Ansätze einer anwendungsorientierten Softwareentwicklung

Heinz Züllighoven

Fachbereich Informatik - Softwaretecnik
Universität Hamburg
Vogt-Kölln-Str. 30
D-22527 Hamburg
+49 40 54 94-2413
zuellighoven@informatik.uni-hamburg.de

Motivation

Aktuell setzt sich zunehmend die Einsicht durch, daß sich Softwareentwicklung vorrangig an der äußeren, d.h. in der Verwendung sichtbaren Qualität orientieren sollte. Gebrauchsqualität zeigt sich in der fachlichen Stimmigkeit und der menschengerechten Handhabung von Anwendungssoftware (vgl. [Flo97]). Viele verschiedenen Techniken und Hilfsmittel werden eingesetzt in der Hoffnung, daß Analyse, Entwurf und Konstruktion von Software zu der angestrebten Gebrauchsqualität führen. Oft zeigt sich dann aber bei der Auslieferung und im Einsatz, daß die Vorstellungen der Entwickler von Gebrauchsqualität nicht mit den Bedürfnissen und Anforderungen der Anwender übereinstimmen.

Aus unserer Sicht liegt der Schlüssel zu einer wirklichen Gebrauchsqualität von Software im Anwendungsbereich, d.h. dort wo das Softwaresystem zur Unterstützung der täglichen Aufgaben eingesetzt werden soll. Die Entwickler müssen einerseits diese Aufgaben und die damit verbundenen Abläufe wirklich verstehen. Andererseits sollte das Softwareprodukt die wesentlichen Konzepte und Gegenstände des Anwendungsbereichs in für die Benutzer geeigneter Form repräsentieren.

Einsicht der Entwickler in die Aufgaben des Anwendungsbereichs und Repräsentation der fachlichen Konzepte in der Architektur und an der Benutzungsschnittstelle der Software sind die Voraussetzungen für ein hohes Maß an Gebrauchsqualität.

Dieser Beitrag stellt einen erprobten Ansatz der Anwendungssoftwareentwicklung vor, der explizit die Gebrauchsqualität zum Ziel hat (vgl. [Kil94]). Auf der Grundlage objektorientierter Entwurfs- und Konstruktionstechniken vereinigt er so verschiedene Bestandteile wie ein Leitmotiv mit Entwurfsmetaphern, anwendungsorientierte Dokumente und eine evolutionäre Vorgehensweise mit Prototyping. An mehreren Stellen in Forschung und Praxis eingesetzt und weiterentwickelt, ist er unter dem Namen „Werkzeug & Material-Ansatz“ (vgl. [Lie97]) bekannt geworden.

Gebrauchsqualität

Traditionell (und erfolgreich) hat Anwendungssoftware menschliche Routinetätigkeiten automatisiert. Menschliche Arbeit wurde durch Software weitgehend ersetzt oder bis auf wenige notwendige Dateneingaben reduziert. Beispiele im Bankenbereich sind Programme zur Geldüberweisung oder zur Berechnung von Kontogebühren. Diese Programme zeigten wenig anwendungsorientierte Merkmale oder Gebrauchsqualität und waren vorrangig auf „Datenverarbeitung“ ausgerichtet. Betrachten wir ihre Benutzungsschnittstelle, so kann die typische

großrechnerorientierte Kombination von Menüs und Bildschirmmasken kurz als „Fenster auf Daten“ bezeichnet werden.

Mit der Verbreitung graphischer Anwendungen auf Arbeitsplatzrechnern und PCs verschob sich auch der Fokus von der Automatisierung menschlicher Routine zur Unterstützung alltäglicher Arbeitsaufgaben. Dies lag nahe, denn die ersten graphischen Schnittstellen für Büroanwendungen waren ja bewußt nach der Metapher des Schreibtischs konzipiert worden. Benutzer fanden auf diesen elektronischen Schreibtischen Mappen, Formulare und Papierkörbe vor, also Dinge, die in der täglichen Arbeit nützlich sind. Dazu wurden mit Text- und Grafikeditoren die ersten mehr oder weniger generischen Softwarewerkzeuge geliefert, die bei der Bearbeitung der Arbeitsgegenstände halfen. Grafische Benutzungsschnittstellen waren also ein entscheidender Schritt auf dem Weg zur Gebrauchsqualität von Anwendungssoftware. Auf einmal wurde das Design von Software als „Gestaltung“ sichtbar und erfahrbar. Alltägliche Arbeitsformen ließen sich auf den Rechner übertragen: Ein Benutzer konnte gleichzeitig mit mehreren Werkzeugen an unterschiedlichen Gegenständen zur Erledigung einer Aufgabe arbeiten, um je nach Situation etwas „an die Seite“ zu legen und es später im selben Zustand wieder aufzugreifen.

Für die Softwareentwickler hatte das ebenfalls Konsequenzen. Offensichtlich mußten sie sich intensiver mit der Frage beschäftigen, welche Aufgaben Menschen im Rahmen ihrer Arbeit zu erledigen haben und vor allem, was sie dabei an Gegenständen verwenden. Diese Konzentration auf die Entwicklung von Arbeitsmitteln und -gegenständen auf einem elektronischen Schreibtisch hatte auch zur Folge, daß es nicht mehr notwendig war, in der Software Routinen für die Erledigung einer Aufgabe festzuschreiben. Die Benutzer bestimmten selbst, wann sie mit welchem Werkzeug welchen Gegenstand bearbeiten wollten. Eine zentrale Schwierigkeit bei der Softwareentwicklung war damit behoben - die Algorithmisierung komplexer und situationsabhängiger Arbeitsabläufe. Mit dem Verzicht auf die durchgängige Implementation von Arbeitsprozessen wurde es möglich, für immer komplexere Aufgabenstellungen und für neue Anwendungsbereiche Software zu entwickeln. Offensichtlich wurden dabei nicht mehr reine „Daten“ am Bildschirm mit den Eingeben-Ändern-Löschen-Operationen verarbeitet. Softwareprodukte wurden anwendungsspezifisch. Ihre Komponenten mußten entsprechend den Konzepten und Gegenständen des jeweiligen Anwendungsbereichs entworfen werden.

Dieser „Schub“ in der Softwareentwicklung führte dazu, daß in den achtziger Jahren die einflußreichsten Techniken und Methoden auf den Bereich der Entwicklung interaktiver Anwendungssoftware zielten. Mit Blick auf diese Tendenzen läßt sich *Gebrauchsqualität* so charakterisieren:

- Die Funktionalität des Systems orientiert sich an den Aufgaben aus dem Anwendungsbereich.
- Die Handhabung des Systems ist benutzergerecht.
- Die im System festgelegten Abläufe und Schritte lassen sich je nach Anwendungssituation problemlos an die tatsächlichen Anforderungen anpassen.

Mit diesem Verständnis von Gebrauchsqualität haben wir unseren sog. *„Werkzeug & Material-Ansatz“* entwickelt (siehe [Kil94, Gry96]), der eine evolutionäre Vorgehensweise mit den Prinzipien der Objektorientierung vereint. Die zentrale Idee des Werkzeug & Material-Ansatzes ist, die Gegenstände und Konzepte des Anwendungsbereichs als Grundlage des softwaretechnischen

Modells zu nehmen. Das Ergebnis soll eine enge Korrespondenz zwischen dem anwendungsfachlichen Begriffsgebäude und der Softwarearchitektur sein. Diese *Strukturähnlichkeit* hat zwei entscheidende Vorteile: Die Anwender finden die Gegenstände ihrer Arbeit und die Begriffe ihrer Fachsprache im Anwendungssystem repräsentiert. Sie können entsprechend ihre Arbeit in gewohnter Weise organisieren. Die Entwickler können Softwarekomponenten und Anwendungskonzepte bei fachlichen und softwaretechnischen Änderungen zueinander in Beziehung setzen und somit die wechselseitigen Abhängigkeiten erkennen.

Um diese inhaltliche und strukturelle Abbildung zwischen dem fachlichen und dem softwaretechnischen Modell leisten zu können, verwenden wir in unserem Ansatz ein Leitbild mit passenden Entwurfsmetaphern, anwendungsorientierte Dokumenttypen und eine am Prototyping orientierte Vorgehensweise.

Leitbild und Entwurfsmetaphern des Werkzeug & Material-Ansatzes

Interaktive Anwendungssoftware muß eine *aufgabengerechte Funktionalität* mit einer *geeigneten Handhabung und Präsentation* zusammenbringen. Aus dieser Sicht ist Softwareentwicklung eine Design-, besser Gestaltungsaufgabe im ursprünglichen Sinne: Form und Inhalt müssen zueinander passen. Dies ist für Softwareentwickler eine neue Herausforderung. Denn dadurch tritt neben die traditionelle softwaretechnische Aufgabe, einen geeigneten Algorithmus zur Erledigung einer fachlichen „Funktion“ zu finden und zu implementieren, die Frage, welche „Gestalt“ ein Softwareprodukt annehmen und wie es bei der Aufgabenerledigung gehandhabt werden soll. Bei dieser Gestaltungsaufgabe müssen Softwareentwickler unterstützt werden. Dies soll im Rahmen des Werkzeug & Material-Ansatzes das Leitbild leisten.

– Ein *Leitbild* ist in unserem Zusammenhang eine explizite Sichtweise bei der Softwareentwicklung. Es hilft den Anwendern und Entwicklern, die Anwendungssoftware zu verstehen und zu entwerfen. Ein Leitbild umfaßt auch immer Wertvorstellungen.

Ein allgemeines Leitbild sollte „griffig“ sein, damit es wirklich anschaulich wird. Dazu dient ein Satz von passenden Entwurfsmetaphern, die durch ihre „Bildhaftigkeit“ ein Leitbild konkretisieren.

– Eine *Entwurfsmetapher* beschreibt ein Konzept oder eine Komponente des Anwendungssystems durch einen Gegenstand der Alltagswelt. So schafft die Entwurfsmetapher eine Verständigungsbasis für Entwickler und Anwender - sie bezieht sich anschaulich auf den gemeinsamen Erfahrungshintergrund und stellt einen verständlichen Begriff zur Verfügung.

Die Idee, Metaphern für den Entwurf von Anwendungssoftware einzusetzen, ist nicht neu (vgl. [Car88]). Insbesondere in Skandinavien gibt es eine lebhafte Diskussion zu diesem Thema (vgl. [Ehn88]). Was unseren Ansatz vielleicht auszeichnet ist, daß wir einen zusammengehörigen Satz von Metaphern im Rahmen eines integrierenden und orientierenden Leitbildes verwenden. Dies wird im nächsten Abschnitt erläutert.

Leitbild und Entwurfsmetaphern für den interaktiven Arbeitsplatz

Unser häufigstes Leitbild ist der „Arbeitsplatz für eigenverantwortliche Tätigkeiten“. Dieses Leitbild hat sich in unseren zahlreichen Kooperationsprojekten im Banken- und Dienstleistungsbereich bewährt. Dort wird vorrangig Bürotätigkeit qualifizierter Mitarbeiter und Mitarbeiterinnen unterstützt - Tätigkeiten also, die neben Fachwissen und Erfahrung auch eine gewisse Eigenständigkeit und Initiative bei der Erledigung der täglichen Aufgaben verlangen. Dazu passende Entwurfsmetaphern sind Werkzeug, Material, Automat und Arbeitsumgebung (vgl. Abb. 1 und [Rie95,Bäu96]).

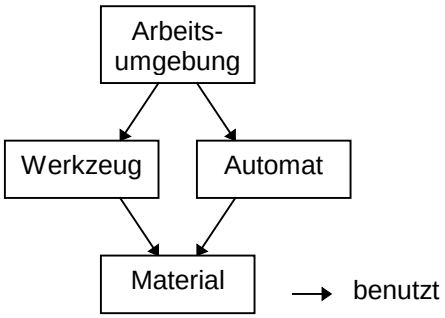

Abb. 1: Beispiele für Entwurfsmetaphern

Offensichtlich stammen diese zentralen Metaphern aus dem Kontext menschlicher Arbeit. Es ist eine allgemein Feststellung in der Arbeitspsychologie oder der Erkenntnistheorie, daß Menschen bei der Arbeit eine intuitive Unterscheidung treffen zwischen den Dingen, die ihr Arbeitsgegenstand sind und den Dingen, die sie zur Bearbeitung als Arbeitsmittel verwenden. Entsprechend haben wir die Begriffe Werkzeug und Material definiert (vgl. [Zül92] und Abb. 1):

- Ein *Werkzeug* unterstützt wiederkehrende Arbeitsabläufe und -handlungen. Es ist bei unterschiedlichen Aufgaben und Zielsetzungen nützlich. Ein Werkzeug wird von seinem Benutzer je nach den Erfordernissen einer Situation gehandhabt oder wieder zur Seite gelegt. Es schreibt keine festen Arbeitsabläufe vor. Als Softwarewerkzeug ermöglicht es den interaktiven Umgang mit den Arbeitsgegenständen.
- *Materialien* sind die Arbeitsgegenstände, die schließlich zum Arbeitsergebnis werden. Materialien werden mit Werkzeugen entsprechend bearbeitet. Softwarematerialien verkörpern „reine“ anwendungsfachliche Funktionalität. Sie werden niemals direkt benutzt und stellen sich auch nicht selbst dar. Ein Softwarematerial ist durch sein Verhalten, nicht durch seine Struktur charakterisiert.

In einer Büroumgebung gibt es nicht nur Werkzeuge und Materialien. Viele lästige Arbeitsroutinen werden heute maschinell, d.h. automatisch erledigt. Wir haben dazu die Entwurfsmetapher Automat eingeführt:

- Ein *Automat* erledigt eine vorab vollständig festgelegte Aufgabe und produziert ein definiertes Ergebnis. Einmal eingestellt, kann ein Automat über einen längeren Zeitraum ohne Eingriff von außen ablaufen. Softwareautomaten laufen meist im Hintergrund. Sie werden von ihren Benutzern initial eingestellt oder im Notfall gestoppt und haben darüber hinaus keine interaktive Schnittstelle.

Werkzeuge, Automaten und Materialien benötigen einen Ort, an dem sie bereitgestellt, angeordnet und aufbewahrt werden. Dazu haben wir die Metapher von der Arbeitsumgebung eingeführt:

- Eine *Arbeitsumgebung* ist der Ort, an dem Arbeitsaufgaben erledigt und die dazu benötigten Arbeitsgegenstände und -mittel bereitgestellt werden. Arbeitsumgebungen verkörpern ein Raumkonzept und Ordnungsprinzipien. Wir unterscheiden zwischen persönlichen Arbeitsplätzen, die Privatheit und eine individuelle Anordnung der Dinge ermöglichen und den allgemein zugänglichen Räumen einer Arbeitsumgebung.

Auf der Basis des Leitbilds vom Arbeitsplatz mit Werkzeugen, Automaten und Materialien haben wir an Universitäten und Forschungseinrichtungen und in unseren ersten Kooperationsprojekten erfolgreich Einzelarbeitsplätze mit Anwendungssoftware ausgestattet. Sie zeichneten sich durch ein hohes Maß an Gebrauchsqualität aus. Diese Erfolge führten bald zu der Forderung, auch kooperative Arbeit zu unterstützen. Darauf weist schon der Begriff der gemeinsamen Arbeitsumgebung hin. Im folgenden Abschnitt werden wir diesen Punkt weiter behandeln.

Entwurfsmetaphern für kooperative Arbeit

Im Kontext von Büroarbeit aber auch in vielen anderen Anwendungsgebieten läßt sich *kooperative Arbeit* durch einige Merkmale charakterisieren, die schon bei der Analyse von Arbeitssituationen ins Auge fallen (vgl. [Kra96]):

- Kooperation findet oft dadurch statt, daß Materialien ausgetauscht oder an einem bestimmten Ort plaziert werden. Häufig werden dabei Dokumente und Unterlagen in Ordnern, Mappen oder anderen Behältern zusammengefaßt.
- Die genauen Arbeitsabläufe mit diesen Materialien sind nicht im Einzelfall festgelegt. Oft gibt es eine Standard-Vorgehensweise, aber je nach Situation wird diese Vorgehensweise modifiziert oder durch eine völlig andere, die aber zum gleichen Ergebnis führt, ersetzt.
- Die Initiative oder Kontrolle über die einzelnen Arbeitsschritte innerhalb einer kooperativen Situation ist meist bei der Person, die aktuell den Zugriff auf das Material hat. Dies gilt zumindest für solche Situationen, in denen das Arbeitsmaterial noch anfaßbar ist und nicht, z.B. als elektronisches Dokument vorliegt. Workflow-Systeme, die die Kontrolle über die Arbeitsschritte übernehmen, sind hier nicht Gegenstand der Diskussion und liegen auch nicht in unserem Interesse.

Diese Beobachtungen legen die Schlußfolgerung nahe, daß kooperative Arbeit in ähnlicher Weise durch Arbeitsmittel und -gegenstände unterstützt werden kann, wie wir das für den individuellen Arbeitsplatz gezeigt haben. Dazu kommen allerdings neue Ressourcen, die dem speziellen Charakter arbeitsteiliger aber weitgehend selbstorganisierter Arbeit gerecht werden. Im folgenden stellen wir die Entwurfsmetaphern „Prozeßmuster“ und „Vorgangsmappen“ vor, die für die skizzierte kooperative Situation eine sinnvolle Unterstützung zu Werkzeug, Material, Automat und Umgebung bieten ([Gry96, GrW96]).

- Ein *Prozeßmuster* ist ein gemeinsam benutzes Material, das den Kooperationsprozeß vergegenständlicht. Alle an der Erledigung einer Aufgabe Beteiligten können an diesem Material den „Stand der Dinge“ ablesen. Dazu besteht ein Prozeßmuster aus einer Liste mit den Namen der Arbeitsschritte oder Teilaufgaben, die zu erledigen sind, und den Namen der jeweils zuständigen Personen oder Rollen. Ein Prozeßmuster läßt sich an eine *Vorgangsmappe* anheften, die die Materialien enthält, die jeweils Arbeitsgegenstand sind.
- Ein Prozeßmuster enthält zunächst den Standard- oder Routinefall zur Erledigung einer Aufgabe. Dieser *Standardfall* resultiert aus einer eingespielten Arbeitsteilung zwischen den Beteiligten. Das wesentliche Merkmal eines Prozeßmusters ist, daß sich diese Repräsentation des Standardfalls je nach Situation durch die Beteiligten *verändern* läßt.
- Der eigentliche Kooperationsprozeß findet in der Weitergabe der Vorgangsmappe mit dem angehängten Prozeßmuster statt. Dies bedeutet in unserer Terminologie, daß ein *Transportautomat* die Vorgangsmappe zur Arbeitsumgebung der Person oder Rolle schickt, die als nächstes im Prozeßmuster genannt ist. Die Vorgangsmappe liegt dann in dem entsprechenden Posteingangskorb bereit. Wenn der nächste Arbeitsschritt oder die Teilaufgabe erledigt ist, hakt die zuständige Person dies auf dem Prozeßmuster ab und legt die Vorgangsmappe in den Ausgangskorb. Es liegt im Ermessen der Person, ob die weiteren Teilaufgaben und Arbeitsschritte noch wie geplant erledigt werden sollen, oder ob der Vorgang im Prozeßmuster verändert werden muß.

Aus dieser Beschreibung sollte klar werden, daß die vorgestellten Konzepte zur Unterstützung kooperativer Arbeit passend zum Leitbild vom Arbeitsplatz für eigenverantwortliche und qualifizierte Arbeit sind und daß die zusätzlichen Entwurfsmetaphern sich recht nahtlos in den vorhandenen Satz von Werkzeug, Material und Automat einfügen. Damit kann der Werkzeug & Material-Ansatz in vielen Bereichen von Büroarbeit eingesetzt werden, in denen eine eingespielte Arbeitsteilung zwischen den Beteiligten besteht.

Das Leitbild vom Arbeitsplatz mit seinen Entwurfsmetaphern bedeutet eine klare Festlegung auf eine anwendungsorientierte Unterstützung von Arbeit im Gegensatz zur prozeßorientierten Automatisierung, die von den meisten Workflow-Systemen verfolgt wird. Auch auf dieser Ebene steht für uns die Gebrauchsqualität im Vordergrund, was bedeutet, daß die Benutzer als Experten im Anwendungsbereich gesehen werden, die je nach Situation entscheiden können, ob sie einen vorformulierten Standardvorgang bearbeiten oder ob spezifische Änderungen im Ablauf und in der Arbeitsaufteilung sinnvoll sind.

Für die Entwickler bedeutet Anwendungsorientierung, daß sie die Aufgaben verstehen müssen, die sie durch entsprechende Arbeitsumgebungen unterstützen sollen. Dazu müssen sie sich den

Zugang zum Fachwissen und zur Erfahrung der Anwendungsexperten verschaffen. Das Schlüsselwort heißt hier „evolutionäre Systementwicklung“, da diese auf eine enge Zusammenarbeit der Entwickler mit den Anwendern ausgerichtet ist. Zusammenarbeit muß aber eine Grundlage haben. Deshalb werden im nächsten Abschnitt einige anwendungsorientierte Dokumenttypen und das Konzept des Prototyping vorgestellt.

Anwendungsorientierte Dokumenttypen und Prototypen

Daß anwendungsorientierte Dokumenttypen eine notwendige Basis zum Verständnis der Konzepte und Aufgaben im Anwendungsbereich sind, ist keine neue Einsicht (vgl. [Car90, Jac92]). Aus unserer Sicht besteht die zentrale Anforderung, diese Dokumenttypen in der Fachsprache des Anwendungsbereichs zu formulieren; genauer, in der Sprache, die in der jeweiligen Organisation tatsächlich verwendet wird. Die im weiteren vorgestellten Dokumenttypen werden von den Entwicklern meist in der Prosa dieser Fachsprache verfaßt. Auch hier ist wieder eine Ausrichtung auf die anwendungsorientierte Gebrauchsqualität erkennbar. Wir verwenden Dokumenttypen, die in der Literatur unter verschiedenen Namen und teils mit leicht abweichender Bedeutung bekannt sind (vgl. [Bür95]).

- *Szenarios* beschreiben die aktuelle Arbeitssituation. Dabei liegt der Augenmerk auf den Aufgaben im Anwendungsbereich und der Art und Weise, wie die jeweiligen Aufgaben unter Verwendung von Arbeitsmitteln und Arbeitsgegenständen erledigt werden. Szenarios werden von den Entwicklern auf der Grundlage von Interviews mit Anwendern und anderen beteiligten Gruppen im Anwendungsbereich geschrieben. Sie können einen unterschiedlichen Detaillierungsgrad haben, von einer allgemeinen Übersicht über die Aufgaben an einem Arbeitsplatz bis zu detaillierten Handlungsstudien (s. Abb. 2).
- Ein *Glossar* definiert und rekonstruiert das Begriffsgebäude der jeweiligen Fachsprache. Die Glossareinträge werden von den Entwicklern parallel zu den Szenarios geschrieben. Sie geben erste Hinweise auf Materialien für den Systementwurf.
- *Systemvisionen* stehen am Übergang zwischen der Analyse des Anwendungsbereichs und der Konstruktion des zukünftigen Systems. Auf der Basis der Szenarios antizipieren sie zukünftige Arbeitssituationen. Sie richten sich vorrangig an Entwickler und helfen, eine gemeinsame „Vision“ des Systems aufzubauen. Sie beschreiben die Vorstellungen über die fachliche Funktionalität und die Handhabung von Werkzeugen, Materialien, Automaten und den anderen Komponenten des Systems. Dabei ist wichtig, daß diese Komponenten zu einer ganzheitlichen und konsistenten Vorstellung von der Unterstützung der jeweiligen Aufgaben und ihrer Erledigung zusammenspielen. Ähnlich wie Szenarios werden Systemvisionen in unterschiedlicher Detaillierung geschrieben (s. Abb. 3).
- Prototypen sind die wesentliche konstruktive Ergänzung der Systemvisionen. Durch sie werden die Ideen über das System „anfaßbar“. Vor allem sog. Funktionale Prototypen sind nicht nur für die Kommunikation mit den Anwendern zentral, sondern geben den Entwicklern die notwendige Erfahrungs- und Experimentiergrundlage für die technische Konstruktion des Systems. Sie sind

aus softwaretechnischen Komponenten aufgebaut, die den Entwurfsmetaphern entsprechen. Damit decken sich Programmkomponenten und fachliche Konzepte und Gegenstände, was eine wesentliche Voraussetzung für die Zusammenarbeit von Entwicklern und Anwendern ist.

Der Berater holt dann seinen *Beratungsordner, sucht* das *Produkt* über ein *Register* und *öffnet* den Ordner an der *entsprechenden Stelle*. Zum Beratungsordner gibt es noch einen *Formularordner*, in dem *Standardformulare* (z.B. *Vertrag zugunsten Dritter*) *abgelegt* sind und einen *Musterordner*, in dem *Ausfüllhilfen* für Verträge und *Schlüsselblätter abgelegt* sind

Abb. 2: Ein Szenario

Der Berater öffnet dann seinen Beratungsordner mit einem Doppelklick, wählt zunächst das Produkt über das sichtbare Inhaltsverzeichnis, wählt anschließend in einem zweiten Verzeichnis die gewünschte Variante mit einem Doppelklick und läßt sich dadurch gleich die zugehörige Verkaufshilfe anzeigen.

Abb. 3: Eine Systemvision

Das Problem beim Verwendung dieser Dokumenttypen liegt wie bei jeder Art von Dokumentation im bereits genannten Übergang von Ist zu Soll, d.h. bei der Projektion der gegenwärtigen Arbeitssituation auf die zukünftige unter Einsatz des neuen Anwendungssystems. Während die Dokumente, die wie Szenarios oder Glossareinträge an der aktuellen Situation orientiert sind, meist ohne große Probleme und mit Engagement zwischen Entwicklern und den Anwendungsexperten diskutiert werden können, stellt sich das bei der Antizipation des zukünftigen Systems schon anders dar. Die meisten Anwendungsexperten können sich anhand eines Dokuments wie einer Systemvision kein ausreichendes Bild vom Entwurf des Systems machen und sich deshalb auch nicht vorstellen, was der Einsatz dieses System bei der Erledigung von Aufgaben bedeutet. Auch wenn die Systemvisionen in Prosa geschrieben und um Oberflächenskizzen ergänzt sind, können besonders bei Projekten mit neuer technologischer Entwicklungsbasis kaum zuverlässige und konstruktive Einschätzungen allein anhand dieser Dokumente erwartet werden. Dies unterstreicht die herausragende Rolle des Prototyping. Auch dies ist nicht neu (vgl. [Lic94, Bud92]). Wesentlich aus unserer Sicht ist aber, daß der Bau von Prototypen die treibende Kraft während eines gesamten Entwicklungsprojektes darstellt. Dabei sind die ergänzenden Dokumente unverzichtbar. Denn Prototypen machen keine Aussagen über den intendierten Einsatzkontext, die wesentlichen Entwurfsentscheidungen oder die Art ihrer Handhabung. Deshalb sind Systemvisionen und andere, hier nicht beschriebene Entwurfsdokumente, notwendige Ergänzungen zu Prototypen.

Dokumenttypen für kooperative Arbeit

Für die Analyse kooperativer Arbeit und den Entwurf entsprechender Anwendungssoftware ist es notwendig, das Repertoire der beschriebenen Dokumenttypen zu erweitern. Wir haben festgestellt, daß sich mit Szenarios, Glossaren und Systemvisionen die verschiedenen Aufgaben und Aktivitäten sehr gut aus der Sicht der einzelnen Arbeitsplätze beschreiben lassen. Was fehlt ist die Gesamtsicht. Diese Gesamtsicht kooperativer Arbeit kann auch nicht einfach erstellt werden, da kaum einer der Anwendungsexperten sich mit dieser Frage in der täglichen Arbeit befassen muß. Zudem stellt die Menge der Szenarios für einen gesamten Arbeitsbereich, etwa eine Bank oder ein Krankenhaus, eine solche Textfülle dar, daß sie nicht alleinige Grundlage einer gemeinsamen Diskussion der Beteiligten über die Kooperationszusammenhänge sein kann. Um diesem Problem zu begegnen, haben wir erfolgreich sog. Kooperationsbilder eingesetzt (vgl. [Kra96]). Diese Bilder basieren im wesentlichen auf allgemeinverständlichen Piktogrammen und stellen dar, welche Arbeitsgegenstände und Informationen auf welchem Weg zwischen den Beteiligten ausgetauscht werden. Dabei können über Pfeile und Numerierungen einzelne Aufgaben und die Art ihrer Erledigung dargestellt werden (s. Abb. 4). Faßt man mehrere zusammengehörige Aufgaben in einem Bild zusammen und läßt die Nummern weg, dann ergibt sich ein Überblick über diskussionswürdige „Brennpunkte“ und „Schleifen“ in der Kommunikation und Kooperation.

Kooperationsbilder werden in gemeinsamen Sitzungen mit den Beteiligten erstellt und diskutiert. Sie dienen zum einen zur Repräsentation der Ist-Situation, können aber ebenso gut für die Darstellung der möglichen Veränderungen beim Einsatz des zukünftigen Systems verwendet werden.

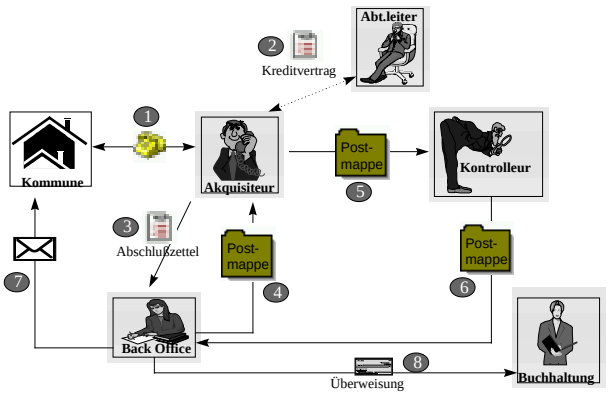

Abb. 4: Kooperationsbild zum Thema „Kreditvergabe“

Der Werkzeug & Material-Ansatz als Prozeß

Mit dem Leitbild, den Entwurfsmetaphern, den Dokumenttypen und dem Bau von Prototypen haben wir einen Satz zusammengehöriger Konzepte und Hilfsmittel, um Softwareentwicklung mit dem Fokus auf den Anwendungsbereich zu ermöglichen. Die Entwickler haben damit die Voraussetzungen an der Hand, um das gesteckte Ziel einer hohen Gebrauchsqualität anzusteuern. Aber erst die richtige Umsetzung der Konzepte und der Einsatz der Hilfsmittel im Entwicklungsprozeß entscheiden darüber, ob sich dieses Ziel wirklich erreichen läßt. Aus unserer Sicht ist es daher unerläßlich, den Entwicklern auch eine Anleitung für die Gestaltung dieses Prozesses zu geben. Das Grundprinzip dabei ist, Softwareentwicklung nicht als eine vorrangig technische oder formale Aufgabe, sondern als einen Lern- und Kommunikationsprozeß zu betrachten. In diesem Sinne haben wir ja bereits den Einsatz der einzelnen Dokumenttypen beschrieben. Dahinter steht der Gedanke, daß Lernen und Kommunikation sich als evolutionäre Prozesse darstellen, die durch ständige Rückkopplung zwischen den Beteiligten vorwärts getrieben werden.

In diesem revolutionären Prozeß zwischen Bearbeitung und Rückkopplung sind prinzipiell alle Dokumente eines Projektes einbezogen. Denn es geht nicht um die sequentielle Abarbeitung einer festgelegten Folge von Meilensteindokumenten, sondern um die Verknüpfung von analysierenden, modellierenden und bewertenden Aktivitäten (s. Abb. 5). Dies steht erkennbar im Widerspruch zu den Prinzipien der klassischen Wasserfall- oder Phasenmodelle, ist aber aus unserer Erfahrung eine wesentliche Voraussetzung für das Gelingen eines anwendungsorientierten Entwicklungsprojektes.

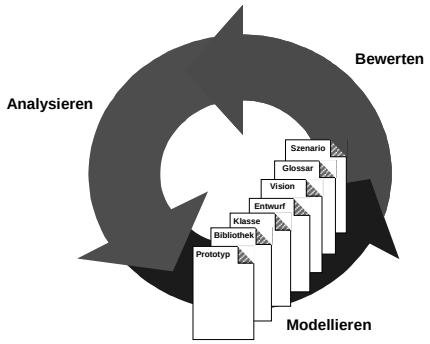

Abb. 5: Ein evolutionärer Entwicklungsprozeß mit Rückkopplungsschleifen

Der evolutionäre Prozeß spielt sich in unseren Projekten in der Form sog. Autor-Kritiker-Zyklen ab. Dabei sind die Arbeitsgegenstände die bereits genannten Dokumenttypen und Prototypen. Im Mittelpunkt steht der Wechsel zwischen Analysieren, Modellieren und Bewerten:
Durch die *Analyse* der Aufgaben im Anwendungsbereich erhalten die Entwickler einen ersten Zugang zu den Anwendungskonzepten, den damit verbundenen Gegenständen und Abläufen. Dieses Verständnis wird als fachliches oder technisches *Modell* in den Dokumenten und Prototypen repräsentiert. Zur *Bewertung* der unterschiedlichen modellierten Aspekten kommen jetzt die „Kritiker“ ins Spiel. Als solche fungieren vorrangig die Anwender und zukünftigen Benutzer, aber auch solche Experten, wie DV-Organisatoren, die aus ihrem jeweiligen Arbeitsbereich heraus am Entwicklungsprozeß beteiligt sind.
Nehmen wir ein Beispiel: Ein Entwicklerteam interviewt Bankberater an ihrem Arbeitsplatz (Analyse). Dies Ergebnisse des Interviews werden in Szenarios und Glossareinträgen festgehalten (Modellierung). Diese Dokumente werden von den Interviewpartnern und anderen Bänkern kommentiert (Bewertung). Diese Kommentare veranlassen die Entwickler, sich bankfachlich weiter einzuarbeiten und einzelne Nachinterviews zu führen. Dann schreiben sie Systemvisionen und erstellen einen ersten Handhabungsprototyp. Dieser Handhabungsprototyp wird in einem Workshop von Bankberatern bewertet. Die Beteiligten kommen zu dem Ergebnis, daß die Arbeit im Back Office bisher im Projekt vernachlässigt worden ist. Neue Interviewpartner aus diesem Bereich müssen befragt werden, etc. Auf diese Weise kommen die Autor-Kritiker-Zyklen zustande.
Zum Verständnis unserer Vorgehensweise ist es wichtig zu sehen, daß die jeweils identifizierten Probleme im Rückkopplungsprozeß die nachfolgenden Aktivitäten und die Auswahl der zu bearbeitenden Dokumente bestimmen. In diesem Sinne gibt es keine vordefinierte Reihenfolge von Dokumenten und Phasen. Als Idealvorstellung kann zu jedem Zeitpunkt prinzipiell jedes Dokument bearbeitet werden. Daß die Aktivitäten insgesamt in einem konkreten Projekt geplant und kontrolliert werden müssen und nicht einer individuellen Beliebigkeit ausgeliefert sind, steht außer Frage. Es sollte aber klar werden, daß der grundlegende Prozeß selbst wieder von anwendungsorientierten Fragestellungen und nicht von einem softwaretechnisch zentrierten Mechanismus bestimmt werden.

Schlußfolgerungen

In diesem Artikel haben wir die Gebrauchsqualität von Software als das vorrangige Ziel jedes Anwendungsprojektes herausgehoben. Gebrauchsqualität heißt für uns Anwendungsorientierung. Dem dienen ein entsprechendes Leitbild und seine Entwurfsmetaphern. Bei der Umsetzung erkennen wir die wesentlichen Vorteile der Objektorientierung, die eine unmittelbare Abbildung der Konzepte des Anwendungsbereichs auf die Komponenten eines Softwaresystems ermöglicht. Darüber hinaus haben wir die Rolle von anwendungsorientierten Dokumenttypen erläutert. Diese Orientierung am Anwendungsbereich bestimmt auch den Entwicklungsprozeß, der mit seinen evolutionären Autor-Kritiker-Zyklen erst die Einlösung des Anspruchs auf Gebrauchsqualität ermöglicht.

Das eigentliche Anwendungssystem ist im Rahmen des Leitbildes vom Arbeitsplatz für eigenverantwortliche und qualifiziert Tätigkeit nach den Entwurfsmetaphern Werkzeug, Material, Automat und Arbeitsumgebung aufgebaut. Wir haben festgestellt, daß durch die Unterstützung kooperative Arbeitssituationen neue Anforderungen an die Softwareentwicklung herangetragen werden. Konzeptionell haben wir dem im Bereich von Büroarbeit durch Metaphern wie Prozeßmuster und Vorgangsmappe in Verbindung mit einem Transportautomaten Rechnung getragen.

Um für die fachliche Modellierung eines Systems das notwendige Verständnis von den Aufgaben und Arbeitsabläufen zu erhalten, setzen wir als Entwickler anwendungsorientierte Dokumenttypen ein. Szenarios, Glossare und Systemvisionen werden ergänzt durch entsprechende Prototypen. Auch hier haben wir besondere Dokumenttypen, wie die Kooperationsbilder, um kooperative Arbeit zu modellieren und diskutierbar zu machen.

Insgesamt hat sich die Kombination der verschiedenen Komponenten unseres Werkzeug & Material-Ansatzes als tragfähig erwiesen. Außer großen und mittlerweile erfolgreich eingesetzten Anwendungssystemen im Bankenbereich haben wir gute Erfahrungen auch in eher technischen Bereichen wie Telefonie- oder Laborsysteme gesammelt.

Danksagung

Dieses Papier ist eine Überarbeitung von [Lie97]. Es faßt wesentliche Ideen von [Lie96, Bäu96, Kra96, GrW96] zusammen. Ich möchte neben den Autorinnen und Autoren dieser Arbeiten vor allem Carola Lilienthal für ihre Unterstützung danken.

Literatur

[Bäu96] Bäumer, D., Gryczan, G., Knoll, R., Züllighoven, H.: *Large Scale Object-oriented Software Development in a Banking Environment*. In: Cointe, P. (Ed.) ECOOP ‘96 - Object-Oriented Programming, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1996, pp. 73-90.

[Bür95] Bürkle, U., Gryczan, G., Züllighoven, H.: *Object Oriented System Development in a Banking Project: Methodology, Experience, and Conclusions*. Proceedings of HCI, Vol. 10, 1995, pp. 293-336.

[Bud92] Budde, R., Kautz, K., Kuhlenkamp, K., Züllighoven, H.: *Prototyping*. Springer-Verlag, Berlin, 1992.

[Car88] Carroll, J. M., Mack, R. L., Kellogg, W. A.: *Interface Metaphors and User Interface Design*. In Helander, M. (Ed.) Handbook of Human-Computer Interaction, 1988, pp. 283–307.

[Car90] Carroll, J. M., Rosson, M. B.: *Human Computer Interaction Scenarios as Design Representation.* Proceedings of the Hawaii International Conference on System Sciences, Los Alamitos CA IEEE Computer Society Press, 1990, pp. 555-561.

[Ehn88] Ehn, P. : *Work–oriented Design of Computer Artifacts*. Almquist and Wiksell International, Stockholm, 1988.

[Flo97] Floyd, C., Züllighoven, H.: *Softwaretechnik*. In: Rechenberg, P., Pomberger, G. (Hrsg.): Handbuch der Inormatik. Carl Hanser Verlag, München, im Druck.

[Gry96] Gryczan, G.: *Prozeßmuster zur Unterstützung kooperativer Tätigkeit.* Wiesbaden: DUV, Deutscher Universitätsverlag, 1996.

[GrW96] Gryczan, G, Wulf, M., Züllighoven, H: ***Prozeßmuster für die situierte Koordination kooperativer Arbeit.*** In: H. Krcmar/H. Lewe/G. Schwabe (Hg.) Herausforderung Telekooperation, DCSCW'96, Springer Verlag 1996. S. 89 - 103.

[Jac92] Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G.: *Object-oriented Software Engineering. A Use Case Driven Approach*. Addison-Wesley, Reading, 1992.

[Kil94] Kilberth, K., Gryczan, G., Züllighoven, H.: Objektorientierte Anwendungsentwicklung - Konzepte, Strategien, Erfahrungen. Vieweg, 1994.

[Kra96] Krabbel, A., Wetzel, I., Ratuski, S.: *Participation of Heterogeneous User Groups: Providing an Integrated Hospital Information System*, In: Blomberg, J., Kensing, F., Dykstra-Erickson, E. (Eds.): PDC'96 Proceedings of the Participatory Design Conference, Cambridge, Massachusetts, November 1996, pp. 241-250.

[Lic94] Lichter, H., Schneider-Hufschmidt, M., Züllighoven, H.: *Prototyping in Industrial Software Projects - Bridging the Gap Between Theory and Practice*. In IEEE Transactions on Software Engineering 20, 11, 1994, pp. 825-832.

[Lie96] Lilienthal, C., Züllighoven, H. : *Techniques and Tools for Continuous User Participation*. In: Blomberg, J., Kensing, F., Dykstra-Erickson, E. (Eds.): PDC'96 Proceedings of the Participatory Design Conference, Cambridge, Massachusetts, November 1996, pp. 153-159.

[Lie97] Lilienthal, C., Züllighoven, H. : *Application-Oriented Usage Quality - The Tools and Materials Approach*. Interactions, to appear.

[Rie95] Riehle, D., Züllighoven, H.: *A Pattern Language for Tool Construction and Integration Based on the Tools & Material Metaphor*. In Coplien, J. O., Schmidt, D. C. Pattern Languages of Program Design. Addison-Wesley, Reading, 1995, pp. 9-42.

[Rie96] Riehle, H. Züllighoven H.: *Understanding and Using Patterns in Software Development.* In: Karl Lieberherr/Roberto Zicari (eds.): Theory and Practice of Object Systems, Special Issue Patterns/Guest Editor: Steve Berczuk. Volume 2/Number 1, 1996, pp. 3-13.

[Zül92] Züllighoven, H.: *Umgang mit Software.* In: W. Coy et al. (Hrsg), Sichtweisen der Informatik. Vieweg, Braunschweig , 1992.