Paul Chlebek

User Interface-orientierte Softwarearchitektur

Aus dem Bereich IT erfolgreich gestalten

Visual Basic .NET mit Methode
von Heinrich Rottmann

Warum ausgerechnet .NET?
von Heinrich Rottmann

Requirements Analysis realisieren
von Karl Scharbert

Management der Software-Entwicklung
von Carl Steinweg

Das neue PL/I
von Eberhard Sturm

Projektmanagement der SW-Entwicklung
von Werner Mellis

Profikurs ABAP®
von Patrick Theobald

SAP R/3® Kommunikation mit RFC und Visual Basic
von Patrick Theobald

Six Sigma in der SW-Entwicklung
von Thomas Michael Fehlmann

Profikurs Eclipse 3
von Gottfried Wolmeringer und Thorsten Klein

Erfolgreiche Datenbankanwendung mit SQL3
von Jörg Fritze und Jürgen Marsch

Web-basierte Systemintegration
von Harry Marsh Sneed und Stephan Henry Sneed

Terminalserver mit Citrix Metaframe XP
von Thomas Joos

Exchange Server 2000
von Thomas Joos

Profikurs PHP-Nuke
von Jens Ferner

Unternehmensweites Datenmanagement
von Rolf Dippold, Andreas Meier, Walter Schnider und Klaus Schwinn

Netzarchitektur - Kompass für die Realisierung
von Thomas Spitz, Markus Blümle und Holger Wiedel

SIP - Die Technik
von Andreas Kanbach

IT-Sicherheit - Make or Buy
von Marco Kleiner, Lucas Müller und Mario Köhler

Mehr IT-Sicherheit durch Pen-Tests
von Enno Rey, Michael Thumann und Dominick Baier

IT-Risiko-Management mit System
von Hans-Peter Königs

IT-Sicherheit mit System
von Klaus-Rainer Müller

Der IT Security Manager
von Heinrich Kersten und Gerhard Klett

User Interface-orientierte Softwarearchitektur
von Paul Chlebek

www.vieweg.de

Paul Chlebek

User Interface-orientierte Softwarearchitektur

Bauentwurfslehre für interaktive Softwareoberflächen – Kompass für die Entwicklung dialogintensiver Anwendungen – Leitfaden für erlebbare User Interfaces

Mit 85 Abbildungen

vieweg

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Auslieferung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säurefreiem und chlorfrei gebleichtem Papier gedruckt. Die Einschweißfolie besteht aus Polyäthylen und damit aus organischen Grundstoffen, die weder bei der Herstellung noch bei der Verbrennung Schadstoffe freisetzen.

1. Auflage August 2006

Lektorat: Günter Schulz / Andrea Broßler

Der Vieweg-Verlag ist ein Unternehmen von Springer Science+Business Media.
www.vieweg.de

Konzeption und Layout des Umschlags: Ulrike Weigel, www.CorporateDesignGroup.de
Umschlagbild: Nina Faber de.sign, Wiesbaden
Druck- und buchbinderische Verarbeitung: MercedesDruck, Berlin
Printed in Germany

ISBN-10 3-8348-0162-3
ISBN-13 978-3-8348-0162-3

User Interface-Architektur

- Anleitung zur User Interface-orientierten Softwarearchitektur
- Bauentwurfslehre für durchdachte interaktive Softwareoberflächen
- Kompass für die benutzerzentrierte Entwicklung dialogintensiver Anwendungen
- Leitfaden für erlebbare und eindeutige User Interface-Spezifikationen

Später für Léon; jetzt für Markus.

Paul Chlebek

Über den Autor

Paul Chlebek ist Experte für UML (Unified Modelling Language) und XML (Extensible Markup Language). Der Schwerpunkt seiner derzeitigen Projekt- und Forschungsarbeit liegt im Entwickeln von Methoden und domänenspezifischen Sprachen für die Konstruktion von Softwareoberflächen.

Durch Verbindung der Spezialgebiete HCI (Mensch Maschine Schnittstellen), DSL (domänenspezifische Sprachen) und MDA (modellgetriebene Architektur) hat er die die User Interface-orientierte Softwarearchitektur aufgestellt.

Er bringt ein breites Spektrum an Erfahrungen aus der Projektpraxis im In- und Ausland in den Rollen Leitender Entwickler, Internationalisierungsingenieur, Projektleiter, Technischer Entwicklungsleiter, Softwarearchitekt und Testmanager mit.

Paul Chlebek arbeitet freiberuflich als Berater und Methodenentwickler für BMW und andere international tätige Unternehmen. Er ist Mitglied in der Gesellschaft für Informatik, in der International Association of Software Architects und im Berufsverband der deutschen Usability Professionals sowie Gründer des Open HCI Knowledge Projekts www.BenutzerOberflaeche.de.

Autorenhomepage

https://www.openbc.com/hp/Paul_Chlebek/

http://www.projectpeople.net/chlebek/

http://www.benutzeroberflaeche.de

Geleitwort

Geleitwort des Praxismentors

Dialoganforderungen werden komplexer

Als die Welt noch keine Dialog-Software kannte, waren die Anstrengungen von Software-Entwicklern auf die funktionale Seite und die Optimierung von Rechenzentrum-Elementen wie einfaches Operating und Datenspeicher-Minimierung gerichtet. Der Benutzer hatte mit der resultierenden Anwendung im Regelfall keine direkte Berührung gehabt und nur deren Resultate oder Wirkungen wahrgenommen. Die Entwicklung der Anwendung dauerte ihre Zeit, aber man lebte damit.

Nach der ersten Dialogisierung war eine interessante Feststellung zu machen: In dem Maße, in dem die Softwareentwicklung komplexer wurde, eine höhere Fertigungstiefe erhielt und Fehler oder konzeptionelle Missinterpretationen sofort sichtbar wurden, stiegen die Anforderungen an die Umsetzungszeit.

Je intensiver also die Benutzer Einfluss auf die Gestaltung nehmen konnten, umso schneller wollten sie Ergebnisse sehen, obwohl der Entwicklungsaufwand zunahm.

Diese Entwicklung ist bis heute nicht abgeschlossen, trotz (oder wegen) agilem Manifest, SOA und vielen anderen Technologie-Parametern.

Die Ausführungen des Autors zur User Interface-orientierten Architektur helfen uns, diesen Optimierungsweg weiter erfolgreich zu beschreiten.

Günter Penzenauer
R&B Consulting Group

Vorwort

Die ersten von Menschen gebauten Werkzeuge - damals, in der Steinzeit - waren nicht sonderlich kompliziert. Ausgefeilte Baupläne waren bei den glücklichen Ingenieuren dieser Zeit nicht groß in Mode: Man nahm einen Stein und hackte einfach auf einen anderen ein, bis dieser eine gefällige Form hatte.

Menschen bauen immer komplexere und größere Werkzeuge.

Später, als Baumeister und Ingenieure der Antike, der Renaissance und des Industriezeitalters große, komplizierte Gebäude und Maschinen zu errichten begannen, wurde das Zusammenwirken mehrerer Entwickler (z.B. Baumeister und seine Gesellen) und die Abstimmung mit dem Auftraggeber (z.B. König, Fürst, Fabrikant) unabdingbar. Damit wurde auch das gemeinsame Verstehen des geplanten Werkes wichtig, und so kamen ***Baupläne und Modelle*** ins Spiel.

Ausschlaggebend: Verstehen, wie das zu bauende Werk verwendet wird.

Beim Bauen von Werkzeugen und Maschinen ist das Verstehen der späteren ***Verwendung*** ausschlaggebend dafür, ob das Produkt praxistauglich wird. Das Wahrnehmen des Anwendungszwecks und der Anwendungsweise ist Vorbedingung für korrekte Entscheidungen der Konstrukteure.

Alles andere muss sich in die Weise, wie ein Produkt verwendet wird, einfügen: Die ***Funktionsweise***, die ***Form***, die ***Konstruktionsmittel***.

Dieses Buch handelt von der Zusammenarbeit und vom gegenseitigen Verstehen von Menschen, die gemeinsam komplizierte elektronische Maschinen beauftragen und bauen. Diese Maschinen, die man nicht in die Hand nehmen und anfassen kann, nennt man ***Software***.

Software ist eine Maschine, die man nur über ihre Benutzeroberfläche anfassen kann.

Software ist nicht aus Holz und nicht aus Stein. Die Gesetze der newtonschen ***Mechanik*** gelten für sie nicht; man kann sie also nicht über ihre mechanischen Eigenschaften wahrnehmen. Softwarefunktionen quietschen nicht beim Starten, Menü- und Symbolleisten biegen sich nicht unter der Last ihres Inhalts durch, und Dialogseiten können einem nicht auf den Fuß fallen.

Wenn Software solche mechanischen Eigenschaften hätte, könnten diese durchaus Aha-Erlebnisse bei Anwendern und Entwicklern auslösen.

Verwenden ist der Schlüssel zum Verstehen.

Das Zugangstor zum Wahrnehmen, Auslösen und zum Definieren der Softwarefunktionen ist jedoch allein die Art und Weise, in der man Software ***verwendet*** - ihre ***Benutzerschnittstelle*** (in Computerenglisch: ***User Interface***, abgekürzt „UI“).

Da das User Interface eine Schlüsselrolle spielt, ist demnach eine ***User Interface-orientierte Softwarearchitektur*** nötig, um die Softwareoberfläche im Entwicklungsvorgehen und in der Konstruktionsweise der Software stärker zu fokussieren, als es die heutigen Architekturen tun.

Das Buch über User Interface-orientierte Architektur

User Interface-orientierte Softwarearchitektur zeigt, wie man Softwareanwendungen für Menschen baut, indem man das Verwenden der Software in den Mittelpunkt der Abstimmung zwischen Auftraggeber und Entwicklern stellt. Die Verwendung findet über die Oberfläche statt - für den Anwender ist die Oberfläche die Software. Beschreibt man die Oberfläche im Detail - also das, wie sich die Software im Dialog mit dem Anwender verhält - dann ist auch die Funktion beschrieben, und Auftraggeber und Entwickler werden sich einig, was gebaut wird.

Bauentwurfslehre

Das Buch zeigt auf, welche Begriffe, Redewendungen und ***Beschreibungsstrukturen*** benötigt werden, um die Verwendung einer Softwareanwendung (nicht nur Use Cases, sondern die komplette Oberfläche einer Büroanwendung mit jedem Pieps, den sie von sich gibt) eindeutig und missverständnisfrei zu beschreiben.

Fokus auf kommerzielle Anwendungen

Der Fokus des Buchs liegt auf Oberflächen für ***kommerzielle***, ***wirtschaftlich orientierte Software***. Diese Anwendungen, etwa Verwaltungssoftware, Beratungs-, Verkaufs- und Buchungssysteme unterstützen Arbeitsabläufe der Unternehmen durch elektronische Workflows, dialoggestütztes Ausfüllen von Formularen sowie durch verschiedene Such- und Auswertungsfunktionen.

Heute werden solche Anwendungen in der Regel mit grafischen Benutzeroberflächen als Windows- oder als Browserapplikation implementiert. Diese Art von Softwareoberflächen kommt nach meiner Einschätzung in 90% aller geschäftorientierten Anwendungen vor.

Die UI-orientierte Architektur geht zum Teil auch auf Fragestellungen von sprachgesteuerten Dialogen und von eingebetteten Systemen (z.B. Radionavigationssysteme).

Andere User Interface-Arten wie freie Dialoge, Virtual Reality, Augmented Reality, Spiele, KI-Systeme sowie neue, experimentelle UI-Formen werden nur am Rande erwähnt, weil sie in der Praxis kommerzieller Projekte vergleichsweise selten vorkommen.

Softwareoberflächen gibt es heute überall. Sie werden aber mit wenig methodischer Unterstützung gebaut - gewissermaßen nach dem Freihandprinzip und nach der Art einer Hundehütte zusammengezimmert.

Verwenden ist der Schlüssel zum Beschreiben

Wie aber entwirft und konstruiert man das, was an der Oberfläche einer Software geschieht, oder geschehen soll, eindeutig und hinreichend genau? Mit welcher Kombination aus Methoden und Werkzeugen lässt sich das Geflecht aus Verwenden, Dialog, Wirkung, Erscheinen und Funktionieren planen und beschreiben?

Darüber gibt dieses Buch über User Interface-orientierte Architektur Auskunft und Anleitung. Das Buch zeigt Mittel und Wege auf, um Benutzeroberflächen strukturierter und professioneller zu entwickeln.

Mit dem Know-how aus dem Buch ist der Leser in der Lage, Anwendungen so zu entwickeln, dass jederzeit Klarheit und Sicherheit über die Funktionsweise und die Funktionalität der entwickelten Anwendung gegeben ist. Dadurch können Kommunikationsaufwände gesenkt, die Budgets und Zeitvorgaben des Entwicklungsprojekts abgesichert und die Anzahl der Änderungsaufträge minimiert werden.

Paul Chlebek
Affing, im Juni 2006

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1: Einführung

… in dem Sie erfahren, warum Sie dieses Buch lesen sollten und was Sie in seinen Kapiteln finden.

Kapitelziele

- *Verstehen der Problemstellungen und der Lösungsansätze der User Interface-orientierten Architektur.*
- *Überblick über Themen, Zielgruppe und Aufbau dieses Buchs.*

1.1 Mensch und Software

Anwendungsprogramme sind Dinge, die Computer für Menschen nutzbar machen. Durch sie kann der Computer dem Anwender bei seinen Aufgaben helfen, z.B. Buchhaltung machen, Termine organisieren, Kundenkontakte verwalten, Grafiken und Texte erstellen.

Was wir als Anwender von einer Anwendungssoftware wahrnehmen können, ist ausschließlich ihre Benutzerschnittstelle: Die so genannte ***Softwareoberfläche*** oder das ***User Interface***.

Für Anwender ist die Oberfläche die Software

Wenn wir über eine Software sprechen, sprechen wir darüber, was wir wahrnehmen (sehen, hören, etc.) und tun können (Knöpfe drücken, Menüpunkte wählen, Daten eingeben, etc.), also eben über die Oberfläche der Software. Für Anwender ist Software nur das, was sie wahrnehmen und steuern können.

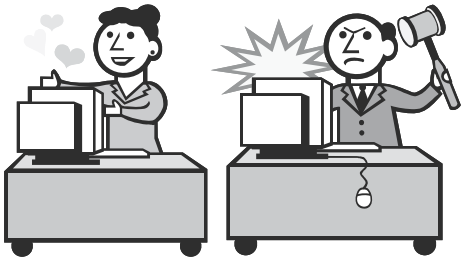


Bild 1: Software macht Computer für Menschen verwendbar.

Software soll die vielfältigen Funktionen des Computers für Menschen erschließen und leichter verwendbar machen. Durch ihre Oberfläche wird Software anfassbar und begreifbar. Hingegen ist für die Nutzer eines Computerprogramms - oder eines anderen vielseitigen Werkzeugs - dessen innerer Aufbau meist nur am Rande wichtig. Sie interessieren sich hauptsächlich dafür, ob das

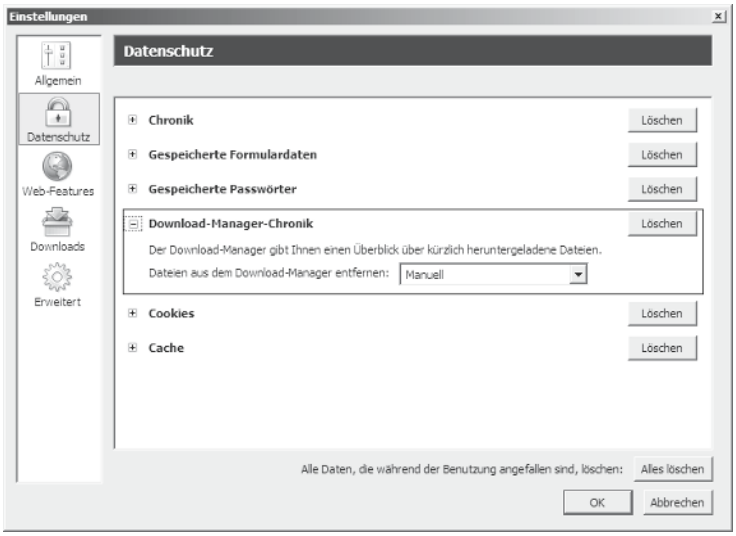
Ding gut in der Hand liegt und seinen Zweck gut erfüllt. Bei Software interessiert den Anwender die Oberfläche, bei Fahrzeugen das Fahrerlebnis, bei Maschinen das Bedienpult.

Gemeint ist: Das Verstehen einer Maschine erfolgt anhand dessen, wie die Maschine vom Anwender beim Benutzen erlebt wird. Das gilt sowohl für reale Maschinen wie einen Korkenzieher oder einen Toaster, als auch und insbesondere für virtuelle Softwaremaschinen und ihre Benutzeroberflächen, wie zum Beispiel ein Textverarbeitungsprogramm, eine Suchmaschine im Internet oder das Navigationssystem in einem Fahrzeug.

Funktionale Beschreibungen ergänzen Beschreibungen der Verwendung.

Anwendungssoftware ist ein Gerät, das nur durch das ***Verwenden*** der Softwareoberfläche für den Anwender materiell existiert. Also kann man Software hinreichend nur anhand der ***Interaktion*** des User Interface mit dem Anwender, nicht aber über den inneren Aufbau der Softwarekomponenten allgemein definieren, beschreiben, vermitteln und verstehen.

Bild 2: Software anhand der Verwendens der Oberfläche verstehen



Beschreibung des User Interface hat Vorrang vor funktionalen Spezifikationen.

In der Konsequenz heißt das aber: Datenmodelle und Flussdiagramme, Klassen- und Komponentendiagramme können eine Verwendungsbeschreibung sinnvoll ergänzen, sie können aber nicht von der Verwendungsbeschreibung losgelöst erstellt oder verstanden werden.

Daraus folgt, dass funktionale Spezifikationen und Modelle nachrangig zur Spezifikation des User Interface einer Softwareanwendung sind.

Beispiel für User Interface-Orientierung

Zum Beispiel ein Fernseher

Platt gesagt: Man kann zum Beispiel einen Fernseher nicht anhand seines Bauplans erklären, sondern nur darüber, dass man das Fernsehen als Tätigkeit beschreibt. Im ausgeschalteten Zustand sieht ein Computermonitor einem Fernsehgerät ja nicht unähnlich.

Die Aufgabe, den Fernseher anhand des Fernsehens zu beschreiben, ist ein Beispiel für das Wahrnehmen und Verstehen anhand der Verwendungsszenarien. Wir wissen, dass wir uns verschiedene Programme ansehen können, je nachdem, welche Tasten wir auf der Fernbedienung drücken. Wir wissen auch, wann was im Fernseher kommt, sofern wir eine Fernsehzeitung lesen. Wir können einen Kanal auswählen und gucken. Wenn das Antennenkabel eingesteckt ist, aha.

Am Anfang steht der Use Case

Man kann die Tätigkeit des Fernsehens nicht vermitteln, indem man den Schaltplan des Fernsehgeräts erklärt oder beschreibt, wie ein Fernsehstudio nebst Sender funktioniert. Vielmehr muss man die Interaktionen an der ***Bedienerschnittstelle*** des gedachten Anwendungssystems „Fernsehen“ aufzeigen.

Wissen über das Verwenden bestimmt die mögliche Qualität der technischen Umsetzung

Um das Fernsehen zu definieren, muss man also beschreiben, wie man fernsieht. Aus der Beschreibung der Tätigkeit und wie dabei Werkzeuge verwendet werden, leiten sich die Existenz, Funktion und Konstruktion dieser Werkzeuge ab. Am Ende dieses Ableitungsprozesses hat man den Schaltplan des Fernsehgeräts und die Fernbedienung gleich dazu.

Das Wissen darüber, wie man fernsieht, bestimmt, wie ein guter Fernseher und ein guter Fernsehsessel gemacht werden. Die Klarheit darüber, wie die Software an der Oberfläche verwendet wird, schafft Klarheit darüber, was technisch programmiert werden muss. Es funktioniert nicht umgekehrt, auch nicht bei Software.

Heute sind Softwareoberflächen des täglichen Gebrauchs ***Bedienpulte*** für die in der Software enthaltenen Anwendungsfunktionen.

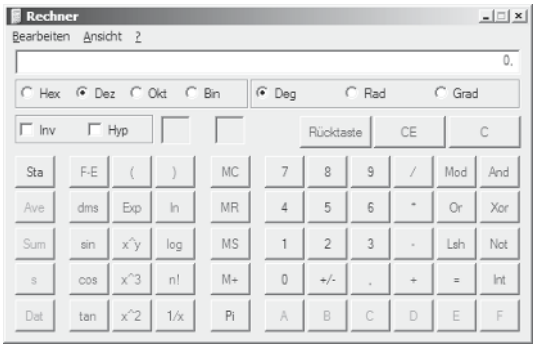


Bild 3: Oberflächen sind Bedienpulte für Funktionen der Software

In den meisten Anwendungen des Büroalltags wird der Anwender durch einen Dialogfluss aus auszufüllenden Formularen geführt. Er führt dabei die von der Anwendung vorgesehenen Schritte auf verschiedenen Bildschirmmasken durch, wählt Funktionsbereiche aus bzw. steuert den Anwendungsablauf innerhalb des von der Software vorgegebenen Angebots..

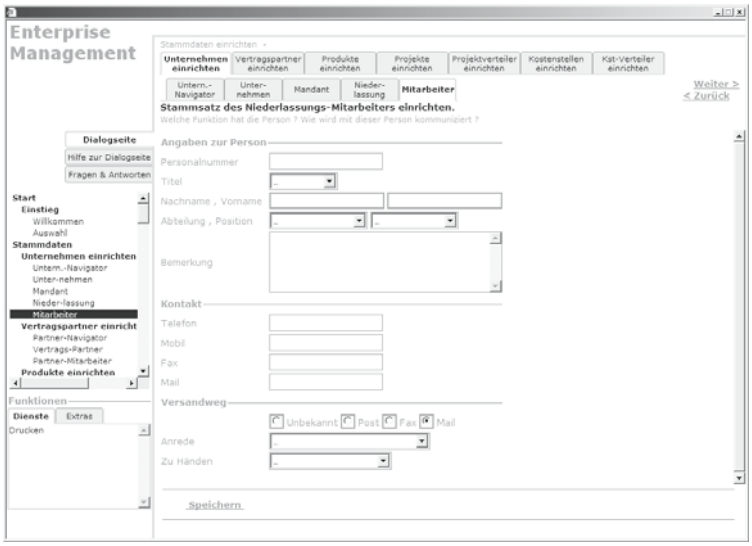


Bild 4: Typisches User Interface einer kommerziellen Dialoganwendung

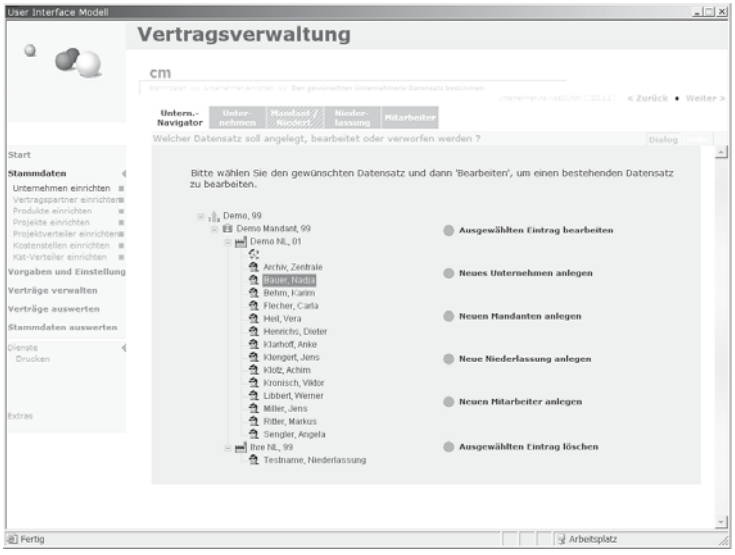


Bild 5: Navigieren in kommerziellen Anwendungen

Andere Formen von User Interfaces, z.B. der freie Dialog oder Augmented Reality sind in der Anwendungssoftware des täglichen Gebrauchs eher selten.

Eigenschaften, die über die Funktion eines guten Bedienpults hinausgehen, zum Beispiel die oft beschworene Mensch-Maschine-Kommunikation (wie zum Beispiel die zwischen Dr. David Bowman und dem Computer HAL) [Ebeling 1988, S.96f], werden von einer Anwendungsoberfläche gar nicht gewünscht.

Oder wollen Sie mit dem Navigationssystem in Ihrem Auto ausdiskutieren, ob es aus seiner Sicht Sinn macht, von A nach B über C zu fahren? Zumal eine intelligente Anwendung auch lügen darf, um den Turing Test zu bestehen [Rose 1985, S.62].

1.2 Zielgruppe

Wo Software entsteht, gibt es stets jene, die sie beauftragen und sie anschließend nutzen: ***Die Auftraggeber***. Und natürlich gibt es auch jene, die die Software herstellen: ***Die Entwickler***. Das Buch, das Sie aufgeschlagen haben, ist diesen beiden Zielgruppen zu gleichen Teilen gewidmet.

Für die, die Software herstellen, und für die, die Software beauftragen

User Interface-orientierte Softwarearchitektur ist für Leute, die Software herstellen: Programmierer, Architekten, Designer und Analytiker. Und sie ist für Leute, die die Herstellung von Software beauftragen und steuern: Projektleiter, Anwendervertreter, Anforderungssteller und Qualitätssicherer.

Zur primären Zielgruppe der User Interface-orientierten Architektur gehören:

- ***Softwareentwickler***
- ***Softwarearchitekten***
- ***Requirement Engineers (z.B. Spezifizierer und Analytiker)***

Weitere Zielgruppen des Buchs sind:

- ***Projektleiter***
- ***Anforderungssteller (z.B. Anwendervertreter und Fachexperten)***
- ***Qualitätssicherer und -prüfer***
- ***Designer***

Bild 6: Leute, die am Entstehen einer Software beteiligt sind

Es ist genau genommen ein Buch, das eine Brücke zwischen diesen Gruppen und Rollen baut. Diese Brücke sorgt für ***Verständigung*** und friedliches Zusammenleben dieser seit je her zu Missverständnissen im Umgang miteinander neigenden Völker.

User Interface-orientierte Softwarearchitektur zeigt die ***Kommunikationslücken*** zwischen den am Bau einer Softwareanwendung beteiligten Rollen auf. Zum Beispiel werden typische Verständigungsprobleme zwischen Softwarehersteller und Auftraggeber sowie Lösungsmöglichkeiten dafür aufgezeigt.

Bild 7: Gemeinsames Entwickeln macht Pläne und Modelle notwendig

Das Buch zeigt, wie die beteiligten Personen in ihren jeweiligen Rollen mit weniger Missverständnissen und Informationsverlusten gemeinsam an der Planung und Realisierung einer Software zusammenarbeiten können. Es stellt allgemeinverständliche und effiziente Möglichkeiten zur eindeutigen Beschreibung der Funktionen der Anwendung aus Anwendersicht zur Verfügung und zeigt, wie auf dieser Basis die Effizienz der Softwareentwicklung in allen Projektphasen entscheidend verbessert wird.

1.3 Nutzen der UI-orientierten Architektur

Methodensammlung für die Projektpraxis

Die User Interface-orientierte Softwarearchitektur ist auf die Praxis der Softwareentwicklung gerichtet. Das Buch aus Erfahrungen entstanden, die im Projektalltag gemacht wurden.

Kommerzielle Softwareprojekte haben stets enge Zeitvorgaben, begrenzte Budgets und müssen im vorgegebenen Umfeld mit Menschen gemacht werden, die unterschiedliche, ebenfalls begrenzte Fähigkeiten mitbringen. Man kann es sich nicht aussuchen: Auf Auftraggeberseite gibt der Marktdruck („Time To Market“) das Tempo vor; auf Herstellerseite ist das eigene technische Know-how bestimmend für die Wahl der Werkzeuge und Methoden. Auf beiden Seiten spielen die Kosten und die Verfügbarkeit von Ressourcen eine entscheidende Rolle.

Eine keimfreie Atmosphäre, in der man Lehrbuchwissen unter Laborbedingungen anwenden könnte, findet man im Projektge-

schäft nicht. Das Buch konzentriert sich daher auf alltägliche Projektsituationen, auf Beispiele aus der Praxis und auf konkrete Herausforderungen der Entwicklung von komplexen und dialogintensiven Anwendungen unter Zeit- und Erfolgsdruck. Theorien und Betrachtungen über den Besenstiel, z.B. Vorgehensmodelle und Metamodelle formaler Sprachen bleiben weitgehend außen vor.

Im Projekt zählt die Praxistauglichkeit in einem Umfeld, bei dem das Vorgehensmodell und die Werkzeuge, die Programmiersprachen und die Laufzeitplattform weitgehend vorgegeben sind. Es geht also darum, wie man mit gegebenen Mitteln und in einem gegebenen Projektumfeld das Projektziel sicherer erreicht. Best Practice eben.

Ich verspreche Ihnen nicht, dass Sie nach dem Lesen der User Interface-orientierten Architektur mehr Qualität in kürzerer Zeit erreichen oder die Projektkosten senken. Mehr und bessere Turnierkrokodile in kürzerer Zeit mit weniger Geld: Das funktioniert nur in der New Economy und die hat ja nur für wenige funktioniert.

Planungssicherheit

Ich verspreche Ihnen stattdessen, dass Sie besser schlafen können, weil Sie genauer wissen, was Sie eigentlich bauen oder gebaut kriegen. Sie schlafen umso besser, weil Sie auch sicher sein können, dass Ihr Projektpartner - Ihr Auftraggeber, Ihr Softwarehersteller - das gleiche Verständnis davon hat, wie die Software, die Sie gemeinsam erarbeiten, aussieht und funktioniert.

Sicherung der Projektziele

Die Sicherung der Projektziele wird erreicht, indem Risiken systematisch eliminiert werden. Das kann mit folgenden Maßnahmen erreicht werden.

- Abgestimmte Schnittstellen
- Erlebbarer User Interface-Entwurf
- Durchgängige und eindeutige Spezifikation von der Softwareoberfläche bis zu den Systemfunktionen.

In Bezug auf die Oberfläche kann die Sicherung der Projektziele durch folgende Eigenschaften der Oberflächenspezifikation erreicht werden.

- Formulierbarkeit
- Erlebbarkeit
- Nachvollziehbarkeit
- Durchgängigkeit

Doppelarbeit vermeiden

Sie können mit der User Interface-orientierten Softwarearchitektur keine Arbeit sparen, dafür aber Zeit.

Wenn Sie etwas bauen, z.B. einen Schreibtisch, können Sie nicht, sagen wir mal, die Tischplatte ungehobelt lassen, wenn Sie diese am Ende schleifen und polieren wollen.

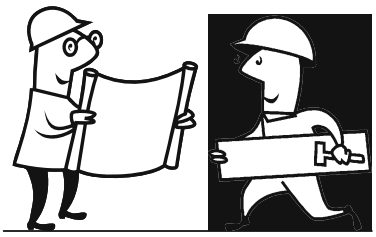
Das heißt: Alle zu einem Entwicklungsprozess gehörenden Arbeitsgänge müssen durchgeführt werden. Das Buch hilft Ihnen allerdings dabei, die Entwicklungsarbeit im gemeinsamen Kontext und mit der größtmöglichen Transparenz für alle Projektbeteiligten zu erledigen. Sie schlafen ziemlich ruhig, wenn das sichergestellt ist.

1.4 Benutzeroberfläche in den Mittelpunkt rücken

Missverständnisse bei Auftragsvergabe und Systementwurf verursachen oft erhebliche Folgekosten. Die Vermeidung von Missverständnissen lässt sich durch eindeutige Oberflächenbeschreibungen, so genannte ***Dialogbaupläne*** erreichen. Eindeutige Dialogbaupläne braucht man, um Softwareoberflächen gemäß Kundenbestellung zu programmieren.

Dialogbaupläne beschreiben das User Interface eindeutig und helfen, Missverständnisse zu vermeiden.

Bild 8: Dialogbaupläne helfen, Missverständnisse zu vermeiden.



Das Buch ist eine Anleitung zum Erstellen von Dialogbauplänen.

Die Oberfläche ist das, was Kunden und Anwender von der Software sehen; für sie ist ***das*** die eigentliche Software. Sie teilen den Entwicklern ihre Anforderungen an die Software über ihre Vorstellung von dem Geschehen an der Bedienerschnittstelle mit. Deshalb ist ein für Kunde und Softwarehersteller gleichermaßen verständlicher, eindeutiger, genauer und einvernehmlicher Bauplan der Oberfläche ausschlaggebend für den Erfolg eines Softwareprojektes.

Pläne und technische Zeichnungen sind schon beim Bau von Häusern, Schiffen, Automobilen und anderen Maschinen unabdingbar. Während der Planungsphase werden Draufsichten, Seitenansichten, perspektivische Ansichten und verschiedene Querschnitte des Bauvorhabens erstellt. Ohne sie wäre die Beauftragung, Genehmigung und Durchführung des Bauvorhabens undenkbar.