# Management kundenorientierter Softwareentwicklungprojekte mit Objektorientierung

Guido Gryczan, Stefan Roock, Henning Wolf, Heinz Züllighoven

## *Zusammenfassung*

In diesem Artikel arbeiten wir die wesentlichen Merkmale einer zielorientierten Projektplanung für gebrauchstaugliche Software heraus. Wir stellen unsere grundsätzliche Sichtweise der Softwareentwicklungsprojekte im Rahmen einer konsequenten Kundenorientierung vor und diskutieren den Stellenwert der Objektorientierung. Dabei betonen wir auch die Bedeutung von Leitbildern und Entwurfsmetaphern für die Planbarkeit des Entwicklungsprozesses. Wir erläutern, wie kundenorientierte Softwareentwicklung, zielorientiertes Management von Entwicklungsprojekten und objektorientierte Programmiertechniken ineinandergreifen und die Planbarkeit von Projekten gegenüber klassischen Managementansätzen erhöhen. Abschließend betrachten wir den Einfluß von Rahmenwerken auf die Projektabwicklung und -organisation.

## *Vorstellung der Autoren*

**Dr. Guido Gryczan** ist Akademischer Oberrat am Arbeitsbereich Softwaretechnik (AB SWT) der Universität Hamburg und Gesellschafter der POS Halbzeug Dr.-Ing. Heinz Züllighoven GbR (www.jwam.de). Sein Arbeitsschwerpunkt ist die Unterstützung von Kooperation und Koordination mit objektorientierten Softwaresystemen. Er liest e-mail unter: gryczan@informatik.uni-hamburg.de

**Stefan Roock** und **Henning Wolf** sind wissenschaftliche Mitarbeiter am AB SWT der Universität Hamburg im EU-Drittmittelprojekt REFORM. Sie sind Architekten des Anwendungsrahmenwerks JWAM. Ihre Arbeitsschwerpunkte sind Techniken des Managements objektorientierter Projekte und Konstruktion objektorientierter Rahmenwerke. Sie lesen e-mail unter {roock, wolf}@informatik.uni-hamburg.de

**Dr.-Ing. Heinz Züllighoven** ist Professor am AB SWT der Universität Hamburg und Leiter des dortigen Softwaretechnik-Centers. Sein Arbeitsschwerpunkt sind Projekte und Publikationen zu den Themen objektorientierte Softwareentwicklung, Frameworks, Prototyping und evolutionäre Systementwicklung. Er liest e-mail unter: zuelligh@informatik.uni-hamburg.de

Wir betreuen seit 1991 industrielle Softwareentwicklungsprojekte vorrangig im Finanzdienstleistungsbereich. Ein wesentlicher Faktor für den Erfolg dieser Projekte ist der konsequente kundenorientierte Ansatz bei der Softwareentwicklung unter Verwendung objektorientierter Techniken. Ein Projekt ist für uns dann erfolgreich, wenn für den Auftraggeber gebrauchstaugliche Arbeitsmittel und -gegenstände in einem für ihn durchschaubaren Entwicklungsprozeß entstehen. Dieses Ziel können wir nur erreichen, wenn Softwareentwickler im ständigen Dialog mit den Kunden stehen. Geeignete Dokumente und Prototypen helfen bei der Beantwortung projektspezifischer Aufgabenstellungen. Das Projektmanagement muß die Herstellung dieser Dokumente und Prototypen planen und ihre fristgerechte Fertigstellung sicherstellen. Dazu planen wir unsere Projekte zielorientiert.

## *Konsequente Kundenorientierung als Unternehmensstrategie*

*Kundenorientierung* gilt heute als Schlüssel zur Konkurrenzfähigkeit von Unternehmen im globalisierten Wettbewerb. Als aktuelle Unternehmensstrategie führt sie zu speziellen Anforderungen an die IT-Unterstützung: IT-Systeme müssen aufgabenorientiert gestaltet werden. Nur so können Anwender kundenorientiert arbeiten. Das Kennzeichen aufgabenorientierter IT-Systeme ist ihre hohe Gebrauchsqualität, die sich in adäquater fachlicher *Funktionalität* und benutzungsgerechter *Handhabung* zeigt. Wir fassen Kundenorientierung hier als zweistufigen Prozeß - ein Unternehmen bietet kundenorientiert Dienstleistungen oder Produkte an; eine Entwicklerorganisation gestaltet kundenorientiert Anwendungssoftware, damit dieser Anbieter die geforderte Kundenorientierung überhaupt erbringen kann.

Zur Festlegung der *Funktionalität* analysieren wir die Arbeitspraxis der Anwender. Dabei betrachten wir sowohl die Art der Aufgaben wie auch den Prozeß ihrer Erledigung. Auf der Grundlage dieser Analyse gestalten wir das IT-System so, daß Anwender bei ihren Aufgaben möglichst gut unterstützt werden. Dazu realisieren wir im IT-System die Gegenstände der Arbeit und nicht die Abläufe. Der Anwender soll die bereitgestellten Gegenstände

möglichst flexibel zur Aufgabenerledigung kombinieren können. Die Konstruktion von Arbeitsgegenständen ist auch deshalb vorteilhaft, weil sich die Abläufe häufig ändern, während die verwendeten Gegenstände stabil bleiben.
Die benutzungsgerechte *Handhabung* versuchen wir durch die Identifikation von Arbeitsplatztypen zu erreichen. Wir haben festgestellt, daß verschiedene Arbeitsplatztypen jeweils unterschiedliche Anforderungen an das IT-System stellen: dazu gehört der Arbeitsplatz für eigenverantwortliche qualifizierte Tätigkeiten oder der Funktionsarbeitsplatz für die wiederholte, routinisierte Ausführung von Aufgaben (vgl. [ZGK+99]).
Kundenorientierung bedeutet für uns, die verschiedenen Typen von Arbeitsplätzen so zu unterstützen, daß sowohl anwender- als auch unternehmensspezifische Anforderungen berücksichtigt werden. Diese Ausrichtung birgt Zielkonflikte: Einerseits sind gerade qualifizierte Anwender bei der Lösung komplexer Problemstellungen - wie z.B. der Gewährung von Krediten im Firmenkundenbereich – auf Freiheitsgrade bei der Aufgabenerledigung angewiesen. Es gibt keinen „one best way“, um diese Aufgabenbereich zu unterstützen. Andererseits haben Unternehmen ein Interesse daran, nachvollziehbare und rationelle Arbeitsprozesse organisatorisch festzulegen. Schließlich ist ein Softwareprodukt kein Selbstzweck, sondern dient unternehmensstrategisch der effizienten Durchführung von Geschäftsprozessen.
Wir werden zeigen, mit welchen Mitteln wir diesen Zielkonflikt beherrschbar machen und welche Techniken des Projektmanagements wir dazu verwenden. Zuvor ist es aber notwendig, den Gegenstand des Entwicklungsprozesses aus unserer Sicht zu betrachten.

## *Wiederverwendung und Änderbarkeit als IT-Strategie*

Die Objektorientierung ist mit zwei großen Versprechen angetreten. Objektorientierte Systeme sollen änderungsfreundlich sein und gleichzeitig eine hohe Wiederverwendung erlauben.
Um diese Versprechen einzulösen, müssen Vorkehrungen getroffen werden. Zunächst muß die Modellierung eines objektorientierten Systems an Konzepten festgemacht werden, die möglichst stabil sind. Die Erfahrung lehrt, daß die Arbeitsgegenstände (Materialien) im Anwendungsbereich geeignete Kandidaten sind. So sind Konzepte wie Konto, Überweisungsformular und Auszahlungsbeleg im Bankenbereich seit langem weitgehend stabil, während sich die Technologien ändern, mit denen diese Gegenstände bearbeitet werden. Es ist also sinnvoll, ein IT-System entlang dieser Gegenstände zu entwerfen und die Software als neue Technologie zu begreifen, mit der die bewährten Materialien bearbeitet wird.
Dieses Prinzip fassen wir unter den Begriff der *Strukturähnlichkeit*. Wir beziehen Strukturähnlichkeit auf das Verhältnis von Software und Anwendungsbereich: Die technischen Komponenten eines Softwaresystems modellieren die relevanten Konzepte und Gegenstände des Anwendungsbereichs. Die Architektur des Anwendungssystems spiegelt die wesentlichen Bezüge zwischen den Konzepten und Gegenständen des Anwendungsbereichs wider (nach [Zül98]).
Ähnlichkeit bezieht sich auf die Handhabung des Systems und auf seine Architektur. Für die Handhabung heißt dies, daß die Anwender die vertrauten Gegenstände ihrer Arbeitswelt im System wiederfinden. Bezogen auf die System-Mikroarchitektur bedeutet es, daß die Klassen an den Gegenständen der Anwendung orientiert werden.

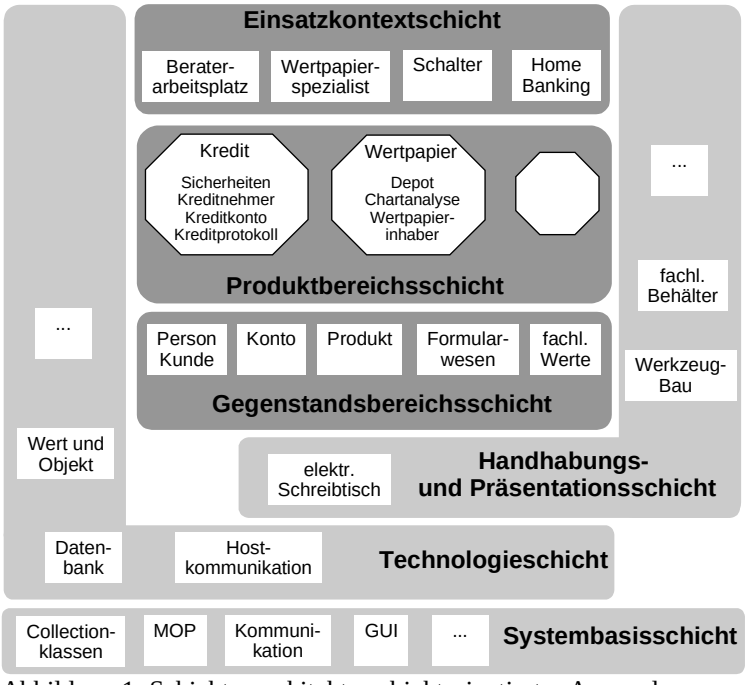

Abbildung 1: Schichtenarchitektur objektorientierter Anwendungen

Diese Form der Systemgestaltung unterstützen wir durch Leitbilder und Entwurfsmetaphern. Ein *Leitbild* ist dabei eine grundlegende Sichtweise auf ein Anwendungssystem und seine Einbettung in den Kontext. Bewährt hat sich im Büro- und Dienstleistungsbereich das Leitbild vom Arbeitsplatz für kooperative, eigenverantwortliche Aufgabenerledigung. Das jeweilig gewählte Leitbild konkretisieren wir durch *Entwurfsmetaphern*, welche die Gegenstände der Analyse und der Modellierung für Entwickler und Anwender fachlich einordnen. Zum Repertoire unseres Ansatzes gehören die Entwurfsmetaphern Werkzeug, Material, Automat, Behälter und Arbeitsumgebung (vgl. [Zül98]). Diese Metaphern setzen wir auf der Ebene der Mikroarchitekturen anhand von Entwurfsmustern um (vgl. [GHJ+98]).

Die System-Makroarchitektur realisieren wir mit Rahmenwerken anhand der organisatorischen Strukturen des Anwendungsbereichs. Als geeignet stabile Makro-Strukturen im Anwendungsbereich bieten sich Produktbereiche oder Sparten an. Banken organisieren sich z.B. traditionellerweise in Sparten wie Wertpapier oder Kredit.
Abbildung 1 zeigt, welche Art von Architektur für Anwendungssysteme aus dieser Sichtweise folgt: In der Produktbereichsschicht werden die relevanten Anwendungskomponenten realisiert. Gemeinsamkeiten über verschiedene Produktbereiche werden als Kernabstraktionen in der Gegenstandsbereichsschicht realisiert. Für einzelne Arbeitsplatztypen sinnvolle Kombinationen von Werkzeugen und Materialien werden in der Einsatzkontextschicht konfiguriert. In der Einsatzkontextschicht können so neue Typen von Arbeitsplätzen sinnvoll zusammengestellt werden. Insbesondere wird es dadurch möglich, neue Organisationsformen - die durch neue Technologien entstehen - effizient zu unterstützen.
Die anwendungsfachlich motivierten Schichten müssen technisch umgesetzt werden: Die Handhabungs- und Präsentationsschicht kapselt die Art und Weise wir fachliche Konzepte vergegenständlicht und bearbeitet werden. Die Technologieschicht realisiert die Konzepte, die notwendig sind, um Anwendungssysteme auf einer bestimmten, aber veränderlichen technischen Grundlage aufzubauen (vgl. [Bäu98], [BGK+97], [Zül98]).
Das Prinzip der Strukturähnlichkeit führt zu sehr stabilen Systemen, in denen sich Änderungen nur lokal auswirken. Dieses Prinzip fördert auch die Wiederverwendung von Konzepten im Softwaresystem. In den seltensten Fällen erfüllen die fachlichen Gegenstände der Anwendungswelt nur einen speziellen Zweck. Oft werden dieselben Gegenstände für sehr viele Aufgaben in vielfältiger Weise verwendet. Wir können also die im Anwendungsbereich vorhandene Wiederverwendung in das Softwaresystem übertragen.
So schaffen wir den Gegenstand, auf dessen Basis eine verläßliche Planung von Projekten ermöglicht wird. Die Trennung von fachlichen, technischen und handhabungsspezifischen Fragestellungen in Form von Rahmenwerken in einer Schichtenarchitektur ist eine wesentliche Voraussetzung, um eine anwendungsorientierte Projektplanung zu ermöglichen.

## *Unternehmensstrategie und IT-Strategie zur Deckung bringen*

Kundenorientierung als Unternehmensstrategie führt in unserem Ansatz also zum Prinzip der Strukturähnlichkeit bei der Gestaltung des IT-Systems. Dies hat Auswirkungen auf den Entwicklungsprozeß. Die Analyse der Arbeitsaufgaben und die Identifikation von relevanten Gegenständen und Konzepten erfordert ein hohes Anwendungswissen. Die Experten sind hier naturgemäß eher im Anwendungsbereich, sprich den Fachabteilungen zu suchen, als im Entwicklerteam. Daher ist es sinnvoll, Anwender aktiv in den Entwicklungsprozeß einzubeziehen. Im Prozeß sollen sich in kurzen Entwicklungszyklen Analyse, Entwurf, und Bewertung abwechseln. Damit die beteiligten Anwender hierbei eine aktive Rolle spielen können, müssen geeignete Entwicklungsdokumente bereitstehen. Dies führt zu anwendungsorientierten Dokumenten und Prototypen, die von zentraler Bedeutung für einen planbaren Entwicklungsprozeß sind.

## *Dokumente für den objektorientierten Entwicklungsprozeß*

Bei der Modellierung des Anwendungsbereichs stehen die Arbeitsmittel und –gegenstände im Vordergrund. Sie sollen flexibel die unterschiedlichen Arbeitsprozesse des Anwendungsbereichs unterstützen. Diesen Grundgedanken wenden wir auch bei den Dokumenten des objektorientierten Entwicklungsprozesses an: Diese Entwicklungsdokumente sollen flexibel von Entwicklern, Anwendern und dem Management zur Beantwortung spezifischer Fragestellungen eingesetzt werden können. Zentrale Fragen sind:
**Wie werden am Arbeitsplatz momentan Aufgaben erledigt?**
Hier konzentrieren wir uns darauf, mit Anwendern gemeinsam ihre spezifische Art der Aufgabenerledigung zu modellieren. Diskussionsgegenstand zwischen Anwendern und Entwicklern sind *Szenarien*, ein Dokumenttyp, in dem Aufgabenerledigung in der Sprache der Anwender auf unterschiedlichen Detaillierungsebenen beschrieben wird (vgl. [Zül98]).

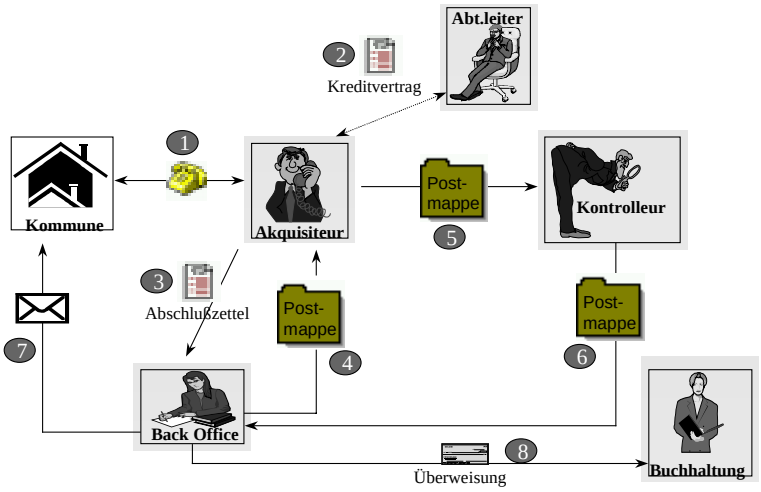

Abbildung 2: Kooperationsbild zur Vergabe von Kommunalkrediten

**Wer kooperiert mit wem, um eine Aufgabe zu erledigen?**
Gerade komplexere Aufgaben werden in der Regel nicht von einer Person allein erledigt. Deshalb muß die Art der Kooperation bei der Aufgabenerledigung herausgearbeitet werden: Welche Dokumente werden zwischen Arbeitsplätzen

ausgetauscht? Wer kann in einer Kooperation welche Aufgaben übernehmen? Welche Abhängigkeiten bestehen zwischen den Teilaufgaben, die an einem einzelnen Arbeitsplatz erledigt werden?
Wir setzen *Kooperationsbilder* zur Vergegenständlichung der Kooperationssituation ein. Abbildung 2 zeigt das Kooperationsbild zum Abschluß eines Kredits. Eine typische, aber nicht zwingende Reihenfolge von Bearbeitungsstationen wird angegeben. So wird vom Akquisiteur in Schritt 2 ein Kreditvertrag erstellt und mit dem Abteilungsleiter abgestimmt.
Kooperationsbilder können sowohl von Anwendern als auch vom Anwendungsmanagement und Entwicklern im Entwicklungsprozeß diskutiert werden: Anwendern werden die Zusammenhänge deutlich, in denen ihre persönlichen Aufgaben stehen. Zusammen mit dem Management können dabei auch die zentralen Fragen zur Reorganisation der Prozeßgestaltung behandelt werden. Unsere Erfahrung zeigt, daß die Diskussion von Kooperationsbildern gleichzeitig zu Vorstellungen über die zukünftige Arbeitsgestaltung führt.
Mit Szenarien und Kooperationsbildern stehen uns mächtige Hilfsmittel für eine aufgabenorientierte Anforderungsermittlung (vgl. [Flo87]) zur Verfügung. Diese Dokumenttypen erlauben uns, ein mit Anwendern und Management abgestimmtes Verständnis über den Ist-Zustand und die organisatorische Weiterentwicklung eines Unternehmens herzustellen.

**Wie kann eine Vision über die zukünftige Aufgabenerledigung aussehen?**

Es wäre zu kurz gegriffen, wenn aus den vorhandenen Dokumenten über die momentane Arbeitsorganisation und Aufgabenerledigung am Arbeitsplatz gleich ein ablauffähiges Modell des zukünftigen Systems konstruiert würde. Wir können zwar davon ausgehen, daß am Arbeitsplatz verwendete Materialien konzeptionell stabil bleiben, aber unsere Aufgabe als Softwareentwickler besteht darin, für eine Organisation neue Technologien so zu erschließen, daß effizientere Arbeitsabläufe für die Kernaufgaben möglich werden.
Dies bietet für Softwareentwickler einen großen Gestaltungsspielraum, der aber zu Fehlentwicklungen führen kann, wenn die technische Diskussion im Vordergrund steht oder wenn Strukturähnlichkeit als direkte Übernahme vorhandener Arbeitsprozesse betrachtet wird.
Wenn wir als Softwareentwickler eine Vision des zukünftigen Systems entwickeln, werden wir sicherlich moderne Technologien einsetzen. Doch primär bilden die in Szenarien und Kooperationsbildern dokumentierten anwendungsfachlichen Aufgaben den Rahmen für den zukünftigen Umgang mit einem Softwaresystem. Unsere Vorstellungen von dem zukünftigen System dokumentieren wir in sog. *Systemvisionen*, einem Dokumenttyp, der vergleichbar zu Use Cases als Text das Gegenstück zu den Szenarien bildet.
Systemvisionen sind im Entwicklungsprozeß nur ein Zwischenschritt. Sie sind für Entwickler ein notwendiger Diskussionsgegenstand, um die Aufgabenerledigung am Arbeitsplatz „durchzuspielen“. Systemvisionen sind aber nicht hinreichend, da sie nicht ausreichen für die Rückkopplung mit zukünftigen Anwendern.

**Wie können wir unsere Modelle überprüfen?**

Das entscheidende methodische Mittel zur Rückkopplung von Entwurfsentscheidungen sind *Prototypen*. Verschiedene Arten von Prototypen werden konstruiert, um spezifische Fragestellungen zu diskutieren:

- Demonstrationsprototypen mit geringer fachlicher Funktionalität zeigen prinzipielle Designentscheidungen. Wie soll die Aufgabenerledigung grundsätzlich unterstützt werden? Soll ein in Grenzen frei gestaltbarer Arbeitsplatz mit Werkzeugen und Materialien entwickelt werden oder ein Funktionsarbeitsplatz mit geringem Gestaltungsspielraum, so daß die Einarbeitungszeiten für Anwender minimal ist?
- Funktionale Prototypen zeigen die zukünftige Aufgabenerledigung in der Tiefe. Fragestellungen sind hier: Wie kann eine Aufgabe unter Verwendung der neuen Arbeitsmittel und –gegenstände erledigt werden? Welche Variationen sind möglich? Welche neuen Arbeitsschritte sind notwendig?

Die verschiedenen Prototypen sind in den jeweiligen Projektstadien entscheidende Mittel zur Vergegenständlichung des Projektstandes und zur anwendungsfachlichen Bewertung der Zwischenergebnisse.
Fassen wir zusammen: Die hier vorgestellten fachlich motivierten Dokumenttypen bilden die Basis für das Management objektorientierter Projekte. Hier betreiben wir Objektorientierung auf Dokumentationsebene: Die Entwicklungsdokumente sind die Objekte des Entwicklungsprozesses. Wir als Softwareentwickler sind qualifizierte Anwender des Entwicklungsprozesses.
Wie dieser Prozeß gesteuert werden kann, und wie wir den objektorientierten Gedanken dabei weiter verfolgen, ist Thema des nächsten Abschnittes. Mit den hier vorgestellten Dokumenten ist zunächst nur eine Basis für einen anwendungsorientierten Entwicklungsprozeß geschaffen.

## *Management des objektorientierten Entwicklungsprozesses*

Grundlegend für das Management des Entwicklungsprozesses ist eine *Zielvorstellung* für das Projekt. Die Zielvorstellung als Vision des zukünftigen Systems ist das treibende Moment des Entwicklungsvorhabens. Sie wird sich im Laufe eines Entwicklungsvorhabens verändern, muß aber zu Beginn eines Projektes skizziert werden, da die beteiligten Gruppen sonst keine Orientierung haben.
Zentrale Fragen für die Erarbeitung einer Zielvorstellung sind:

- Was ist Sinn und Zweck des Softwareprojekts, d.h. welche Veränderung im Anwendungsbereich soll erreicht werden?

- Welche Mittel (z.B. das zu entwickelnde Softwaresystem) in welchem Umfang erfüllen das Ziel? Welche Alternativen sind denkbar?
- Wer stellt die Zielerreichung fest? Ist es eine oder sind es mehrere Personen- oder Interessensgruppen?
- Welche Zeitvorstellungen gibt es für die Zielerreichung? Wie flexibel sind diese Zeitvorstellungen?

Anhand einer Zielvorstellung planen wir ein Entwicklungsprojekt unter Verwendung von *Projektetappen* und *Referenzlinien* zeitlich von „hinten nach vorne“, d.h. vom anvisierten Zeitpunkt des Projektendes hin zum Zeitpunkt des Projektanfangs. Grundgedanke dabei ist, herauszuarbeiten, welche Schritte jeweils notwendig sind, um ein (Teil-)Ziel zu erreichen. Anders ausgedrückt: Welche Projektetappen müssen wann durchlaufen werden, um das Ziel des Projektes zu erreichen?

## Projektetappen

Eine Projektetappe definiert einen bestimmten Zustand des Projektes zu einem festgelegten Zeitpunkt. Das Erreichen einer Projektetappe ist immer mit der Konstruktion eines Prototypen verbunden. So werden Projektetappen zu Synchronisationspunkten für Entwickler und Anwender. Durch den Zwang, ausführbare Systeme in die Projektplanung zu integrieren, kann in kurzen Abschnitten der Projektstand vom Kunden überprüft werden. Anders als bei traditionellen Entwicklungsmodellen auf der Basis von Meilensteinen sind Entwickler so gezwungen, jeweils ein Teilziel im Sinne des Gesamtprojektziels anzusteuern. Auf der Grundlage der Bewertung eines Teilziels anhand von Prototypen wird ggf. das Gesamtziel in Abstimmung mit dem Kunden neu ausgerichtet.

| **Teilziel:** | **Umsetzung:** | **Wann:** |
|---|---|---|
| Vorstand akzeptiert Reorganisationskonzept | Demo: Handhabungsprototyp plus Szenarios, Glossar, Visionen im Bereich Accountmanager | 31.3. |
| Zentrale Entwicklung nimmt Architektur ab | Review: Laborprototyp mit Anschluß an Hostrechner und Rel.DB | 16.5. |
| Vorstand akzeptiert einen reorganisierten Kernprozeß | Demo: Pilotsystem Accountmanger-arbeitsplatz in einer Bankfiliale im Testeinsatz | 30.9. |
| Vorstand von Machbarkeit des Konzepts überzeugt | Bericht: Accountmanagerarbeitsplatz in drei Banken im Echtbetrieb | 1.12. |
| Revision nimmt System ab | Review: Software gemäßt Projekt-handbuch dokumentiert und getestet | 10.12. |
| Erste Ausbaustufe vom Vorstand akzeptiert | Workshop mit Anwendern | 15.2. |

Abbildung 3: Projektplanung mit Teilzielen

Abbildung 3 zeigt eine leicht modifizierte Etappenplanung aus einem unserer Projekte. Dabei ist zu beachten, daß unterschiedliche Arten von Prototypen in den jeweiligen Projektetappen eingesetzt werden.
Ein weiterer Unterschied zu traditionellen Managementtechniken fällt ins Auge: Für die aktuelle Projektplanung werden alle bis dahin erstellten Dokumente überprüft. So ist es üblich, daß bei der Rückkopplung über einen Handhabungsprototyp für Anwender und Entwickler klar wird, daß die Erhebung des Ist-Zustands mit Szenarien nicht ausreichend war und detailliert werden muß. Auch Visionen werden wiederholt auf ihre Tragfähigkeit überprüft.

## Referenzlinien

Typischerweise umfassen einzelne Projektetappen einen Zeitraum von ca. drei Monaten. Innerhalb dieses Zeitraums wird die Feinplanung von Aufgaben über Referenzlinien durchgeführt. Auf dieser Ebene wird zeitlich von „vorne nach hinten“ geplant. Eine Referenzlinie beschreibt, welche Aufgaben von wem erledigt werden sollen. Sie definiert einen qualitativen oder quantitativen Dokumentenzustand oder überprüfbare Aufgaben. Da Referenzlinien sehr kleine Arbeitspakete beschreiben, können sie zeitlich sehr genau (bis auf Tagesbasis) terminiert werden. Abbildung 4 zeigt einen Ausschnitt aus einer Referenzlinienplanung.

| **Wer** | **macht Was mit Wem/Was** | **Wozu** | **Wie geprüft** |
|---|---|---|---|
| PB | Terminabsprachen mit Pilotbank | Vorbereitung Interviews | email an Team |
| RS | Ausarbeiten Interviewleitfaden | Vorbereitung Interviews | Vorstellung auf Meeting |
| PB, RS | Interviews mit Accountmanagern | Basis für Szenarios | Protokolle in ProjektDB |
| DM | Werkzeug-Framework erstellen | Vorbereitung WZ-Implementation | Laborprototyp eines Werkzeugs lauffähig |
| AK, WS | Entwurf u. Implementation eines Werkzeugs für Abschlußzettel | Test für Framework, Vorbereitung AnwenderWS | Laborprototyp eines Werkzeugs lauffähig |

Abbildung 4: Referenzlinien zur Planung einer Projektetappe

## Strukturierung der Projektplanung durch Systemdekomposition

Neben Projektetappen und Referenzlinien verwenden wir das Konzept des *Kernsystems*, um diskutierbar zu machen, welche Anteile des Zielsystems wir in welchen Projekten und wann bearbeiten (siehe [Zül98]).
Das Konzept des Kernsystems mit Ausbaustufen sowie der Spezialsysteme erlaubt die Dekomposition des zu entwickelnden Systems. Das Kernsystem wird dabei zeitlich zuerst realisiert. Es stellt die Basisdienste zur

Verfügung, die von fast allen anderen Teilen der Anwendung benötigt werden. Außerdem erfüllt das Kernsystem dringende Anforderungen. Da das Kernsystem selbst häufig immer noch zu umfangreich ist, um es „aus einem Guß“ zu konstruieren, unterteilen wir es weiter in *Ausbaustufen*. Die Ausbaustufen werden anhand logischer Abhängigkeiten zwischen den Systemkomponenten definiert. Die Komponenten, von denen alle anderen Komponenten abhängen, werden zuerst realisiert.
Dies ist im Bankenbereich mit seiner traditionellen Sparteneinteilung (Aktiv-, Passiv-Geschäft, Wertpapierhandel, Schalter usw.) offensichtlich. Daß diese Art der Systempartitionierung auch in ganz anders strukturierten Domänen greift, zeigt Abbildung 5. Hier ist ein in fünf Ausbaustufen geplantes Kernsystem einer Prozeßautomatisierungssoftware für Stahlwalzwerke dargestellt. Die Softwarekomponenten der Ausbaustufe 1 werden für die in Ausbaustufe 2 zu erstellenden Komponenten benötigt, z.B. setzt der „Primary data handler“ auf den „Telegram Handler“ auf. Die in Ausbaustufe 4 zu entwikkelnde Komponente für Modellrechnungen (Model computation) kann nicht ohne die Komponenten der Ausbaustufe 3 Messwertverarbeitung (Measured Value Processing), Modellabgleich (Model control) und Materialverfolgung (Material Tracking) entwickelt werden.

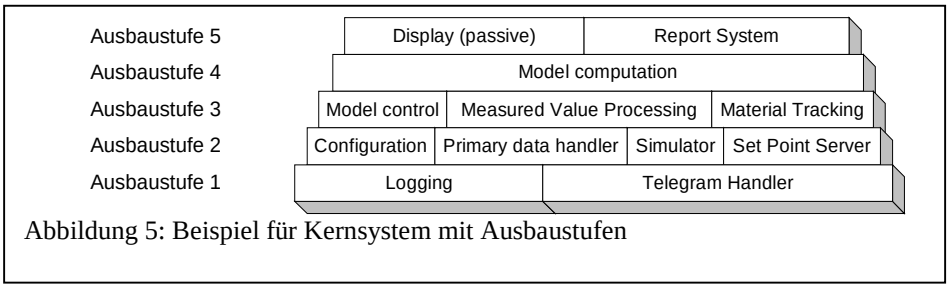

Abbildung 5: Beispiel für Kernsystem mit Ausbaustufen

Nach Möglichkeit wählen wir die Ausbaustufen so, daß sie den Anwendern (oder zumindest einem Teil der zukünftigen Anwender) bereits vor Fertigstellung des Gesamtsystems ausgeliefert werden können, mindestens als Prototypen.
Das Kernsystem kann durch Spezialsysteme ergänzt werden. Die Spezialsysteme sollten sich nur auf das Kernsystem beziehen, aber ansonsten unabhängig voneinander sein. So können sie auch in Teilprojekten und zeitlich parallel entwickelt werden.

## Diskussion

Neben den oben beschriebenen Dokumenten und Prototypen setzen wir also weitere Dokumenttypen für das Management des Entwicklungsprozesses ein: Dokumentation von Projektetappen und Referenzlinien bezogen auf das Kernsystem und seine Ausbaustufen. Auch hier findet sich der zentrale Gedanke wieder: Konzentriere dich auf die Gegenstände (in diesem Fall: Dokumente) des Prozesses und nicht auf die Abläufe. Die Konzentration auf Dokumente anstelle von Abläufen macht die Prozesse anpaßbar. Der Prozeß selbst wird durch die ihn repräsentierenden Gegenstände für das Projektmanagement zum Diskussionsgegenstand. Diese Vergegenständlichung ist notwendig, um mit der Dynamik der Softwareentwicklung umzugehen. Dazu gehören die Änderung von Anforderungen, oder ein erst im Entwicklungsprozeß wachsendes Verständnis für den Anwendungsbereich.
Unser Ansatz führt also nicht zu einem unplanbaren oder beliebigen Entwicklungsprozeß, sondern zu einem Entwicklungsprozeß, der sich an die Gegebenheiten der Entwicklungssituation anpassen läßt.
Die hier beschriebenen Konzepte werden in Entwicklungsorganisationen eingesetzt, in denen bis zu 60 Entwickler parallel in Projekten tätig sind. Einzelne Projekte umfassen dabei in der Regel nicht mehr als fünf bis sieben Entwickler. Diese Größenordnung für Projekte sollte nach unseren Erfahrungen nur selten überschritten werden, da der kommunikative Aufwand für die Synchronisation der Entwickler exponentiell mit der Anzahl (der Entwickler) wächst.
Vor diesem Hintergrund kommt dem Projektleiter eine neue Rolle zu. Er wird zum "Facilitator", also der Person, die für die Funktionstüchtigkeit des Teams sorgt. Er beschafft Ressourcen, koordiniert Termine, ist Ansprechpartner für die Teammitglieder bei jeglicher Art von Problemen. Zu seinen Aufgaben gehört auch die Außenvertretung des Projekts. In seltenen Fällen wird der Projektleiter Zeit und Qualifikation besitzen, um selbst wesentliche Designentscheidungen zu treffen.
Für eine effektive Kommunikation und Koordination innerhalb einer Entwicklungsorganisation ist es notwendig, daß gemeinsam genutzte Dienstleistungen in Form von Rahmenwerken als Teil der Entwicklungskultur begriffen werden. Über diesen Zusammenhang handelt der letzte Abschnitt unserer Darstellung.

## *Der Einfluß objektorientierter Rahmenwerke auf das Projektmanagement*

Wiederverwendung muß systematisch geplant und umgesetzt werden. Sie wird nicht automatisch dadurch erreicht, daß objektorientierte Technologien eingesetzt werden. Heute werden in diesem Zusammenhang vor allem Rahmenwerke (engl. Framework, siehe z.B. [JWAM]) sowie Komponenten (siehe [Szy97]) diskutiert. Auf Rahmenwerke sind wir bereits im ersten Teil dieses Beitrags eingegangen. Damit diese Technologien allerdings zum Erfolg führen, sind nicht zuletzt die organisatorischen Randbedingungen zu berücksichtigen. Darauf haben

wir bereits in [RWZ98] hingewiesen und den Begriff „Frameworking“ geprägt: Der Prozeßaspekt läßt sich bei Rahmenwerken genauso wenig vom Produktaspekt isolieren, wie sich Herstellung und Einsatz wiederverwendbarer Komponenten voneinander trennen lassen. Betrachten wir diese Zusammenhänge.

Komponenten werden in Projekten stets in einem konkreten Einsatzkontext entwickelt. Um diese Komponente wiederverwenden zu können, ist in jedem Fall eine Generalisierungsphase notwendig (siehe [Mey95]). Für diese Phase fehlt in den Projekten in der Regel das Geld und die Zeit. Außerdem sind dazu Entwickler mit hohem fachlichen und technischen Wissen notwendig.

Es müssen also Strukturen geschaffen werden, welche die Generalisierung der Komponenten aus den Anwendungsprojekten sowie die Pflege und Verwaltung der wiederverwendbaren Komponenten sicherstellen. Wir schlagen Rahmenwerke als technische Infrastruktur und eine Architekturgruppe als organisatorische Maßnahme vor. Die wiederverwendbaren Komponenten werden in Rahmenwerke eingepaßt. Dadurch erhalten sie einen Rahmen, der ihren Einsatzkontext verdeutlicht. Gleichzeitig erhalten die Komponenten durch das Rahmenwerk einen definierten Zusammenhang mit anderen Komponenten. Die *Architekturgruppe* ist für die Weiterentwicklung und Pflege der Rahmenwerke sowie für die Unterstützung der Anwendungsprojekte zuständig. Eine kleine Gruppe von hoch qualifizierten Entwicklern mit einem großen fachlichen wie technischen Erfahrungsschatz soll die Architekturgruppe bilden. Sie stellt im Rahmen der Weiterentwicklung und Pflege der Rahmenwerke den Wissenstransfer sicher. Die Architekturgruppe bringt die Rahmenwerke in die Anwendungsprojekte ein, schult Entwickler für den Einsatz der Rahmenwerke und betreut die Anwendungsentwickler. Die Entwickler der Architekturgruppe können sicherstellen, daß die Rahmenwerke in der intendierten Weise eingesetzt werden. Sie identifizieren frühzeitig potentiell wiederverwendbare Komponenten in den Anwendungsprojekten und sorgen dafür, daß diese in der notwendigen Qualität erstellt werden.

Wiederverwendbare Komponenten und Rahmenwerke müssen den höchsten Qualitätsstandards genügen. Fehler und Inkonsistenzen würden sich ansonsten multiplizieren. Wir sehen verschiedene Mechanismen, um diese Qualität sicherzustellen:

1. In der Architekturgruppe müssen die „Besten der Besten“ arbeiten. Sie müssen den finanziellen und zeitlichen Freiraum haben, um die geforderte Qualität zu erbringen. Das bedeutet insbesondere, daß die Mitglieder der Architekturgruppe nicht regelmäßig als „Feuerwehr“ in Anwendungsprojekten eingesetzt werden dürfen.
2. Es muß ein enger Kontakt zwischen Rahmenwerks- und Anwendungsentwicklern bestehen. Die Rahmenwerksentwickler dürfen sich nicht als „etwas Besseres“ verstehen. Zudem muß darauf geachtet werden, daß die Architekturgruppe nicht den Kontakt zum Anwendungsbereich verliert. Dies kann durch die Mitarbeit in Anwendungsprojekten erreicht werden. Vor diesem Hintergrund kann dann auch entschieden werden, welche anwendungsfachlichen Komponenten in welcher Form in das Rahmenwerk Eingang finden sollen. Damit ist auch klar, daß Rahmenwerke nicht ohne Anwendung entwickelt werden können. Wenn die Architekturgruppe sehr viel Erfahrung aus anderen Rahmenwerksprojekten hat, kann sie Teile der technischen Rahmenwerksbasis auch ohne konkrete Anwendung entwickeln. Allerdings sollten sich die Rahmenwerksentwickler an dieser Stelle in Bescheidenheit üben. Denn oft stellt sich Technologieentwicklung „auf Vorrat“ als kostspieliger Irrweg heraus.
3. Alle Kernkomponenten des Rahmenwerkes müssen ausgiebig getestet werden. Hier bietet es sich an, für jede Klasse eine Testklasse zu schreiben und diese zusammen mit dem Rahmenwerk auszuliefern. So können Anwendungsentwickler leicht sehen, welche Fälle getestet wurden und welche nicht. Testklassen können auch verwendet werden, um herauszufinden, ob eine Fehlfunktion in der Anwendung auf einen Fehler im Rahmenwerk oder im Anwendungsprogramm zurückzuführen ist. Weiterhin können Umbaumaßnahmen am Rahmenwerk selbst (das sogenannte „Refactoring“) nur dann sinnvoll durchgeführt werden, wenn mit Hilfe der Testklassen jeweils schnell festgestellt werden kann, ob neue Fehler oder unbeabsichtigte Seiteneffekte entstanden sind. Wir haben im Rahmen der Entwicklung eines Java-Rahmenwerks zur Unterstützung unseres Ansatzes (siehe [JWAM]) die Erfahrung gemacht, daß sich das derzeit heiß diskutierte Pair-Programming (siehe z.B. [XP]) positiv auf die Qualität auswirkt. Beim Pair-Programming sitzen stets zwei Entwickler vor dem Bildschirm und diskutieren während der Programmierung kontinuierlich die getroffenen Entwurfsentscheidungen. So werden Flüchtigkeitsfehler, Inkonsistenzen, redundanter Code etc. in den meisten Fällen schon während des Programmierens festgestellt. Unsere Erfahrung deutet hier darauf hin, daß sich für Kernkomponenten Pair-Programming in einer deutlich höheren Qualität bei gleichbleibender oder sogar gestiegener Entwicklungsgeschwindigkeit niederschlägt.

Die Entwicklung und Pflege von Rahmenwerken stellt eine zusätzliche Herausforderung für die Projektorganisation dar. Sowohl das Rahmenwerk als auch die Existenz der Architekturgruppe haben einen weitreichenden Einfluß auf die Anwendungsentwicklung. Einige der Auswirkungen sind positiver Natur und beabsichtigt: kürzere Entwicklungszeiten, höhere Qualität etc. Dazu kommen weitere Effekte, die zumindest bedacht werden müssen.

Rahmenwerke geben nicht nur einen Rahmen für die Anwendungssysteme sondern auch für die Entwicklungsprozesse selbst vor.

Bewegt sich ein Anwendungssystem mit der geforderten Funktionalität in diesem Rahmen, läßt es sich sehr schnell realisieren. Paßt ein Anwendungssystem nicht genau ins Rahmenwerk, so kann dies sehr negative Folgen auf die Entwicklungszeit haben. Oft muß das Rahmenwerk an vielen Stellen umgangen werden, damit es für die Anwendung eingesetzt werden kann. Alternativ wird das Rahmenwerk um die fehlenden Konzepte erweitert.
Der erste Ansatz führt neben höheren Entwicklungskosten für die Anwendungssoftware zu einem tendenziellen Strukturverlust des Systems. Jede Änderung des Rahmenwerks führt zu neuerlichen Umgehungsversuchen. Der zweite Ansatz ist auf Dauer mit weniger Aufwand verbunden. Allerdings lassen sich die neuen Anforderungen an das Rahmenwerk nicht sofort umsetzen. Das Anwendungsprojekt muß mindestens auf die nächste, häufig sogar auf die übernächste Version des Rahmenwerkes warten, ehe die benötigten Änderungen integriert sind.
Sollen Zeit und Kosten für die Anwendungsentwicklung abgeschätzt werden, muß der Aufwand für die Anpassung des Rahmenwerks einkalkuliert werden. Die oben beschriebene Schichtenarchitektur gibt wertvolle Hilfestellung bei dieser Abschätzung: Fachliche Änderungen dauern länger, je weiter unten in der Schichtenarchitektur sie angesiedelt sind. Änderungen in der Gegenstandsbereichsschicht sind am teuersten, während Änderungen an der Einsatzkontextschicht meist kostengünstig und schnell realisiert werden können.
Da eine Reihe von Anwendungsprojekten auf demselben Rahmenwerk basiert, bekommt das Rahmenwerk Produktcharakter. Dies gilt auch, wenn das Rahmenwerk nur im eigenen Unternehmen eingesetzt wird. Dadurch entsteht das Problem der Versionierung. Auf der einen Seite spürt die Architekturgruppe einen gewissen Stabilitätsdruck, da die Anwendungsentwickler nicht ständig damit beschäftigt sein können, die Anwendung auf die aktuelle Rahmenwerksversion umzustellen. Andererseits baut sich bei den Anwendungsentwicklern ein Migrationsdruck auf, da die Architekturgruppe nicht für beliebig viele Rahmenwerksversionen Support und Pflege leisten kann. Einen Kompromiß bringt das Vorhalten von drei Rahmenwerksversionen (siehe Abbildung 6).

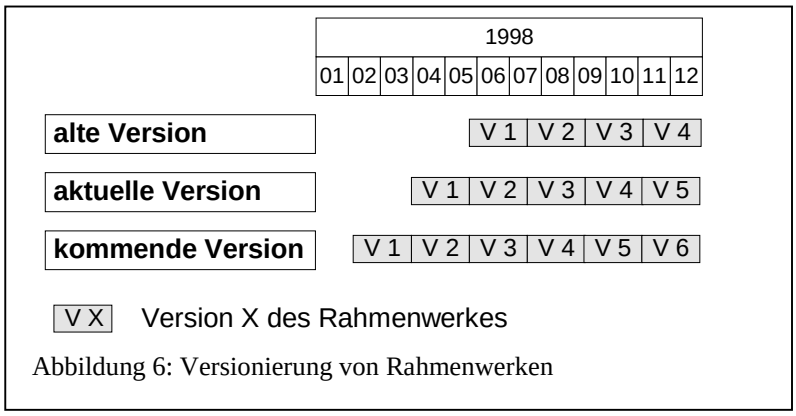

Abbildung 6: Versionierung von Rahmenwerken

Zu jedem Zeitpunkt existiert eine aktuelle Version des Rahmenwerkes, eine alte und die kommende Version. Die aktuelle Version des Rahmenwerkes ist die zuletzt freigegebene Version. Sie hat die alte Version ersetzt. Die aktuelle und die alte Version werden gleichzeitig von der Architekturgruppe unterstützt. Neue Anwendungsprojekte stützen sich auf die aktuelle Version ab. Projekte, deren Ergebnisse bereits „in Produktion“ sind, können noch eine Weile auf der alten Version des Rahmenwerkes arbeiten, da diese nach wie vor unterstützt wird. Sie haben so ausreichend Zeit, sich auf die aktuelle Version umzustellen. Die kommende Version bietet erste Einblicke in die nächste Version des Rahmenwerkes. So können „junge“ Projekte schon gegen die kommenden Schnittstellen programmieren oder zumindest abschätzen, wie groß der Anpassungsaufwand für die nächste Version wird.
Das Versionierungsproblem zeigt, daß die Anwendungsprojekte durch den Rahmenwerkseinsatz an das Rahmenwerksprojekt gekoppelt werden. Dadurch werden gleichzeitig die Anwendungsentwickler an die Architekturgruppe „gekoppelt“. Zwischen ihnen muß eine reibungslose Kommunikation sichergestellt werden. Da Kommunikation zwischen Entwicklern „teuer“ ist (siehe [Bro95]), hat es sich bewährt, kleine Projektgruppen (5-10 Entwickler) zu bilden. Durch die Verkopplung der Entwicklungsteams müssen jedoch deutlich mehr Entwickler miteinander kommunizieren. Die Architekturgruppe ist ein erster Schritt, diesem Problem zu begegnen. Systematische Kommunikation ist „nur“ zwischen der Architekturgruppe und den Anwendungsentwicklern notwendig, aber nicht zwischen den einzelnen Anwendungsprojekten. Dennoch müssen hier Mechanismen gefunden werden, welche einerseits ausreichenden Informationsaustausch sicherstellen und andererseits die Architekturgruppe nicht in Diskussionen mit Anwendungsentwicklern „aufreibt“.

## *Zusammenfassung*

Wir haben dargestellt, welche Anforderungen an die Softwareentwicklung sich aus der Kundenorientierung ergeben. Dabei spielt die Gebrauchsqualität des Anwendungssystem eine zentrale Rolle. Wir haben ausgeführt, daß die Objektorientierung prinzipiell geeignet ist, die geforderte Gebrauchsqualität zu erreichen. Es müssen jedoch geeignete organisatorische Maßnahmen geschaffen werden. Dies führt zu einem objektorientierten Entwicklungsprozeß, der nicht über Meilensteine, sondern spezielle Objekte – den Dokumenttypen – gesteuert wird. Aus dieser Sicht stellt sich das Projektmanagement in einem neuen Licht dar. Planbarkeit und Machbarkeit orientieren sich dabei vorrangig am fachlichen Ziel und an der anwendungsfachlichen Gestaltung des IT-Systems.

Um zu einer hohen Wiederverwendung und den damit verbundenen Zielen wie Verkürzung der Entwicklungszeiten und Erhöhung der Softwarequalität zu gelangen, sind weitere organisatorische Maßnahmen notwendig. Wir meinen, daß Objektorientierung in Kombination mit Rahmenwerken *und* Komponenten geeignet ist, die Wiederverwendung zu erhöhen. Die richtige Organisation einer Architekturgruppe ist notwendige Voraussetzung, um dies zu erreichen.

## *Literatur*


[Bäu98] Dirk Bäumer. Softwarearchitekturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme. Dissertationsschrift am Arbeitsbereich Softwaretechnik. Fachbereich Informatik. Universität Hamburg. 1998.

[BGK+97] Dirk Bäumer, Guido Gryczan, Rolf Knoll, Carola Lilienthal, Dirk Riehle, Heinz Züllighoven. Framework Development for Large Systems. Communications of the ACM, October 97, Vol. 40, No. 10, 1997.

[Bro95] Frederick Phillips Brooks: The Mythical Man-Month : Essays on Software Engineering. Reading, Mass. [u.a.] : Addison-Wesley, 1995

[Flo87] Christiane Floyd. Outline of a Paradigm Change in Software Engineering. In: Computers and Democracy – A Scandinavian Challange. Hrsg.: G. Bjerknes, P. Ehn, M. Kyng. Avebury 1987. S. 191-210.

[GHJ+98] Gamma, Helm, Johnson, Vlissides: Entwurfsmuster : Elemente wiederverwendbarer objektorientierter Software. 2. Auflage. Bonn. Addison-Wesley. 1998.

[JWAM] The JWAM framework. http://www.jwam.de

[Mey95] Betrand Meyer. Object Success. Prentice Hall. London. 1995

[RWZ98] Stefan Roock, Henning Wolf, Heinz Züllighoven. Frameworking. In: N.J. Buch et. al. (Hrsg.): Proceedings of the 21st Information Systems Research seminar in Scandinavia. Dänemark. 8.-11. August 1998. S. 743-758.

[Szy97] C. Szyperski. Component Software. Addison-Wesley. New York. 1997.

[XP] Extreme Programming. http://www.kinetica.com/ootips/xp.html, http://www.armaties.com/extreme.htm

[ZGK+99] H. Züllighoven, G. Gryczan, A. Krabbel, I. Wetzel: Application-Oriented Software Development for Supporting Cooperative Work. Angenommes Papier auf der HCI '99.

[Zül98] Heinz Züllighoven. Das objektorientierte Konstruktionshandbuch nach dem Werkzeug- und Materialansatz. dpunkt Verlag. 1998.