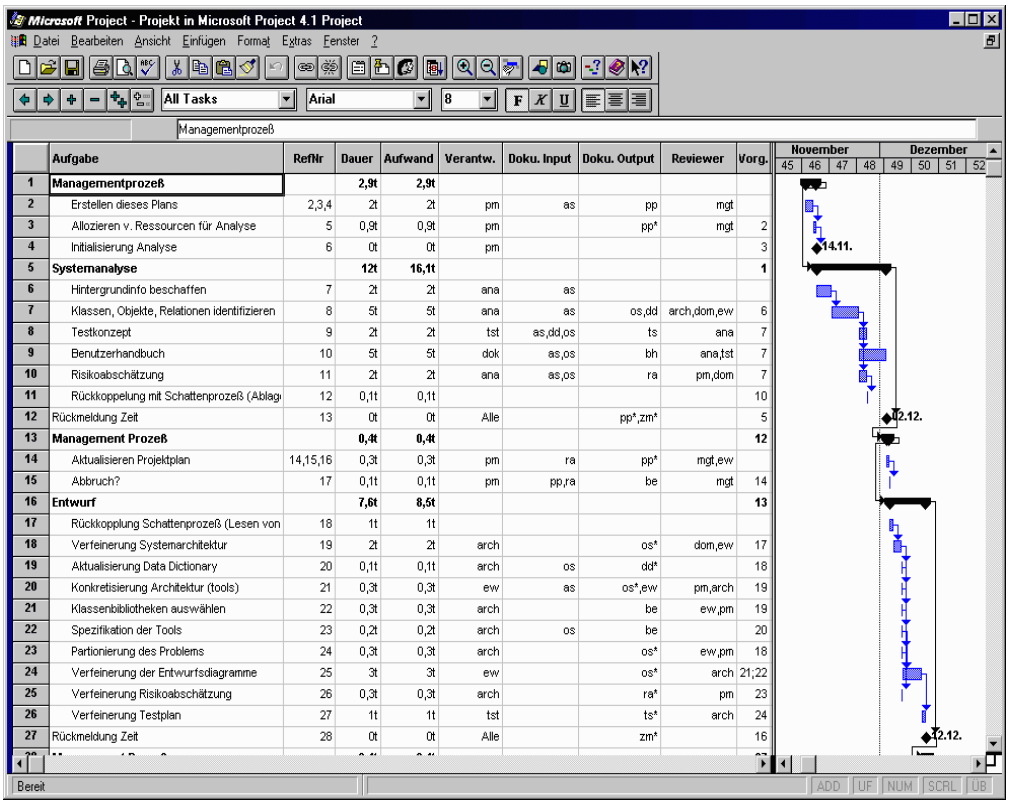
3.2 Technische Realisierung

3.2.1 Das Projektplanungswerkzeug

Als Projektplanungswerkzeug wurde MS Project ausgewählt. Folgende Eigenschaften machten es zum Produkt unserer Wahl:

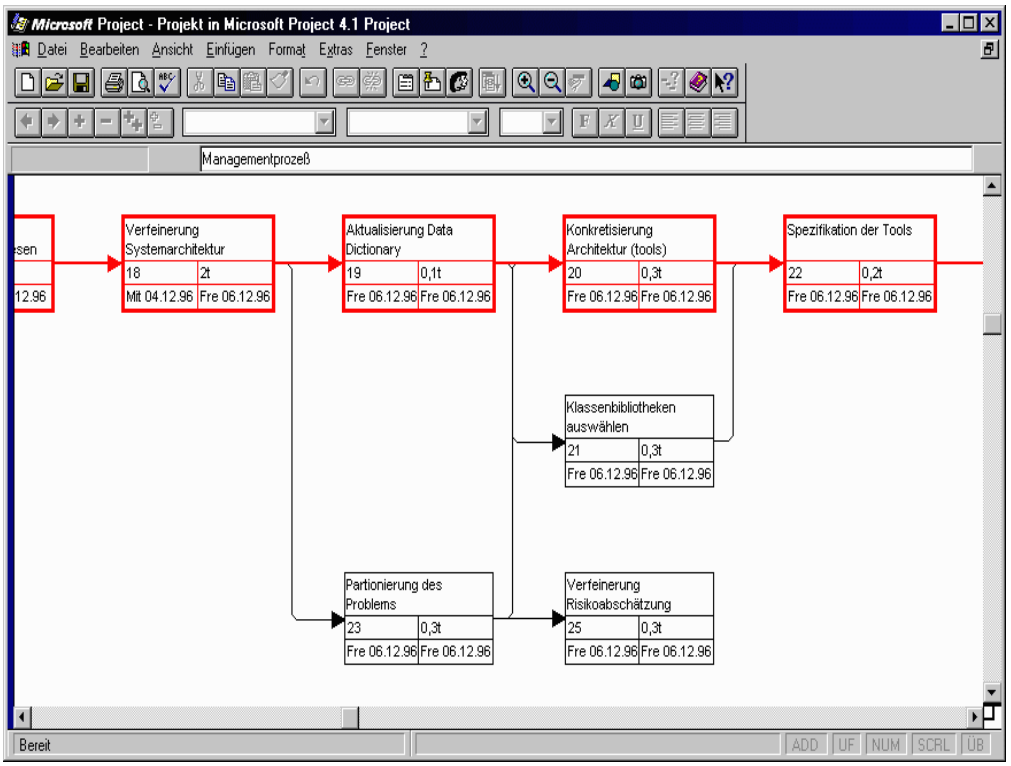
- Vielfältige Möglichkeit der graphischen und tabellarischen Darstellung und Strukturierung, z.B. Gantt-Diagramme (Bild 13) oder Netzplan-Diagramme (Bild 14).

Bild 13
Gantt-Diagramm



	Aufgabe	RefNr	Dauer	Aufwand	Verantw.	Doku. Input	Doku. Output	Reviewer	Vorg.
1	**Managementprozeß**		**2,9t**	**2,9t**					
2	Erstellen dieses Plans	2,3,4	2t	2t	pm	as	pp	mgt	
3	Allozieren v. Ressourcen für Analyse	5	0,9t	0,9t	pm		pp*	mgt	2
4	Initialisierung Analyse	6	0t	0t	pm				3
5	**Systemanalyse**		**12t**	**16,1t**					**1**
6	Hintergrundinfo beschaffen	7	2t	2t	ana	as			
7	Klassen, Objekte, Relationen identifizieren	8	5t	5t	ana	as	os,dd	arch,dom,ew	6
8	Testkonzept	9	2t	2t	tst	as,dd,os	ts	ana	7
9	Benutzerhandbuch	10	5t	5t	dok	as,os	bh	ana,tst	7
10	Risikoabschätzung	11	2t	2t	ana	as,os	ra	pm,dom	7
11	Rückkoppelung mit Schattenprozeß (Ablag	12	0,1t	0,1t					10
12	Rückmeldung Zeit	13	0t	0t	Alle		pp*,zm*		5
13	**Management Prozeß**		**0,4t**	**0,4t**					**12**
14	Aktualisieren Projektplan	14,15,16	0,3t	0,3t	pm	ra	pp*	mgt,ew	
15	Abbruch?	17	0,1t	0,1t	pm	pp,ra	be	mgt	14
16	**Entwurf**		**7,6t**	**8,5t**					**13**
17	Rückkopplung Schattenprozeß (Lesen von	18	1t	1t					
18	Verfeinerung Systemarchitektur	19	2t	2t	arch		os*	dom,ew	17
19	Aktualisierung Data Dictionary	20	0,1t	0,1t	arch	os	dd*		18
20	Konkretisierung Architektur (tools)	21	0,3t	0,3t	ew	as	os*,ew	pm,arch	19
21	Klassenbibliotheken auswählen	22	0,3t	0,3t	arch		be	ew,pm	19
22	Spezifikation der Tools	23	0,2t	0,2t	arch	os	be		20
23	Partionierung des Problems	24	0,3t	0,3t	arch		os*	ew,pm	18
24	Verfeinerung der Entwurfsdiagramme	25	3t	3t	ew		os*	arch	21;22
25	Verfeinerung Risikoabschätzung	26	0,3t	0,3t	arch		ra*	pm	23
26	Verfeinerung Testplan	27	1t	1t	tst		ts*	arch	24
27	Rückmeldung Zeit	28	0t	0t	Alle		zm*		16

Bild 14
Netzplan-Diagramm



- Ist- und Sollpläne,
- Erweiterbarkeit durch benutzerdefinierte Attribute für Aufgaben,
- Ressourcenverwaltung,
- MS Project ist OLE-fähig,
- Macro-Programmierung in Visual Basic,
- Erweiterungen können in der Projektdatei gespeichert werden; spezielle Installationen sind nicht erforderlich,
- MS Project ist für kmU bezahlbar.

3.2.2 Der Projektplan

Der SPU-Projektplan ist ein MS Project-Projekt. Es besteht im wesentlichen aus einer Aufzählung von Teilaufgaben. Diese werden in Sammelaufgaben zusammengefaßt. Eine Beschreibung der Attribute erfolgt in Anhang B. Zum Teil werden sie direkt zur Projektplanung benötigt, z.B. Aufwände und Termine, zum Teil sind sie zur automatischen Kontrolle des Workflows erforderlich, z.B. Typen der Eingabedokumente. Die Struktur der zur Beschreibung von Aufgaben definierten Klasse »Aufgabe« ist in Anhang B dargestellt.

3.2.3 Ein- und Ausgabedokumente

Für jede Aufgabe im Workflow sind im Projektplan Ein- und Ausgabedokumente festgelegt. Die Festlegung erfolgt über Abkürzungen in besonderen Textfeldern. Zu jedem Dokumenttyp befindet sich in der Workflow-Komponente ein Musterdokument, das beim Start des Workflows kopiert und im weiteren Verlauf des Projektes modifiziert wird. Die Musterdokumente sollten den im Unternehmen gültigen ISO 9000 Qualitätssicherungsverfahren unterliegen.

Folgende Dokumenttypen werden im SPU-Workflow benutzt:

AP	Abnahmeprotokoll
AS	Aufgabenstellung
BE	Bericht
EW	Entwicklungsumgebung & Beschreibung
DD	Data Dictionary
IM	Implementierung
IPROT	Integrationsprototyp
PROD	Integrationsprodukt dieser Iteration (Globale Konfiguration)
Proto	Prototyp
TP	Testprotokoll
SYS	System entsprechend lokaler Konfiguration
	(Lauffähiges System + Make-Vorschriften)
PP	Projekt Plan
OS	Objekt Spezifikation (z.B. in ROSE)
	Enthält:
	- Architekturbeschreibung
	- Klassendiagramme
	- Objektdiagramme
	- Kategoriediagramme
TS	Test Spezifikation
ZM	Zeitmeldung (zur automatischen Aktualisierung des Plans)

3.2.4 Anbindung an das Workflowmanagenent

Viele der Daten, die die Projektplanung benötigt, benötigt auch das Workflowmanagement. Es liegt deshalb nahe, beide auf derselben Datenbasis aufzusetzen. Für das Workflowmanagement wurde Lotus Notes ausgewählt. Als Schnittstelle zwischen dem Projektplan in MS Project und dem Workflowmanagement in Notes war OLE vorgesehen. OLE bietet den Zugriff auf alle Objekte, die in einer anderen Applikation gespeichert sind. Somit hätte das Workflowmanagement vollen Zugriff auf die Informationen im Projektplan erhalten. Leider stellte sich während der Implementierung heraus, daß die OLE-Schnittstelle aufgrund

eines Fehlers in Notes nicht funktioniert. Es mußte deshalb ein Workaround mit Import- und Exportdateien erstellt werden.

Die Schnittstellendefinition zwischen der Projektverwaltung und dem Workflowmanagement ist im Anhang C zu finden.

3.2.5 Ansichten

Die große Zahl von Attributen, die jeder Aufgabe zugeordnet sind, macht es erforderlich, nur den Teil an Attributen anzuzeigen, die in der aktuellen Phase der Projektverwaltung erforderlich ist. Dieses wird durch mehrere MS Project-Tabellen erreicht, die im SPU-Musterprojektplan abgelegt sind.

- SPU Workflow: Dient in der ersten Phase zur Kontrolle und Modifikation des Workflows,
- SPU Projekt planen: Dient zur Planung des Projektes nach Festlegung des Workflows,
- SPU Projekt verfolgen: Dient zur Projektverfolgung.

3.2.6 Funktionen

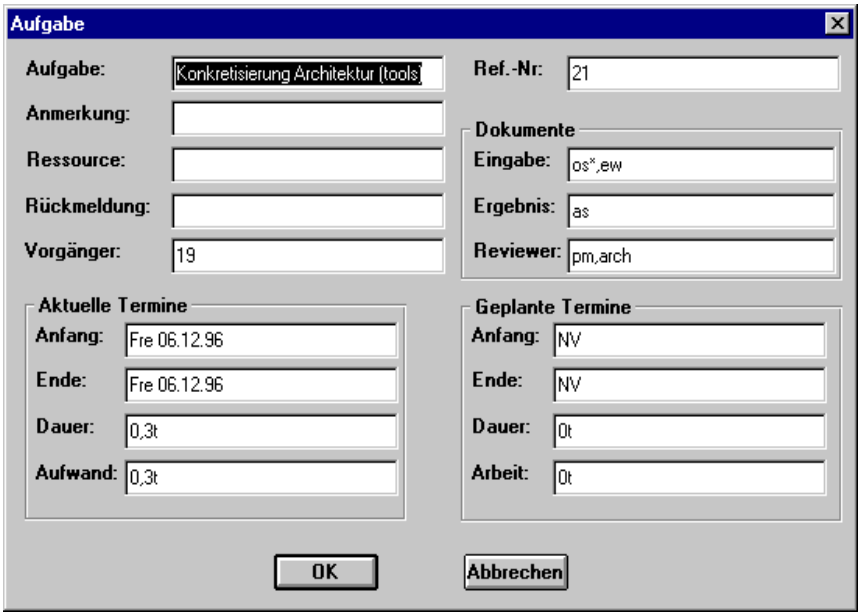
Dem Projektleiter stehen folgende Supportfunktionen zur Verfügung, die ihm die Arbeit erleichtern sollen:

- Verantwortlichen setzen: Setzt in das Feld Ressource einen eingegebenen Namen, wenn im Feld Verantwortlicher eine bestimmte Rolle eingetragen ist.
- Dauer nach Aufwand kopieren: Kopiert die von MS Project anhand des Start- und Endedatums berechnete Dauer in den Aufwand.

3.2.7 Masken

Eine spezielle Maske, die mit Hilfe einer besonderen Tastenkombination aufzurufen ist, faßt alle im Rahmen von SPU wichtigen Informationen zu einer Aufgabe in einem Formular zusammen. Die Maske kann sowohl zum Anzeigen als auch zum Ändern der Attribute genutzt werden.

Bild 15
MS Project-Maske
»Aufgabe«



4 Werkzeugunterstützung für den SPU-Entwicklungsprozeß

Sowohl auf dem Gebiet der objektorientierten Methoden als auch auf dem der unterstützenden Werkzeuge findet z.Z. eine beschleunigte Entwicklung statt. Deshalb können Bewertungen bzw. Empfehlungen von Entwicklungswerkzeugen nur eine temporäre Gültigkeit besitzen. Zusätzlich ist bei den Werkzeugen zu bedenken, daß diese die entsprechenden Methoden meist nicht vollständig unterstützen, so daß man im Endeffekt nicht in einer bestimmten objektorientierten Methode, sondern mit einer Variante einer solchen entwickelt, die sich beim Umstieg auf eine neuere Version des Werkzeuges verändern kann.

Der SPU-Entwicklungsprozeß (SPU-EP) als ein inkrementeller, iterativer Prozeß, der eine Folge von Prototypen liefert, an denen der Projektfortschritt gemessen werden kann, sollte idealerweise in allen drei Phasen der Analyse, des Entwurf und der Implementation durch ein einziges Werkzeug durchgängig unterstützt werden. Derartige Werkzeuge existieren jedoch derzeit noch nicht [CKW96], so daß die gewünschte Funktionalität durch eine Kombination von Werkzeugen realisiert werden muß.

Für den SPU-Prototypen wurden für die Phasen Analyse und Entwurf das Werkzeug Rational/Rose und für die Implementationsphase SNiFF+ von TakeFive Software gewählt. Im folgenden werden die Gründe für diese Entscheidung und notwendige Anpassungen aufgrund spezieller Anforderungen des SPU-EPs näher erläutert. Daran anschließend werden die Prozesse bei der Arbeit mit der Entwicklungsumgebung beschrieben.

4.1 Werkzeugunterstützung für Analyse und Entwurf

Generell unterscheidet man bei objektorientierten Analyse- und Entwurfswerkzeugen zwischen sog. Meta-Werkzeugen, bei denen die Benutzer eine große Freiheit bei der Anpassung sowohl der verwendeten objektorientierten Methode als auch bei der Anpassung der Code-Generierung haben, und speziellen Werkzeugen, die für die Code-Generierung in einer bestimmten Programmiersprache ausgelegt sind. Eine Modifikation des Verhaltens der Entwurfsmethode ist bei dieser Art von Werkzeugen nicht möglich.

Zum Zeitpunkt unserer Untersuchungen war die Code-Generierung in dem getesteten Meta-Werkzeug sehr ineffizient, so daß wir uns für ein Entwurfswerkzeug für eine spezielle Programmiersprache (C++ und Java) entschieden haben.

Rational/Rose für C++ bzw. Rational/Rose Java (im folgenden Rational/Rose genannt) sind graphische Modellierungswerkzeuge, die aus Entwurfsdiagrammen Klassenrahmen in den Sprachen C++ bzw. Java erzeugen können. Da Rational/Rose die Philosophie des »Round-Trip-Engineering« nach Booch [Boo94] zugrundeliegt, ist die Unterstützung des SPU-EPs als inkrementellem, iterativem Prozeß gewährleistet. Rational/Rose unterstützt die 3 gängigsten objektorientierten Entwurfstechniken:

- »Unified Modelling Language« [UML97],
- Entwurfsmethode nach Booch [Boo94],
- die OMT-Methode nach Rumbaugh u.a. [RBP91].

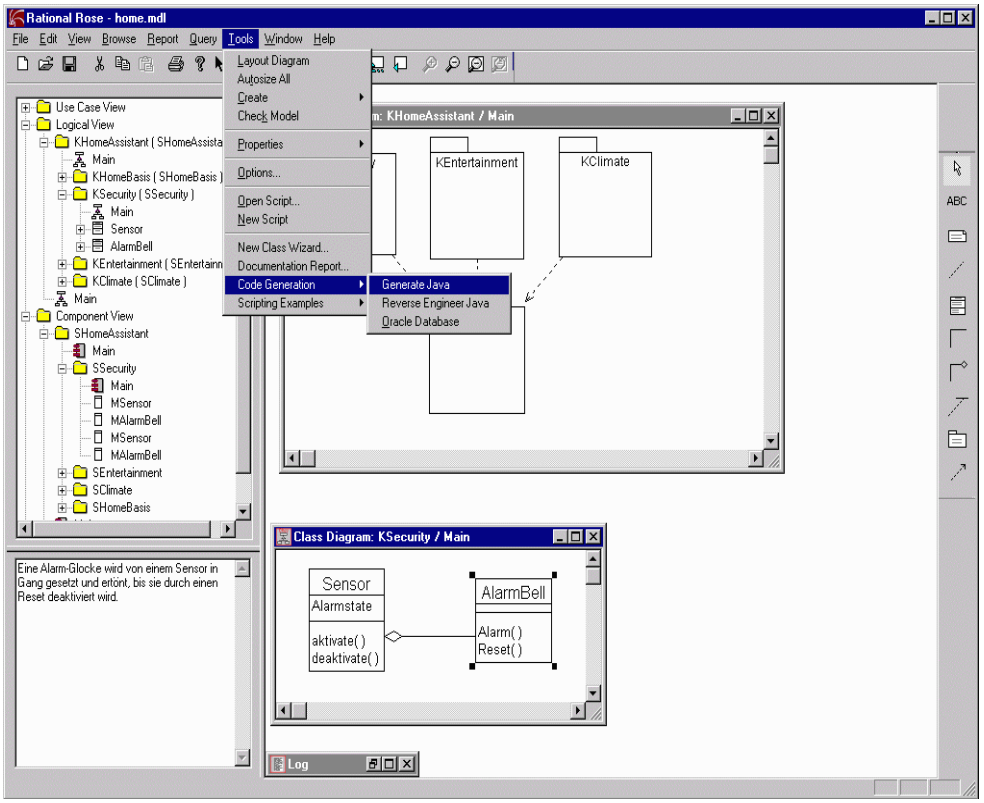
Rational/Rose entspricht in wesentlichen Teilen den im ersten Meilenstein aufgestellten Anforderungen:

- Rational/Rose bietet mit sog. »Klassenkategorien« in der Booch-Methode oder »Paketen« in der UML eine Möglichkeit, den logischen Entwurf weiter zu strukturieren und somit übersichtlich zu gestalten.
- Grobstrukturierungen auf der logischen Ebene wie Klassenkategorien können auf entsprechende Strukturen der physikalischen Ebene (Subsysteme, Komponenten) abgebildet werden. Die Klassen selbst werden auf sog. »Module« abgebildet. Ein Modul besteht wiederum aus einer Spezifikation und einem Körper, die bei der Code-Generierung in Header- und Programm-Dateien abgebildet werden.
- Rational/Rose unterstützt den Anschluß beliebiger Versionsverwaltungssysteme mittels Shell-Programmen. Sowohl Klassenkategorien als auch Subsysteme können als sog. »controlled units« spezifiziert werden. Diese können dann separat durch ein Werkzeug zur Versionsverwaltung administriert werden, so daß mehrere Entwickler parallel an Teilen des Modells arbeiten können.
- Die Generierung von Klassenrahmen selbst ist hinreichend gut konfigurierbar, so daß die Nutzung objektorientierter Datenbanken oder Klassenbibliotheken schon bei der Generierung der Klassenrahmen berücksichtigt werden kann.
- Rational/Rose unterstützt einen zyklischen Entwicklungsprozeß. Die generierten Code-Rahmen besitzen sog. »preserved regions«. Der darin eingeschlossene Code bleibt auch bei der erneuten Generierung von Code-Rahmen erhalten.
  Erfolgreich getesteter Code kann dann über die »CodeCycle«-Funktionalität des Rose-Analyzers einem Reverse Engineering unterzogen werden, um die in der Implementationsphase hinzugefügten Erweiterungen auch in das logische Modell zu übernehmen. Ein »Model Differencer« unterstützt dabei den Vergleich von ursprünglichem und neuem Modell.
- Rational/Rose kann mit der Version 4.0 sowohl als OLE Automation Server als auch als Controller eingesetzt werden. Damit ist es u.a. möglich, Rose-

Modelle direkt in den Ergebnisdokumenten der Lotus-Notes-Workflow-Komponente zu hinterlegen und zu bearbeiten.

- Neben Rational/Rose C++ und Rational/Rose Java existieren weitere Versionen für die Unterstützung der Implementation in Smalltalk, Visual Basic und für Entwicklungsumgebungen wie SQL Windows, Forté und Power Builder.

Bild 16
Rational Rose-Bildschirmausschnitt



Der in Bild 16 dargestellte Bildschirmausschnitt zeigt ein Rational Rose-Fenster mit Navigator zur Auswahl der Komponenten, geöffnetem Paket- und Klassendiagramm und dem aufgeklappten Menü zur Erzeugung von Code.

4.2 Werkzeugunterstützung für die Implementation

SNiFF+ ist eine Multiplattformentwicklungsumgebung, die als offener und erweiterbarer Werkzeug-Rahmen konzipiert ist. Großer Wert wurde auf Möglichkeiten der Visualisierung, der Strukturierung der zu bearbeitenden Dateien und auf die Unterstützung kooperativen Arbeitens gelegt.

SNiFF ist durch den Benutzer einfach erweiterbar und bietet viele Werkzeuge für die Programmierung wie »Differenzer«, »Konfigurationsmanager«, »Retriever«, Hierarchie-, Klassen- und Symbol-»Browser«. Da die Browser mit dem verwendeten Editor kooperieren, kann analog zur Arbeit in den Entwurfswerkzeugen auf Objekt-Niveau navigiert werden. Bei der Auswahl eines Objekts in einem Browser wird im Editor-Fenster der diesem Objekt zugeordnete Quelltext angezeigt. Außerdem besitzt SNiFF konfigurierbare Schnittstellen zu verschiedenen Compilern, Debuggern, Versionsverwaltungssystemen und zum Make-System. Sprachspezifische Unterstützungen werden z.B. für die folgenden Sprachen angeboten: C++, Java, Fortran, IDL.

Von besonderem Interesse ist SNiFF jedoch, weil es Konzepte wie Projekte und getrennte Arbeitsbereiche unterstützt.

Projekte und Projekthierarchien

Als Hauptstrukturierungselement zur Handhabung einer großen Anzahl von Dateien wird das Konzept »Projekt« verwendet. Für jedes Projekt existiert eine Beschreibungsdatei, die Informationen über die im Projekt verwalteten Arten von Dateien und über die verwalteten Dateien selbst enthält. Ein Projekt kann Dateien beliebigen Typs verwalten. Datei-Typen können vom Nutzer definiert werden, wobei jeder Datei-Typ durch eine entsprechende Attribut-Struktur beschrieben wird. Darin können z.B. die Werkzeuge spezifiziert werden, mit denen eine Datei eines bestimmten Typs geöffnet werden soll. Jedem Projekt ist ein Verzeichnis zugeordnet. Projekte selbst können wieder Unterprojekte enthalten, so daß eine Strukturierung des Systems in Analogie zur Grobstruktur auf der Entwurfsebene vorgenommen werden kann. Zur Manipulation und Navigation in den zu einem Projekt gehörenden Dateien existiert ein spezielles Werkzeug, der Projekt-Editor. In ihm können die Dateien unter verschiedenen Gesichtspunkten angezeigt werden, z.B. nur Dateien eines bestimmten Typs, eines bestimmten Unterprojektes oder Dateien, die sich im lokalen bzw. globalen Arbeitsbereich (siehe unten) befinden. Der Projekt-Editor ist mit dem Editor gekoppelt, so daß Dateien auch über ihn in den Editor geladen werden können.

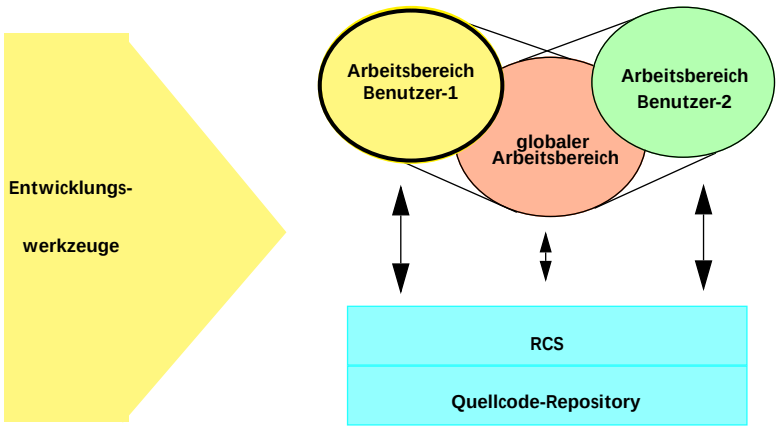
Arbeitsbereiche

Unter einem Arbeitsbereich (siehe Bild 16) wird im folgenden eine Verzeichnisstruktur verstanden, in der ein bestimmtes Projekt bearbeitet wird. In SNiFF wird zwischen privaten Arbeitsbereichen, die einzelnen Entwicklern zugeordnet sind, und globalen Arbeitsbereichen unterschieden. Globale Arbeitsbereiche[3] sind der Rolle des Integrators zugeordnet und enthalten alle für den Entwicklungsprozeß relevanten Artefakte. In den privaten Arbeitsbereichen der einzelnen Entwickler befinden sich nur Versionen von Artefakten, die in ihrem Bearbeitungszustand von der entsprechenden Version des globalen Arbeitsbereiches abweichen. Alle

3 Zu einem Projekt können mehrere globale Arbeitsbereiche verwaltet werden. Der Projektadministrator spezifiziert, in welcher Reihenfolge sich diese globalen Bereiche überlagern.

im Entwicklungsprozeß erzeugten Artefakte (bzw. deren Generierungsvorschriften, z.B. in Form von Makefiles) werden in einem gemeinsamen Repository mit einem Revisions-Kontroll-System verwaltet. Für den Prototypen der Entwicklungsumgebung wurde das Revisionskontrollsystem RCS[Tic86] verwendet.abschatten

Bild 17
Struktur der Arbeitsbereiche



Eine derartige Aufteilung in Arbeitsbereiche sichert eine unabhängige und stabile Arbeit im SPU-EP. Durch die SPU unterstützt kann sich jeder Entwickler aus dem Repository Versionen von Artefakten reservieren und in seinen privaten Arbeitsbereich kopieren. Da die im lokalen Arbeitsbereich befindlichen Artefakte diejenigen im globalen Arbeitsbereich abschatten, können lokale Änderungen von Artefakten durchgeführt werden, ohne daß parallel arbeitende Entwickler dadurch in ihrer Arbeit beeinträchtigt werden. Die Konfiguration eines Arbeitsbereiches kann über einen in SNiFF integrierten Konfigurationsmanager verwaltet werden.

4.3 Kopplung der Werkzeuge im Entwicklungsprozeß

Aufgrund des offenen Charakters von SNiFF wird dieses Werkzeug als Master-Werkzeug im Entwicklungsprozeß verwendet. Bei der Realisierung des Prototypen standen keine Versionen von SNiFF und Rational/Rose zur Verfügung, die über Protokolle wie Tooltalk oder OLE angesprochen werden konnten. Die Kopplung der Werkzeuge erfolgte deshalb nur implizit über die Verwaltung der im Entwurfsprozeß erzeugten Artefakte in einem gemeinsamen Repository, welches über Menüs im SNiFF-Projekt-Editor verwaltet wird.

Bild 18
Der SNiFF-Projekteditor

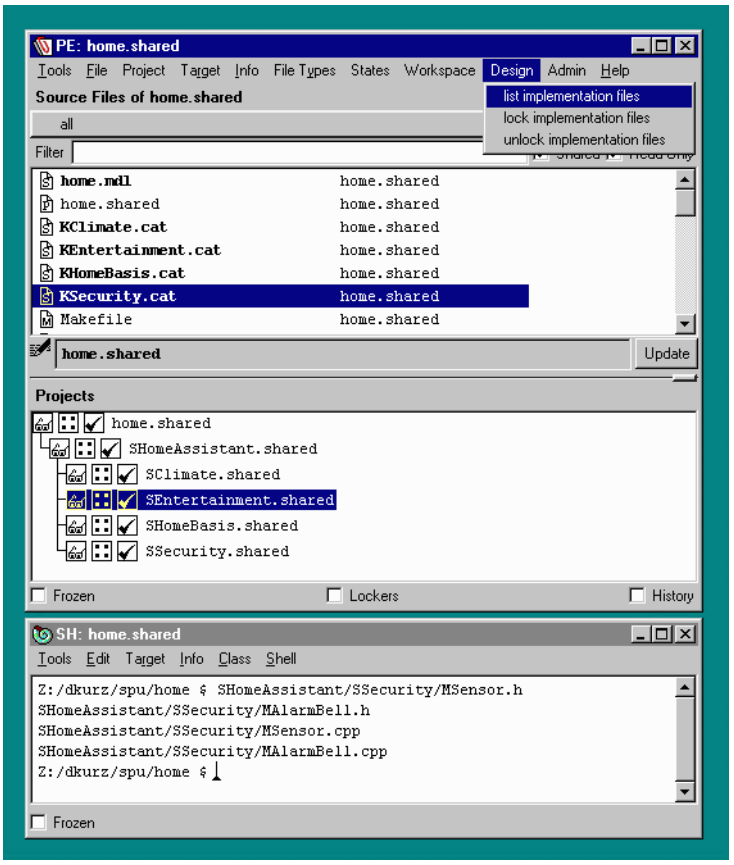


Bild 18 zeigt den SNiFF-Projekt-Editor mit aufgeklapptem Menü zur Reservierung aller zu einem Modell gehörenden Implementationsartefakte.

Bei der Entwicklung reservieren sich einzelne Entwickler Artefakte und bearbeiten diese in den privaten Arbeitsbereichen. Als Folge dieser asynchronen Entwicklung entsteht bei jedem Bearbeiter ein anderer Entwicklungszustand des Systems. Um die Integration der lokal modifizierten Artefakte zu vereinfachen, wurden in Analogie zu [Lam94] für die Revisionen eines Artefakts die folgenden Bearbeitungszustände eingeführt:

- »Save«: Der Bearbeitungszustand wurde vom Bearbeiter als private Sicherung im Repository abgespeichert.
- »Proposed«: Das Artefakt hat den lokalen Integrationstest des Bearbeiters überstanden und wird für die anderen Bearbeiter freigegeben.
- »Published«: Das Artefakt hat den globalen Integrationstest bestanden.

Um die Arbeit mit Bearbeitungszuständen möglichst einfach zu gestalten, wurde der Projekt-Editor von SNiFF, über den alle Dateien eines Projektes verwal-

tet werden, um ein spezielles Menü zur Verwaltung von Bearbeitungszuständen erweitert. Damit können Bearbeitungszustände über Maus-Aktionen sowohl für einzelne Dateien als auch für alle Dateien eines Projektes geändert werden. Außerdem wurde eine Prozedur implementiert und über einen Menü-Eintrag zur Verfügung gestellt, die die von einem speziellen Nutzer zuletzt bearbeiteten Revisionen eines bestimmten Bearbeitungszustands in einem Fenster ausgibt.

Aktualisierung der Arbeitsbereiche

Da in SNiFF die Arbeit mit Bearbeitungszuständen nicht unterstützt wird, konnte die im Werkzeug implementierte Methode zur Aktualisierung der Arbeitsbereiche nicht verwendet werden. Daher wurde eine eigene Prozedur implementiert, die die Arbeitsbereiche mit den letzten Revisionen eines bestimmten Bearbeitungszustandes aktualisiert. Um sowohl lokale als auch globale Integrationstests durchführen zu können, wird bei der Aktualisierung der Arbeitsbereiche zwischen Dateien, die zuletzt vom Nutzer (»user«) bearbeitet wurden, und Dateien anderer Nutzer (»other«) unterschieden. Die Aktualisierungsstrategie für einen lokalen Integrationstest besteht dann darin, alle eigenen Dateien mit einem Bearbeitungszustand besser oder gleich »Save« und alle anderen Dateien mit einem Bearbeitungszustand besser oder gleich »Proposed« in den privaten Arbeitsbereich zu kopieren. Für die Auswahl der zur Aktualisierung des Arbeitsbereiches verwendeten Bearbeitungszustände wurde ein eigenes Menü zur Verfügung gestellt.

Im folgenden Abschnitt werden die Prozesse beschrieben, die durchgeführt werden müssen, um die lokalen Entwicklungen wieder zu integrieren.

4.4 Prozesse in der Entwicklungsumgebung

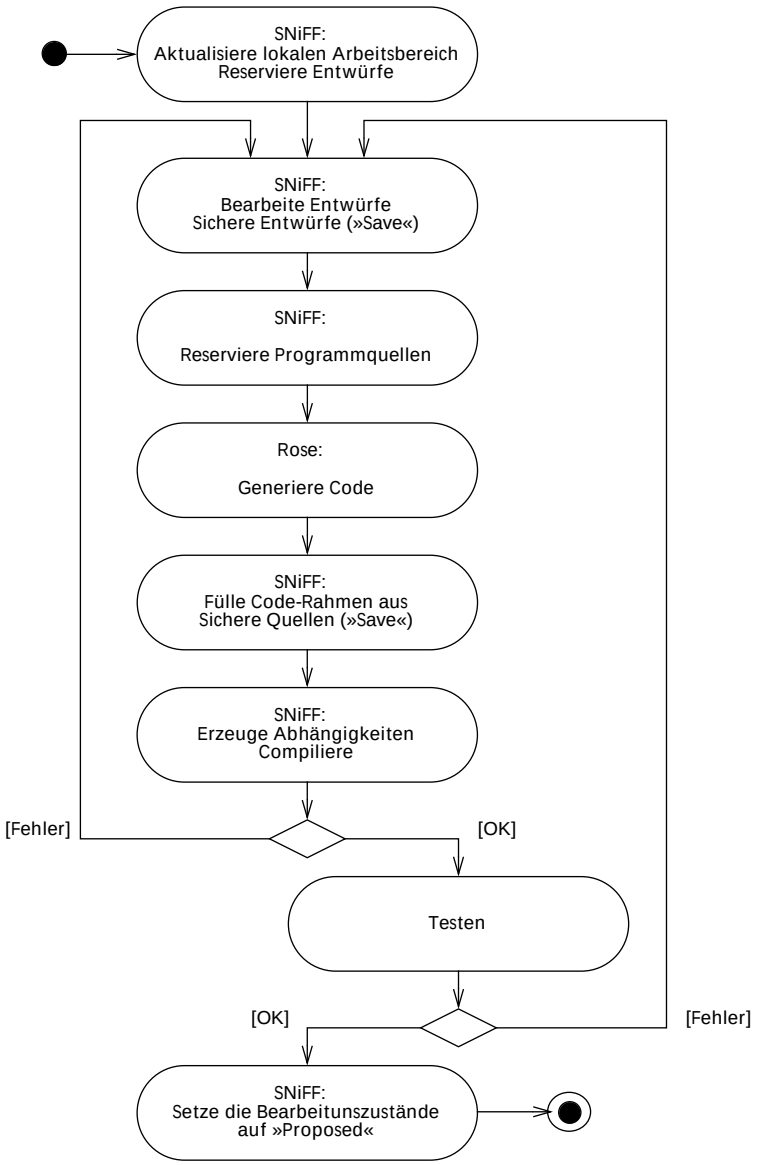
Im folgenden sollen einige der bei der Arbeit mit der Entwicklungsumgebung ablaufenden Prozesse detailliert beschrieben werden. Zur Darstellung werden »activity diagrams« entsprechend der UML-Notation [UML97] benutzt.

4.4.1 Prozesse bei der Software-Entwicklung

In Bild 19 ist die Arbeit mit der Entwicklungsumgebung schematisch dargestellt. Nach dem Starten von SNiFF als Steuer-Zentrum der Entwicklungsumgebung wird der private Arbeitsbereich über ein projektspezifisches Menü in SNiFF aktualisiert; jeweils die jüngsten Revisionen einer Datei mit einem bestimmten Bearbeitungsstatus werden bereitgestellt. Danach wird Rational/Rose zur Bearbeitung der Entwürfe gestartet. Die zu bearbeitenden Teilmodelle werden über die RCS-Schnittstelle von SNiFF reserviert. Daran anschließend können die Modelle bearbeitet und im Repository mit dem Zustand »Save« gesichert werden. Nun werden diejenigen Programmquellen reserviert, für die aus den bearbeiteten Entwürfen Code generiert werden soll. Dazu wurde ein zusätzliches Menü in SNiFF eingebaut, über das alle zu einer Rose-Kategorie gehörenden

Implementationsdateien ausgecheckt werden können. Der generierte Code muß nun manuell weiterverarbeitet werden, z.B. müssen Rümpfe neu deklarierter Funktionen ausgefüllt werden. Zur Sicherung werden abschließend die modifizierten Programmquellen mit dem Bearbeitungszustand »Save« ins Repository geschrieben. Der Aufruf von »Make« aus SNiFF heraus erzeugt eine neue Version des Programms. Sind die vom Entwickler durchgeführten Tests erfolgreich, werden die modifizierten Dateien zum Integrationstest freigegeben, indem deren Bearbeitungszustände auf den Wert »Proposed« gesetzt werden. Beim Fehlschlagen des Compiliervorgangs oder der Tests wird der Entwicklungszyklus erneut durchlaufen, evtl. mit Überspringen einzelner Phasen.

Bild 19
Aktivitätaten in der Entwicklungsumgebung

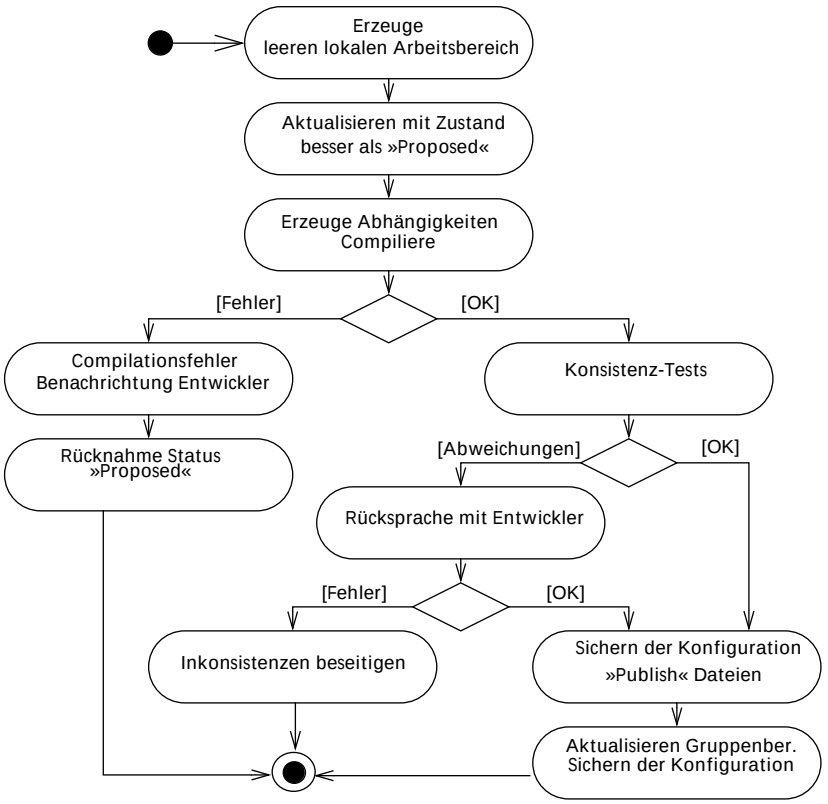


4.4.2 Prozesse bei der Integration paralleler Entwicklungsarbeiten

Um das asynchrone Arbeiten der einzelnen Entwickler zu synchronisieren, muß der Gruppenarbeitsbereich vom Integrator gepflegt und aktualisiert werden. Die dabei durchgeführten Aktivitäten sind in Bild 20 dargestellt. Zunächst wird zu

definierten Zeitpunkten ein leerer privater Arbeitsbereich erzeugt. Dieser wird mit den Revisionen aller Artefakte aktualisiert, die einen Bearbeitungszustand besser oder gleich »Proposed« haben. Danach werden alle Aktionen durchgeführt, die für die Vorbereitung der Erzeugung eines neuen Prototypen notwendig sind. Dazu zählen z.B. bei C++-basierten Projekten die Erzeugung von Abhängigkeiten für den Make-Mechanismus. Anschließend wird ein neuer Prototyp generiert. Treten bei der Compilation Fehler auf, so wird der Prozeß abgebrochen, die Entwickler werden über E-Mail benachrichtigt. Andernfalls laufen die Konsistenz-Tests ab. Der Prozeß wird abgebrochen und der Projektadministrator wird benachrichtigt, wenn die Tests Veränderungen zur vorherigen Version zeigen. In diesem Fall müssen die Entwickler entscheiden, ob die Abweichungen gewollt sind. Ist die Abweichung eine Folge der Weiterentwicklung des Systems, wird der Gruppenbereich aktualisiert. Andernfalls versuchen die Bearbeiter, die Inkonsistenz zu beseitigen. Zur Aktualisierung des Gruppenbereichs wird die Konfiguration des Arbeitsbereiches im Repository mit dem Status »Published« gesichert, und abschließend werden die Bearbeiter über die Aktualisierung des globalen Arbeitsbereiches und die erfolgreiche Integration ihrer Änderungen informiert.

Bild 20
Aktualisieren des
Gruppenbereiches



Literaturverzeichnis

[Boo94] G. Booch, »Object Oriented Analysis and Design« , Second Edition, Benjamin/Cummings, 1994

[Bro94] Brockschmidt, Kraig, »Inside OLE 2 - the Fast Track to Building Powerful Object-Oriented Applications«, Microsoft Press, Redmond, 1994

[CKW96] I. Classen, E.U. Kriegel, S. Wasserroth, »Object-Oriented Software Development - Experiences and Trends«, Fraunhofer ISST, 32/1996

[Jac92] Ivar Jacobson, »Object-Oriented Software Engineering, A Use Case Driven Approach«, Addison-Wesley, 1992

[Kri+96] E. Ulrich Kriegel, J. Altenhein, M. Schlösser-Fassbender, P. Thierse, »Eine Softwareproduktionsumgebung für kleine und mittelständische Unternehmen«, Fraunhofer ISST, 36/1996

[Lam94] A. Lampen,»Attributierte Softwareobjekte als Basis zur Datenmodellierung in Software-Entwicklungsumgebungen« , Dissertation, TU-Berlin, 1994

[RBP91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, »Object-Oriented Modeling and Design« , Prentice-Hall, 1991

[Ros97] Rational Software Corp., http://www.rational.com

[Tai95] S. Tai,»Grobstrukturierung objektorientierter Modelle zur Unterstützung arbeitsteiliger Softwareentwicklungsprozesse« , Diplomarbeit an der TU Berlin, Fachbereich Informatik, 1995.

[Tic86] W. F. Tichy, »RCS - A System for Version Control« , Software--Practice & Experience 15, 7 (July 1985), 637-654.

[Too93] »The tooltalk service - an interoperability solution«, Technical Report, Sunsoft Press Prentice Hall, 1993

[SNi95] TakeFive Software GmbH, »SNiFF+ User‘s Guide and Reference« , SNiFF-URG-002, 1996, http://www.takefive.com

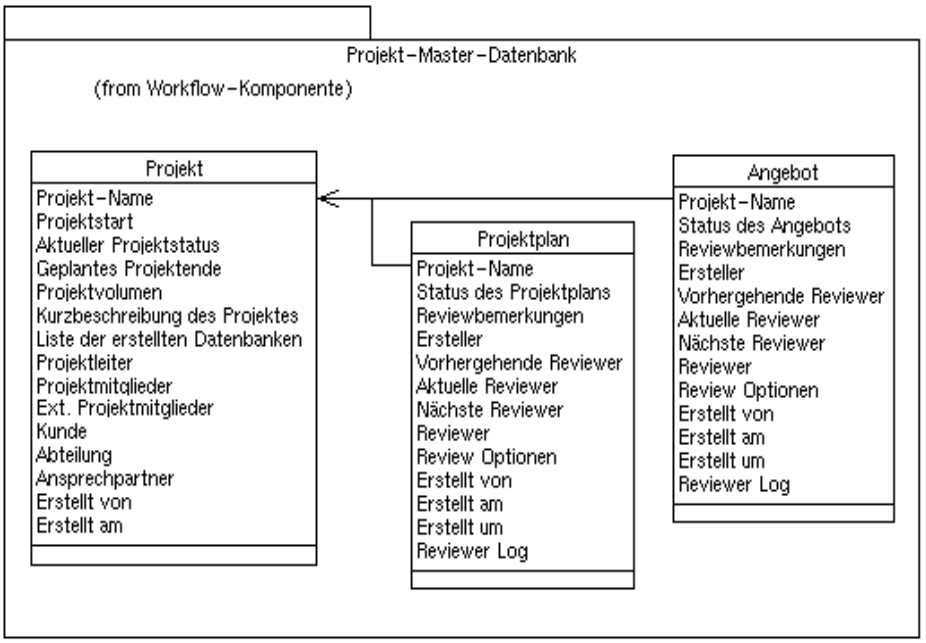
[UML97] »Unified Modelling Language: Notation Guide«,ad/97-01-09, Rational Software Corporation, 1997

Anhang

A Spezifikation von Objekten im Workflow- und Dokumentenmanagement

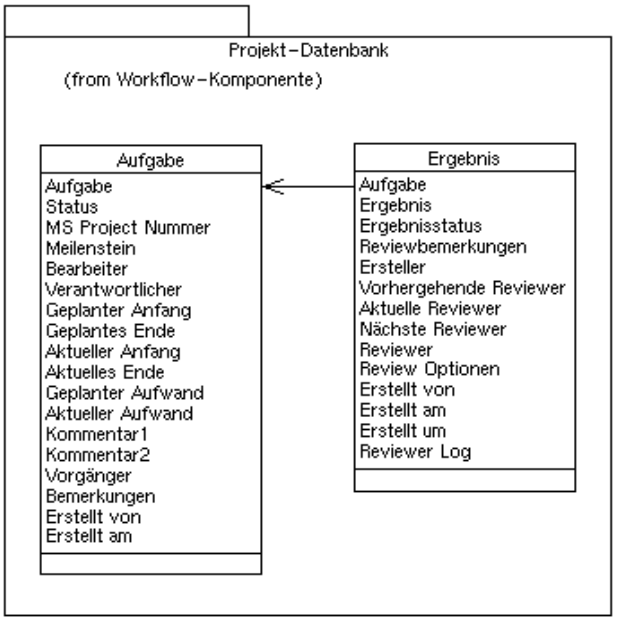
A.1 Das Paket »Projekt-Master-Datenbank«

Bild 21
Projekt-Master-Datenbank



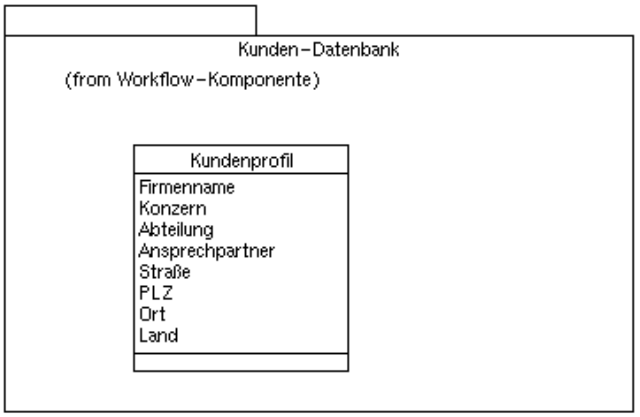
A.2 Das Paket »Projekt-Datenbank«

Bild 22
Projekt-Datenbank



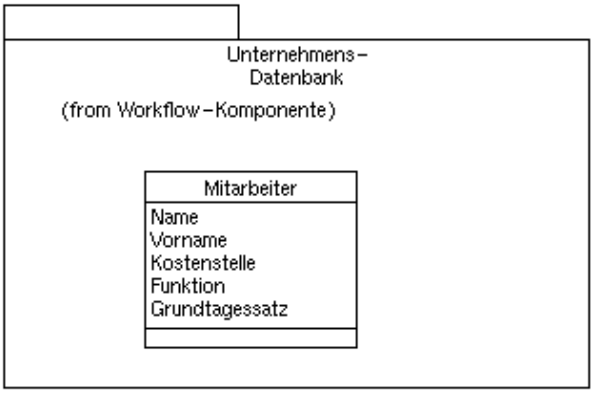
A.3 Das Paket »Kunden-Datenbank«

Bild 23
Kunden-Datenbank



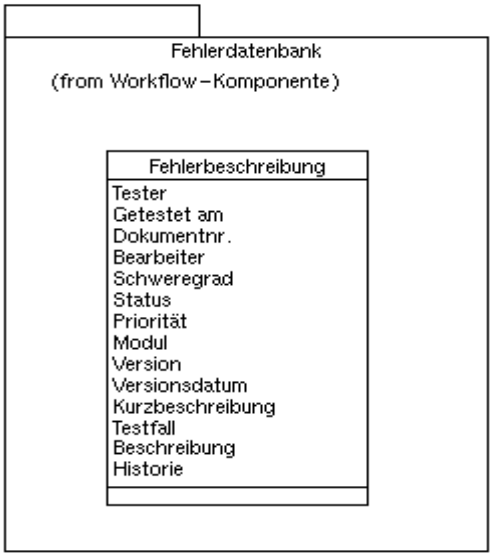
A.4 Das Paket »Unternehmens-Datenbank«

Bild 24
Unternehmens-Daten-
bank



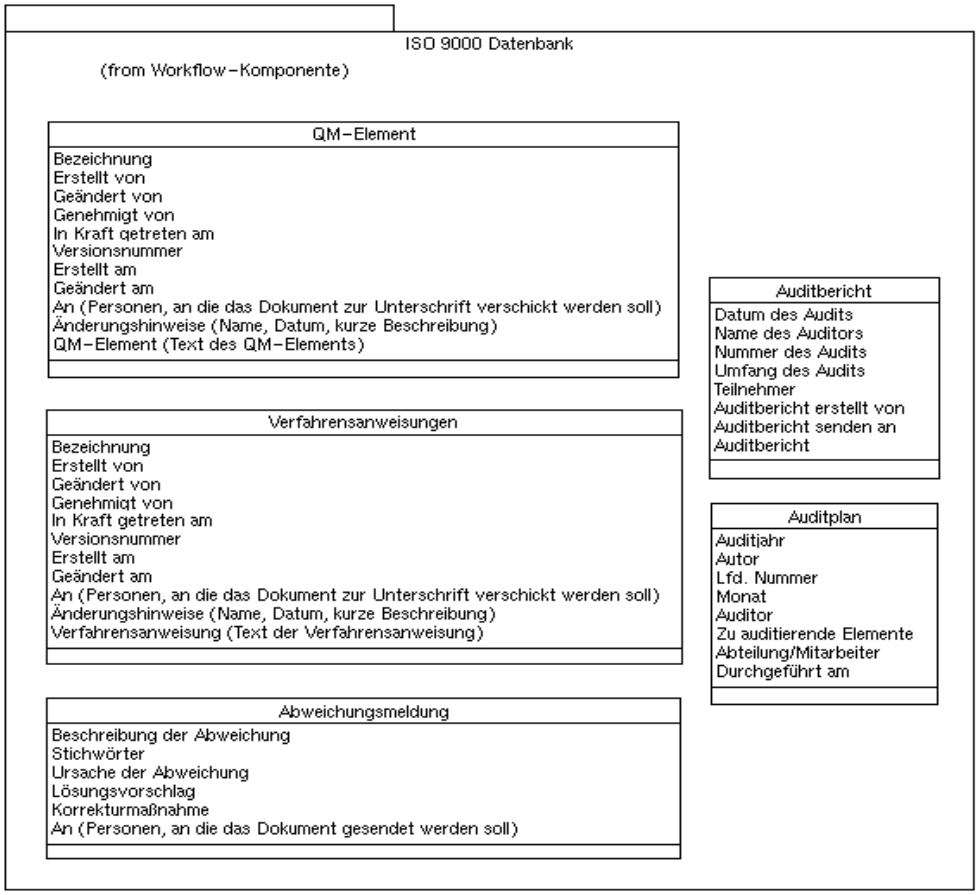
A.5 Das Paket »Fehlerdatenbank«

Bild 25
Fehlerdatenbank



A.6 Das Paket »ISO 9000 Datenbank«

Bild 26
ISO 9000 Datenbank



Zu den einzelnen Klassen werden die folgenden Buttons und Aktionen angeboten:

A.6.1 QM-Elemente

Buttons

- Dokument zur Unterschrift versenden (verschickt das Dokument an die Personen, die im Feld »An« angegeben wurden)

Aktionen

- Dokument verlassen
- Dokument bearbeiten
- Dokument drucken
- Dokument unterschreiben (setzt elektronische Unterschrift. Falls bereits ein unterschriebenes Dokument existiert, wird dieses ins Archiv gestellt und das unterschriebene Dokument wird als genehmigt gesetzt. Alle Mitglieder der Gruppe ISO 9000 werden per Mail benachrichtigt.)

A.6.2 Verfahrensanweisungen

Buttons

- Dokument zur Unterschrift versenden (verschickt das Dokument an die Personen, die im Feld »An« angegeben wurden)
- Formular einfügen (fügt ein Formular bzw. eine Schablone (z.B. Angebot, Projektplan) in das Dokument ein)

Aktionen

- Dokument verlassen
- Dokument bearbeiten
- Dokument drucken
- Dokument unterschreiben (setzt elektronische Unterschrift. Falls bereits ein unterschriebenes Dokument existiert, wird dieses ins Archiv gestellt und das unterschriebene Dokument wird als genehmigt gesetzt. Alle Mitglieder der Gruppe ISO 9000 werden per Mail benachrichtigt.)

A.6.3 Abweichungsmeldung

Aktionen

- Dokument verlassen
- Dokument speichern
- Dokument drucken
- Dokument versenden

A.6.4 Auditbericht

Aktionen

- Auditbericht verlassen

- Auditbericht bearbeiten
- Auditbericht drucken
- Auditbericht versenden

A.6.5 Auditplan

Aktionen

- Auditplan verlassen
- Auditplan speichern
- Auditplan drucken

B Spezifikation der Objekte in der Realisierung des Management-Prozesses

Speziell für die Verwendung von MS Project mußte die im Anhang in Abschnitt A.2 definierte Klasse »Aufgabe« wie folgt modifiziert werden. Anhang C definiert die entsprechende Schnittstelle für den Austausch zwischen Notes und MS Project.

Bild 27
Attribute einer Aufgabe

Aufgabe
UniqueId (Wird von MS Project vergeben und kann nicht geändert werden) ID Aufgabe Referenznummer im Workflow des Meilensteinpapiers [1] Geplanter Anfang Geplantes Ende Geplanter Aufwand Anmerkung Aktueller Anfang Aktuelles Ende Aktueller Aufwand Rückmeldung Verantwortlicher Ressource (Der Ressourcenname entspricht der E-Mail-Adresse des Bearbeiters) Reviewer Vorgänger Typen der Eingabedokumente Typen der Ausgabedokumente

C Schnittstellenabsprache zwischen MS Project und Lotus Notes im SPU Prototypen

C.1 Einleitung

Für den SPU-Prototypen wurde der SPU-Workflow aus dem Meilensteinbericht 1 in MS Project 4.1 abgebildet. Dabei wurden die Daten des Projektmanagements mit den Daten für den Workflow kombiniert. Der dabei entstandene Projektplan steuert die auf Lotus Notes basierende Workflow-Komponente des SPU-Prototypen. Dadurch konnte sichergestellt werden, daß Workflow und Projektmanagement immer konsistent bleiben.

Dieser Abschnitt definiert die Schnittstelle zwischen dem Projektplan in MS Project und der Workflow-Komponente in Lotus Notes.

C.2 Notes Datenbanken

Es gibt drei Notes Datenbanken, die für diese Schnittstelle von Bedeutung sind:

- Die SPU Datenbank enthält den Projektplan und alle Dokumente, die benötigt oder erstellt werden, bevor das Projekt gestartet wird, d.h., wenn der Projektplan festliegt und der SPU-Workflow beginnt.
- Eine Projekt-Datenbank wird für jedes Projekt eingerichtet. Diese enthält die Entwicklungs- und Aufgabendokumente (s.u.) für ein Projekt.
- Die Template-Datenbank enthält leere Templates für alle Entwicklungsdokumente (s.u.). Die Templates werden bei Projektstart in die entsprechende Projektdatenbank kopiert.

Zusätzlich können Datenbanken mit z.B. Kundennamen existieren, die aber für die Schnittstelle ohne Bedeutung sind.

C.3 Dokumenttypen

Es gibt zwei Dokumenttypen in den Notes-Projektdatenbanken, Entwicklungsdokumente (E-Dok) und Aufgabendokumente (A-Dok).

E-Doks sind OLE- oder Notes-RTF-Objekte. Sie werden bei Projektstart als Kopien der Templates angelegt (Offen: Automatischer Eintrag z.B. des Projektnamens). Es wird während der gesamten Projektlaufzeit auf demselben Objekt gearbeitet. Optional kann bei Abschluß einer modifizierenden Aufgabe eine Kopie abgelegt werden (Versionsverwaltung).

A-Doks fassen die Daten über Vorgängeraufgaben, Input- und Output-Dokumente und Zeitplanung sowie Zeiterfassung zusammen. Beim Anlegen eines Projektes wird für jede Aufgabe des MS Project-Plans ein A-Dok erstellt. Dieses Dokument wird dem Bearbeiter zugesandt, sobald alle Vorgängeraufgaben abgeschlossen sind. Es gibt eine besonders markierte erste Aufgabe, die den ganzen Workflow startet. Attribute in A-Doks werden bei jeder Änderung mit dem Projektplan abgeglichen (z.B. Startdatum, Verantwortlicher, ...). A-Doks können aber nicht nach Projektstart neu erstellt oder gelöscht werden. Der Workflow basiert nur auf den Vorgänger-/Nachfolgerinformationen von MS-Project. Feste Starttermine können nicht eingegeben werden.

A-Doks bieten
- Links zu den Input- (read-only) und Output- (read-write) Dokumenten der Aufgabe,
- Felder, die alle Informationen zur Aufgabe aus MS Project anzeigen,
- Felder, um beim Abschluß der Aufgabe Informationen wieder in den Projektplan zu integrieren, sowie
- Buttons zur Initiierung eines Reviewzyklus für diese Aufgabe und zur Überführung der Aufgabe in den Zustand »Fertig«.

Aufgaben müssen keine Input- bzw. Output-Dokumente enthalten.

A-Doks für Meilensteine werden besonders behandelt, da sie zwar Vorgänger haben können und wiederum selbst Vorgänger sein können, aber keinen Bearbeiter haben. Sie gehen direkt vom Zustand »Warten« in den Zustand »Fertig« über.

Sammelaufgaben (Zusammenfassung mehrerer Aufgaben in einem Balken in MS-Project) werden durch zwei Meilenstein-A-Doks dargestellt. Der Beginn wird durch einen Meilenstein repräsentiert, der die Vorgänger der Aufgabe enthält und Vorgänger aller durch diese Sammelaufgabe zusammengefaßten Aufgaben ist. Der Ende-Meilenstein hat alle zusammengefaßten Aufgaben als Vorgänger und ist Vorgänger aller Aufgaben, die diese Sammelaufgabe als Vorgänger haben.

C.4 Datenattribute von A-Doks

A-Doks haben folgende Attribute, die als Notes-Items gespeichert werden. Wenn vorhanden, wird der MS Project-Name in Klammern [ ] angegeben. Ein »<« heißt, dieser Wert wird dem MS Project-Plan entnommen und ins A-Dok eingetragen, ein »>« heißt, er wird in den MS Project-Plan übernommen. Die Übernahme erfolgt, wenn der Projektleiter das ihm bei der Fertigstellung der Aufgabe zugesandte A-Dok liest. Mit »*« markierte Attribute werden nach der Erstellung des A-Doks nicht mehr modifiziert.

Bild 28
Attribute eines A-Doks

A-Dok
UniqueId [Einmalige Nummer]* Aufgabe < [Name] Geplanter Anfang < [Geplanter Anfang] Geplantes Ende < [Geplantes Ende] Geplanter Aufwand < [Geplanter Aufwand] Kommentar1 < [Text8] Aktueller Anfang > [Aktueller Anfang] Aktuelles Ende > [Aktuelles Ende] Aktueller Aufwand > [aktueller Aufwand] Kommentar2 > [Text6] Verantwortlicher < [Ressoucenname] (Der Ressourcenname entspricht der Lotusadresse des Bearbeiters) Bearbeiter < [?] Reviewer < abgeleitet von [Text3] Unique Ids der Vorgänger < [Einmalige Nr. Für Vorgänger] Zustand (»Warten«, »In Arbeit«, »Im Review«, »Fertig«) Bemerkungen (Notes RTF-Object) Ist Meilenstein [Meilenstein, Sammelvorgang]* Referenzen auf Input Dokumente, basierend auf [Text2]* Referenzen auf Output Dokumente, basierend auf [Text4]*

Als Buttons stehen zu Verfügung:

- Review: Leitet A-Dok den Reviewern zu,
- Fertig: Sichert Dokumente (eventuell in Versionsverwaltung), sendet A-Dok an Projektleiter, triggert Agent für Workflow,
- Save: Sichert den aktuellen Stand,
- In Projektplan übernehmen: Übernimmt die mit »>« versehenen Daten in den Projektplan (nur Projektleiter nach Fertigstellung der Aufgabe).

C.5 Attribute von E-Doks

E-Doks haben folgende Attribute:

- Dokument ist versionsverwaltet: Nach Modifikationen wird bei Fertigstellung einer Aufgabe eine Kopie des Dokuments gesondert abgelegt.
- Im A-Dok direkt anzeigen: Das Dokument selbst und nicht nur ein Link wird im A-Dok angezeigt. Dies soll die Arbeit bei kurzen Aufgaben vereinfachen (nur Notes RTF-Dokumente).

C.6 Algorithmus des Workflow-Agenten

Der Workflow-Agent wird immer gestartet, wenn der Zustand von A-Doks geändert wird. Da immer alle Dokumente überprüft und keine Counter verwendet werden, hat der Algorithmus eine gute Robustheit. Der Projektleiter könnte z.B. A-Doks vorzeitig versenden, ohne daß der Workflow durcheinander gerät.

```
for (doc = »Alle A-Doks in ProjektDB«) {
   Zustand[doc.UniqueId] = doc.Zustand;
}

for (doc = »Alle A-Doks in ProjektDB«) {
   if (doc.zustand == »Warten«) {
      kannArbeiten = TRUE;
      for (vorg = »alle doc.Unique Ids der Vorgänger«) {
         if (Zustand[vorg] <> »Fertig«) {
            kannArbeiten = FALSE;
            break;
         }
      }

      if (kannArbeiten) {
         if (doc.IstMeilenstein) {
            doc.Zustand = »Fertig«;
            »doc an Projektleiter schicken«
            »Agent neu triggern«
         } else {
            doc.Zustand = »In Arbeit«;
            »doc an Verantwortlichen schicken«
         }
      }
   }
}
```

C.7 Offene Punkte

- Unterstützung der Arbeit zwischen dem Verantwortlichen (ist für das Ergebnis verantwortlich und setzt das A-Dok auf »Fertig«) und dem/den Bearbeitern (tun die Arbeit). Bearbeiter und Verantwortlicher sind bei Aufgaben, die von Teams bearbeitet werden, nicht immer identisch.
- Uniquelds für die beiden Meilensteine bei Sammelaufgaben.

C.8 Schnittstelle MS-Project - Lotus Notes

Da der OLE-Zugang zu MS-Project von Lotus-Notes 4.5 (und vorher) nicht funktioniert, wird vom Projekt-Template ein Work-around mit Dateien bereitgestellt. Die ausgetauschten Textdateien bestehen aus Zeilen, die durch Kommas getrennte Werte enthalten. Dabei werden folgende Formatierungen der Werte eingehalten:

- Text ist in doppelte Hochkommas »“« einzuschließen. »“« im Text werden nicht unterstützt.
- Ein Datum ist in der Form #yyyy-mm-dd hh:mm:ss# anzugeben. Ist das Datum nicht gesetzt, wird “NV“ benutzt, der Default-Wert von MS Project für Null-Value.
- Aufwände werden immer in Minuten angegeben
- Boolsche Werte werden als #TRUE# und #FALSE# kodiert

C.8.1 Die Importdatei ».imp«

Beim Öffnen des Projektes wird nach einer Datei gesucht, die mit demselben Namen im selben Verzeichnis wie das Projekt steht, aber die Dateierweiterung ».imp« hat. Diese Textdatei wird in das Projekt eingearbeitet. Der Benutzer wird über die Änderungen informiert. Nach Abschluß der Änderungen wird er gefragt, ob er das Projekt wieder speichern will. Antwortet er mit »Ja«, wird die Projektdatei zurückgeschrieben, um in die Lotus-Datenbank intergriert zu werden und die Datei ».imp«-Datei wird gelöscht.

Die Datei hat folgende Felder:

1. Eindeutige Nummer des Vorgangs
2. Anfangsdatum des Vorgangs
3. Endedatum des Vorgangs
4. Aufwand
5. Anmerkung zur Bearbeitung des Vorgangs

Beispiel:

```
54,#1/12/1997 16:12:00#,#23/7/1998 16:12:00#,960,“rem“
```

C.8.2 Die Ausgabedatei ».out«

Beim Schließen des Projektes wird eine Datei erstellt, die mit demselben Namen im selben Verzeichnis wie das Projekt steht, aber die Dateierweiterung ».out« hat. Diese Datei soll es Notes ermöglichen, alle Änderungen im Projekt in die entsprechenden Notes-Dokumente einzuarbeiten. Wurde das Projekt seit der letzten Modifikation noch nicht gespeichert, dann wird der Benutzer dazu auf-

gefordert. Speichert er das Projekt nicht, wird eine eventuell noch existierende ».out«-Datei gelöscht und keine neue angelegt. In diesem Fall können Inkonsistenzen auftreten, wenn der Benutzer Zwischenstände gespeichert hat.

Die ».out«-Datei hat folgende Felder:

1. Eindeutige Nummer des Vorgangs
2. ID, die dem Benutzer angezeigt wird (n-ter Vorgang)
3. Name
4. Geplanter Start (durch Extras->Überwachung->Basisplan speichern)
5. Geplantes Ende
6. Geplanter Aufwand
7. Kommentar zum Vorgang
8. Aktueller Start
9. Aktuelles Ende
10. Aktueller Aufwand
11. Kommentar zur Bearbeitung des Vorgangs
12. Eingabedokumente (Kommagetrennte Liste der Dokumente)
13. Ausgabedokumente (Kommagetrennte Liste der Dokumente)
14. Summary
15. Milestone
16. Eindeutige Nummern der Vorgänger (Kommagetrennte Liste)
17. Name des Verantwortlichen (Notes Mailadresse)

Beispiel:

```
13,16,“Entwurf“,“NV“,“NV“,0,““,#1996-12-03 10:24:00#,#1996-12-12
16:12:00#,4080,““,““,#TRUE#,#FALSE#,“10“,“Meier“
```