RWTH AACHEN UNIVERSITY

Bringing Usability to Industrial Control Systems

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

In Cooperation with
FEV Motorentechnik GmbH

by
Marcus Reul

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr.-Ing. Stefan Pischinger

Registration date: Dez 01st, 2008
Submission date: Jun 02th, 2009

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Aachen, im Juni 2009
Marcus Reul

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Überblick

Benutzerfreundlichkeit und das Nutzungserlebnis der Endanwender spielen in vielen Bereichen des Konsumentenmarktes eine immer wichtigere Rolle. Mehr und mehr Elektronik- und Unterhaltungs-Produkte werden heutzutage mit Fokus auf eine möglichst einfache Bedienung entwickelt.

Unsere Erfahrung zeigt jedoch, dass im industriellen Sektor Softwareanwendungen weiterhin viel mehr mit dem Hauptaugenmerk auf Funktionalität entwickelt werden, wobei eine einfache und effiziente Bedienung in vielen Fällen auf der Strecke bleiben. Teilweise ist dies auf die in diesem Marktsegment vorherrschenden Entwicklungsprozesse zurückzuführen. Während immer mehr Unternehmen des Konsumentenmarktes iterative Software Entwicklungsprozesse einführen, verwenden die meisten Industrieunternehmen noch sequentielle Software-Entwicklungsprozesse.

Der in dieser Arbeit vorgestellte Ansatz soll unabhängig von der Art des Entwicklungsprozesses die Benutzbarkeit der entwickelten Software verbessern. Entwurfsmuster sollen Produkt-Managern und Software-Entwicklern dabei helfen, Programmoberflächen und Benutzerschnittstellen einfacher zu gestalten.

Um den Ansatz auf seine Anwendbarkeit und Effektivität in der Praxis zu überprüfen, kooperierten wir mit FEV Motorentechnik, einem führenden Entwickler von Verbrennungsmotoren.

Im Rahmen einer Feldstudie habe ich das Anwendungsgebiet der Prüfstandsautomatisierungssysteme mitsamt seinen Benutzern und ihren Bedürfnissen kennengelernt. Basierend auf den Ergebnissen aus dieser Feldstudie habe ich in der Literatur nach geeigneten Entwurfsmustern für Benutzerschnittstellen gesucht. Resultat dieser Recherche ist ein Katalog mit 18 Entwurfsmustern, welche auf aktuell bestehende Benutzbarkeits-Probleme der Prüfstandssoftware eingehen.

In Form eines User Interface Design Workshops wurde ausprobiert, ob diese Entwurfsmuster tatsächlich Entwicklern und Produkt Managern beim Entwerfen einer neuen Programmoberfläche behilflich sein können.

Abstract

During the last years, usability has become recognized as a key competitive factor across many market sectors. Especially web agencies and companies from the consumer market realized that a good usability and a compelling user experience of their products can increase sales and market shares.

Although this change of thinking draws through many different branches, the industrial sector remains relatively unaffected from this new movement. Software applications from the industrial sector are build much more with focus on functionalities instead on an easy and efficient use. We believe that this is partly due to the software development processes that dominate this market segment. Whereas more and more companies from the consumer market implement quite successfully iterative software development processes, most industrial companies still use sequential development processes.

In this thesis, we present an approach that uses human-computer interaction (HCI) design patterns as an transportation medium to bring domain oriented usability knowledge to developers and product managers. We believe that this approach can boost usability independent of the chosen software development process.

To examine the applicability and effectiveness of this approach in practice, we cooperate with FEV Motorentechnik, a leading engineering service provider, specialized in the development of combustion engines.

Within a contextual inquiry, I became familiar with the field of test bench automation systems and its users. Based on the gained findings, I looked in the literature for domain-appropriate HCI design patterns. Result of this research is a pattern catalogue that consists of 18 HCI design patterns.

Later, I organized a user interface design workshop to examine if the patterns can really help developers and product managers while designing new user interfaces.

Dank

Herzlich danken möchte ich allen, die mich während meiner Arbeit an diesem Projekt tatkräftig unterstützt haben.

Ein ganz besonderer Dank gilt meinen beiden Betreuern, Jonathan Diehl und Matthias Salmen, die sich regelmäßig viel Zeit für mich genommen haben und deren Feedback und Ideen maßgeblich für den Fortgang dieser Arbeit waren.

Bedanken möchte ich mich auch bei Prof. Dr. Jan Borchers. Er hat mich durch seine anschaulichen und lebendigen Vorlesungen für das interessante Gebiet der Mensch-Maschine-Interaktion begeistern können.

Auch Herrn Prof. Dr.-Ing. Stefan Pischinger, der das Zweitgutachten dieser Arbeit übernommen hat, möchte ich meinen Dank aussprechen.

Zu guter Letzt möchte ich mich bei allen Interviewpartnern, Workshop-Teilnehmern und Kollegen der FEV Motorentechnik bedanken. Ohne sie hätte dieser Arbeit der Praxisbezug gefehlt.

Konventionen

In dieser Arbeit werden die folgenden Konventionen verwendet:

Exkurse erscheinen in orangefarbenen Kästchen:

EXKURS:
Exkurse beinhalten detaillierte Zusatzinformationen und stellen eine in den Text eingefügte selbständige und in sich geschlossene Abschweifung dar.

Exkurs:
Exkurs

Kapitel 1

Einleitung

"Design is directed toward human beings. To design is to solve human problems by identifying them and executing the best solution."

—Ivan Chermayeff

1.1 Motivation

Anwenderfreundlichkeit und Benutzbarkeit (engl. usability) rückten während der letzten Jahre mehr und mehr in den Fokus führender Elektronikunternehmen. Da die Schnittstelle zwischen Mensch und Maschine heutzutage oft durch Software realisiert wird, entstanden vor allem im Bereich der Softwareentwicklung neue Methoden und Prozesse, die eine einfachere Bedienung und ein besseres Nutungserlebnis der Endprodukte versprechen.

Usability wird zum Wettbewerbsfaktor im Konsumentenmarkt

Vor allem Hersteller des Konsumentenmarktes haben während der letzten Jahre diese neuen Methoden relativ erfolgreich in ihre Prozesse integrieren können und sehen die Anwenderfreundlichkeit ihrer Produkte inzwischen als einen wichtigen Wettbewerbsvorteil, welcher auch mehr und mehr durch die Werbung kommuniziert wird. Der kalifornische Elektronikhersteller Apple wird von vielen als Vorreiter in diesem Gebiet angesehen.

Wie Donald A. Norman [1999] feststellte, wird Benutzerfreundlichkeit vor allem dann wichtig, wenn die anfängliche Begeisterung für eine Technologie erlischt und die Kunden mehr auf eine angenehme und effiziente Nutzbarkeit als auf technische Neuheiten und Funktionsmerkmale achten. Heutzutage erreichen immer mehr Produkte diesen Reifegrad, woraus sich ergibt, dass der Entwicklungsfokus sich mehr und mehr in Richtung Anwenderfreundlichkeit und Nutzungserlebnis verschiebt.

Im industriellen Sektor sind Usability Methoden noch nicht verbreitet

Obwohl sich dieses Umdenken durch viele verschiedene Branchen zieht, blieb der industrielle Sektor bis heute relativ unberührt von dieser Bewegung. Gespräche mit Fachleuten zeigten, dass die Softwareentwicklung in dieser Branche noch viel stärker durch funktionale Anforderungen bestimmt wird. Dadurch bleiben qualitative Anforderungen (wie die einfache Bedienung der Systeme) oft auf der Strecke.

Industrieanlagen werden für viele verschiedene Aufgaben verwendet. Die wohl bekanntesten Vertreter dieser Gruppe sind die sogenannten Fertigungsanlagen. Obschon die einzelnen Arten von Industrieanlagen unterschiedlichste Zwecke erfüllen, teilen sie auch einige Gemeinsamkeiten. So verursachen Bedienungsfehler oft hohe Kosten und können in Extremfällen sogar die Gesundheit und Sicherheit von Menschen und Umgebung gefährden.

Software stellt oft die Schnittstelle zum Benutzer dar

Genau wie bei den Konsumentenprodukten, spielt Software in Industrieanlagen eine immer wichtigere Rolle und stellt auch hier oft die Schnittstelle zwischen System und Benutzer dar. Oft ist diese Steuerungssoftware jedoch nicht auf die Belange der Benutzer ausgelegt. Die kann unter anderem an den in dieser Branche vorherrschenden, sequentiellen Entwicklungsprozessen liegen.

Iterative Entwicklungsprozesse fördern Usability

Jacob Nielsen [1993a] erklärte, dass es nahezu unmöglich ist, eine Benutzerschnittstelle zu entwerfen, welche direkt im ersten Anlauf frei von jeglichen Benutzbarkeits-Problemen ist. Der Einsatz iterativer Entwicklungsprozesse ist die naheliegende Lösung zu diesem Problem. Iterative Entwicklung von Benutzerschnittstellen bedeutet eine kontinuierliche Verbesserung der Entwürfe anhand der Ergebnisse aus Benutzer-Tests und anderen Evaluierungsmetho-

den. Eine frühzeitige Erprobung der Benutzerschnittstelle ermöglicht es, Design-Fehler in einem Stadium zu erkennen, in dem Änderungen noch mühelos möglich sind.

Ein verbreitetes Modell eines iterativen Entwicklungsprozesses ist der DIA Kreislauf (engl. DIA cycle) [Nielsen, 1993b]. Der DIA Kreislauf besteht aus drei Phasen *(siehe Abbildung 1.1)*, welche nacheinander durchlaufen werden:

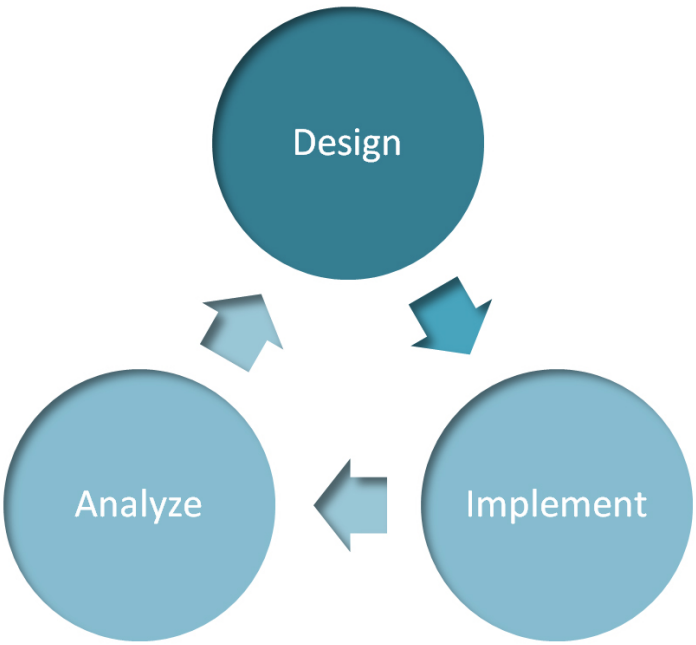


Abbildung 1.1: Der DIA Kreislauf

- **Design:** In der ersten Phase einer jeden Iteration wird die Benutzerschnittstelle entworfen, beziehungsweise verbessert. Basierend auf den Evaluierungsergebnissen aus dem vorherigen Iterationsschritt wird der Entwurf verfeinert und Usability-Probleme werden behoben.

- **Implement:** In der zweiten Phase wird eine lauffähige Version der Benutzerschnittstelle implementiert. In den ersten Iterationsschritten geschieht dies meist noch in Form von Prototypen.

- **Analyze:** Am Ende eines jeden Iterationsschrittes wird die Benutzerschnittstelle evaluiert. Benutzer-Tests sind eine mögliche Evaluierungsmethode.

Dieser Kreislauf wird so oft durchlaufen, bis ein akzeptables Endergebnis vorliegt – sprich: bis keine weiteren Fehler im Design mehr gefunden werden.

Die ISO empfiehlt iterative Entwicklungsprozesse

Die Internationale Standardisierungsorganisation ISO empfiehlt in ihrem Standard für Benutzer-orientierte Entwicklungsprozesse einen ähnlichen Prozess. ISO 13407 [1999] beschreibt einen iterativen Entwicklungsprozess, welcher aus fünf Phasen besteht. Ähnlich wie beim DIA Kreislauf sind vier der Phasen in einem Kreislauf zusammengefasst.

Industrieunternehmen halten an sequentiellen Prozessen fest

Experten scheinen sich einig zu sein, dass iterative Entwicklungsprozesse zu einer besseren Anwenderfreundlichkeit und weniger Usability-Problemen führen. Gespräche mit Branchenkennern zeigten jedoch, dass viele Industrieunternehmen an den herkömmlichen, sequentiellen Entwicklungsprozessen festhalten. Bekannte Beispiele sind v-Modell- und Wasserfall-Entwicklungsprozesse.

Nun stellt sich die Frage, warum Firmen aus dem Industriesektor zögern, iterative Entwicklungsprozesse einzuführen. Dies kann verschiedene Gründe haben. Im folgenden Abschnitt möchte ich die meiner Meinung nach relevantesten Aspekte kurz erläutern:

- **Umstrukturierungskosten:** Einen neuen Entwicklungsprozess implementieren, verursacht in erster Linie Kosten. Der neue Entwicklungsprozess muss auf die vorhandenen Firmenstrukturen abgestimmt werden und erfordert Umstrukturierungen in der Organisationsstruktur des Unternehmens. Solche tiefgehenden Änderungen verlangen eine gründliche und vorausschauende Planung sowie die Mitarbeit aller betroffenen Personen.

- **Durchführungskosten:** Iterative Entwicklungsprozesse erfordern eine regelmäßige Evaluierung der

Benutzerschnittstelle. In herkömmlichen Prozess-Modellen sind Tests nur am Ende der Entwicklung vorgesehen. Die kontinuierliche Evaluierung erzeugt einen erhöhten Entwicklungsaufwand, was wiederum mit Kosten verbunden ist. Vor allem Benutzer-Tests sind in der Praxis schwierig durchzuführen. Die entwickelten Systeme sind meist sehr spezifisch und potentielle Benutzer relativ rar. Zudem müssen die Benutzer während der Tests von der Arbeit befreit werden, was wiederum Kosten verursacht und Organisationsaufwand bedeutet.

- **Verträge:** In der Industrie existieren starke Beziehungen zwischen Firmen und ihren Kunden sowie Zulieferern. Teile der Entwicklung werden oft entlang der Wertschöpfungskette in andere Unternehmen ausgelagert. So kommt es nicht selten vor, dass ein Unternehmen die Spezifikation für ein Softwareprodukt entwickelt, die Implementierung aber durch ein anderes Unternehmen erledigt wird [Schäuffele, 2005]. Diese Form der Arbeitsteilung verlangt klar definierte Schnittstellen zwischen den einzelnen Unternehmen. Verträge bilden die rechtliche Basis einer solchen Kooperation. Grundlage dieser Verträge sind meistens Dokumente, wie das Lastenheft und das Pflichtenheft.

  Herkömmliche Entwicklungsprozesse definieren klare Stichtage für diese Dokumente. Spätere Änderungen sind nur in Form des sehr aufwändigen Change Request Managements möglich. Iterative Entwicklungsprozesse lassen jedoch Änderungen jederzeit zu. In diesem Fall wird der Hauptvorteil der iterativen Prozesse zu einem großen Nachteil. Unternehmen, die Hand in Hand mit ihren Zulieferern arbeiten, benötigen festgesetzte und bindende Verträge um im Nachhinein überprüfen zu können, ob die Arbeit wie vereinbart erledigt wurde.

Die obengenannten Aspekte zeigen, dass Unternehmen aus dem industriellen Sektor einige einleuchtende Argumente dafür haben, die bestehenden Entwicklungsprozesse beizubehalten anstelle iterative Entwicklungsprozesse einzuführen. Wie bereits erläutert, bergen herkömmliche Ent-

wicklungsprozesse jedoch einige Nachteile hinsichtlich der Benutzbarkeit der entwickelten Produkte.

Entwickler übersehen oft Usability-Probleme

Ein weiterer Aspekt, welcher die Anwenderfreundlichkeit der Software beeinträchtigt, ist der Mangel an Usability-Sensibilität und –Fachwissen bei Entwicklern und Produkt Managern. Meistens bestehen die Entwickler und Produkt Management Teams aus Experten des jeweiligen Einsatzgebietes des Software. Experten sind jedoch oft mit der Thematik so sehr vertraut, dass sie über mögliche Benutzbarkeitsprobleme hinweg sehen [Dix et al., 2003].

1.2 Der Ansatz

Richtlinien sollen Entwicklern helfen, Usability zu fördern

Mein Ziel ist es, dass Entwickler und Produkt Manager ein Gespür für mögliche Benutzbarkeitsprobleme entwickeln, indem sie Anwendungsgebiet-spezifisches Usability-Fachwissen an die Hand gegeben bekommen. So möchte ich erreichen, dass sich die Benutzbarkeit der entwickelten Software verbessert - bei Beibehaltung des etablierten Entwicklungsprozesses.

Das Medium, welches benötigt wird, um dieses Fachwissen zu übertragen, sollen sogenannte Entwurfsmuster für Benutzerschnittstellen (engl. HCI design patterns) bilden. HCI Design Patterns wurden von Jan Borchers in [1999] vorgestellt und sind ein relativ neuer Ansatz zum Aufheben und Transportieren von Usability-Fachwissen. Patterns beschreiben bewährte Ansätze und Problemlösungen des Interaktionsdesigns in einfacher und verständlicher Weise.

Unternehmens-bezogene Richtlinien sollen selektiert werden

In der Literatur existieren bereits einige umfangreiche Sammlungen solcher Entwurfsmuster. Jedes Anwendungsgebiet hat jedoch aufgrund der unterschiedlichen Benutzergruppen und Aufgaben seine eigenen Interaktionsprinzipien und Anforderungen an eine Benutzerschnittstelle. Der erste Schritt des von mir beschriebenen Ansatzes besteht darin, solche für das Anwendungsgebiet geeigneten Entwurfsmuster zu selektieren. Um das Anwendungsgebiet sowie die Aufgaben und Anforderungen der einzelnen Benutzer näher kennen zu lernen, schlage ich die im

dritten Kapitel näher beschriebene Methode der Contextual Inquiry vor. Nachdem die Anforderungen der Benutzer bekannt sind, kann eine Menge an für dieses Anwendungsgebiet relevanten Entwurfsmustern ausgewählt werden. Die Entwurfsmuster werden schlussendlich Produkt Managern und Entwicklern an die Hand gegeben.

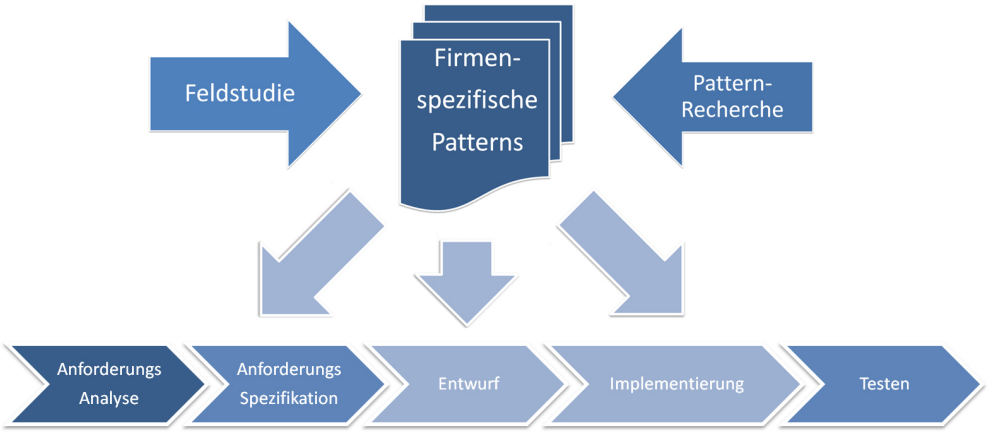


Abbildung 1.2: Unser Ansatz

In meiner Arbeit möchte ich untersuchen, ob und inwiefern solche Entwurfsmuster Produkt Managern und Entwicklern beim Entwerfen von Benutzerschnittstellen behilflich sein können. Diese Arbeit entstand im Rahmen meiner Tätigkeit bei der FEV Motorentechnik, einem weltweit führenden Motorenentwickler aus dem Aachener Raum. Die Zusammenarbeit mit diesem Unternehmen ermöglichte es mir, den vorgestellten Ansatz in der Praxis zu erproben.

Der Ansatz soll in der Praxis ausprobiert werden

1.3 Das Anwendungsgebiet

1.3.1 FEV Motorentechnik

FEV Motorentechnik ist ein Ingenieurdienstleister, welcher sich auf die Entwicklung von Verbrennungsmotoren spezialisiert hat. Neben der Entwicklung von Motoren gehören

auch die Optimierung von bestehenden Motoren sowie die Entwicklung von spezialisiertem Mess- und Testapparaturen zu den Hauptgeschäftstätigkeiten des Unternehmens.

FEV entwickelt spezialisierte Test-Systeme

Das Testen ist ein wichtiger Bestandteil der Motoren-Entwicklung und –Optimierung. Neue Motoren-Designs und Ansteuerungsverfahren müssen ausgiebig in den sogenannten Motorenprüfständen getestet werden, bevor sie in einem Fahrzeug integriert auch auf der Straße überprüft werden können. Spezielle Test-Systeme sind notwendig, um and die benötigten Messergebnisse zu gelangen. FEV Motorentechnik entwickelt und baut zu diesem Zweck verschiedenste Messapparaturen. Angefangen bei Sensoren und Konditionieranlagen, über ganze Test-Zellen bis hin zu Softwareanwendungen, die diese Messapparaturen steuern und die Messdaten aufnehmen. Eine solche Softwareanwendung ist das von der FEV entwickelte TCM System.

In der Praxis existieren verschiedene Arten von Motorenprüfständen. So unterscheidet man zum Beispiel zwischen Prüfständen für die Motorenentwicklung, für die Verbrennungsanalyse und Prüfständen für die akustische Analyse eines Motors. Eine weitere Art der Prüfstände sind die sogenannten end-of-line Prüfstände. Diese werden in der Motorenherstellung zur Funktions- und Qualitätskontrolle eingesetzt.

Exkurs: *Konditionieranlagen*

KONDITIONIERANLAGEN:
Umgebungsbedingungen, wie zum Beispiel die Lufttemperatur und –feuchtigkeit sowie die Kraftstofftemperatur haben einen großen Einfluss auf das Verhalten des Motors und somit auch auf die Messergebnisse. Damit jedoch jederzeit vergleichbare Messergebnisse erzeugt werden, werden sogenannte Konditionieranlagen eingesetzt. Die Konditionieranlagen bringen Luftfeuchtigkeit, Kraftstofftemperatur auf vorher festgelegte Basiswerte. So kann gewährleistet werden, dass Messungen, die im Winter durchgeführt werden mit denen aus dem Sommer vergleichbar sind.

TEST-ZELLE:
Die Test-Zelle ist ein abgeschlossener Raum und beherbergt den Motor sowie die Sensorik und einen Teil der Testapparaturen während des Tests. FEV Motorentechnik entwickelt und fertigt modulare Test-Zellen, die für verschiedene Arten von Motorenprüfständen verwendet werden können.

Exkurs: *Test-Zelle*

1.3.2 Das TCM System

Der Test Cell Manager (TCM) ist ein sogenanntes Prüfstandsautomatisierungssystem. Dieses System ist für die Steuerung der Versuchsanlage und des Versuchsobjekts (der Motor) sowie für die Verarbeitung und Sammlung der Messdaten verantwortlich. Neben TCM existieren noch andere Systeme in der FEV Automatisierungslandschaft. TCM kommuniziert zum Beispiel mit dem Motor über den sogenannten Test Object Manager (TOM). TOM stellt Basisfunktionalitäten für die Steuerung des Versuchsobjekts und der Lastmaschine zur Verfügung.

LASTMASCHINE:
Anders als das Auto auf der Straße, hat ein Motor auf dem Motorenprüfstand per se keinen Widerstand. Um jedoch den Motor bei verschiedenen Lastbedingungen zu untersuchen, wird eine sogenannte Lastmaschine an den Antriebsstrang gehängt. So können beispielsweise Berg- und Talfahrten simuliert werden. Bei Bergfahrten erzeugt die Maschine eine der Motordrehung entgegen gesetzte Last. Bei Talfahrten befindet sich der Motor im sogenannten Schleppbetrieb: die Lastmaschine dreht in gleicher Richtung wie der Motor.

Exkurs: *Lastmaschine*

TCM stellt die Hauptschnittstelle zwischen Prüfstand und Benutzer dar. Obwohl TCM ein mächtiges System ist, stand es in der Vergangenheit häufiger in der Kritik. Vor allem die Benutzbarkeit des Systems wurde bemängelt. TCM wird von der FEV selbst eingesetzt aber auch an Kunden verkauft. Das System ist eine Neuentwicklung und stellt den Nachfolger des älteren Adapt Systems dar. Adapt wird

TCM stellt die Benutzerschnittstelle dar

heute noch bei der FEV an vielen Prüfständen eingesetzt. Es werden jedoch immer mehr Adapt Prüfstände auf TCM umgerüstet – mit dem Ziel, dass Adapt mittelfristig komplett verschwindet.

1.3.3 Der FEV Entwicklungsprozess

FEV nutzt einen sequentiellen Entwicklungsprozess

Sowohl TCM als auch TOM und die anderen Softwareprodukte der TCM-Produktfamilie werden in der Test-Systems Abteilung der FEV entwickelt. Die FEV verwendet hierzu einen sequentiellen Softwareentwicklungsprozess, welcher den CMMI Reifegrad 2 erreicht hat. In diesem Abschnitt möchte ich die wesentlichen Phasen des FEV-Softwareentwicklungsprozesses erläutern:

- **Sammeln und Analysieren der Produkt Ideen:** Zu Beginn eines jeden Entwicklungsprojekts werden Produkt Ideen gesammelt. Beiträge dazu können aus verschiedenen Bereichen kommen. Sowohl Marktanforderungen, als auch Unternehmensziele spielen dabei eine Rolle. Kundenresonanz, interne Anforderungen der FEV, Informationen über den Wettbewerb, Zukünftige Trends und Gesetzesvorschriften müssen dabei berücksichtigt werden. Zum Schluss dieser Phase definiert das Produkt Management die Richtung, welche die Entwicklung einschlagen soll.

- **Lastenheft:** Als nächster Schritt wird das sogenannte Lastenheft erstellt. Diese Aufgabe ist wieder dem Produkt Management zuzuschreiben. Dazu müssen die Produkt-Ideen zuerst hinsichtlich ihrer Durchführbarkeit untersucht werden. Auch Marktanforderungen, Wettbewerb, Kosten- und Risikoabschätzungen spielen hier eine Rolle.

  Während der eigentlichen Erstellung des Lastenhefts müssen Kundenanforderungen und Wünsche sowie Produktanforderungen bestimmt werden. Erwartungen der Stakeholder werden dabei gesammelt und in Kundenanforderungen umgewandelt. Aus den Produktanforderungen werden die einzelnen Komponenten-Anforderungen abgeleitet. Das

Lastenheft definiert letzten Endes die Gesamtheit der Forderungen an das Produkt. Wenn die Benutzeroberfläche ein entscheidendes Merkmal des späteren Produkts darstellt, werden schon im Lastenheft kleine Skizzen oder Screenshots abgebildet, die das Aussehen der späteren Benutzeroberfläche darstellen.

- **Pflichtenheft:** Das Pflichtenheft wird von den Entwicklern erstellt und enthält die aufgrund des Lastenhefts entwickelten Realisierungsvorgaben. Das Pflichtenheft spezifiziert konkret, wie die im Lastenheft geforderten Merkmale technisch umgesetzt werden sollen. Dazu gehört auch der Entwurf der Programmoberfläche.

- **Realisierung:** Die Realisierung wird wieder von den Entwicklern vorgenommen und umfasst die eigentliche Implementierung (Programmierung) des Produkts. Manchmal kommt es vor, dass während der Implementierung noch Entwurfsentscheidungen gefällt werden müssen. Dies ist der Fall, wenn das Pflichtenheft noch Fragen bezüglich der Realisierung offen lässt.

- **Verifikation:** Während der Verifikationsphase wird überprüft, ob das entwickelte Produkt das Pflichtenheft erfüllt. Die Verifikation wird vom sogenannten Testing-Team durchgeführt.

- **Validierung:** Während der Validierung wird überprüft, ob das Produkt den am Anfang des Projekts aufgestellten Kundenanforderungen gerecht wird. Eine Validierung findet meist direkt am Prüfstand und unter realen Bedingungen statt.

1.4 Kapitelübersicht

- **Kapitel 1:** Im ersten Kapitel erläutere ich, weshalb die Anwenderfreundlichkeit von Industrieanlagen in vielen Fällen hinter der von Konsumentenprodukten zurückliegt. Daraufhin erkläre ich den von mir im Team mit meinen Betreuern entwickelten Ansatz, Anwendungsgebiets-spezifische HCI Design

Patterns während der Entwicklung einzusetzen, um so Programmoberflächen konsistenter und benutzbarer zu gestalten. Zum Schluss des Kapitels gebe ich eine Einleitung zu den Prüfstandsautomatisierungssystemen, dem Gebiet, in welchem ich den vorgestellten Ansatz erprobt habe.

- **Kapitel 2:** Im zweiten Kapitel erläutere ich verwandte Arbeiten, auf die ich bei meiner Literaturrecherche gestoßen bin. In einem ersten Teil werden verschiedene Arten von Usability-Leitfäden und Richtlinien vorgestellt. Der zweite Teil geht auf einen Ansatz zum Erstellen von Firmen-spezifischen Leitfäden für die Gestaltung von Benutzeroberflächen ein.

- **Kapitel 3:** In diesem Kapitel beschreibe ich die Feldstudie, welche ich im Bereich der Prüfstandsautomatisierungssysteme durchgeführt habe. Zuerst erläutere ich die Contextual Inquiry, eine Interview-Methode, welche ich für die Feldstudie verwendet habe. Daraufhin beschreibe ich die einzelnen Benutzergruppen der Prüfstandsautomatisierungssysteme und leite aus den in der Feldstudie gewonnenen Informationen Bedürfnisse der verschiedenen Anwendergruppen ab. Zum Schluss beschreibe ich einige der Interaktionsprinzipien des TCM-Systems.

- **Kapitel 4:** Das vierte Kapitel beschreibt die Vorgehensweise, die ich für die Auswahl der Patterns gewählt habe. Hier erläutere ich, welche Pattern-Sammlungen ich bei der Suche berücksichtigt habe und welche Patterns schließlich ausgewählt wurden. Zu jedem der 18 ausgewählten Patterns gebe ich eine kurze inhaltliche Beschreibung und erläutere, warum es für das Gebiet der Prüfstandsautomatisierungssysteme relevant ist.

- **Kapitel 5:** In diesem Kapitel beschreibe ich den Workshop, welchen ich zur Evaluierung unseres Ansatzes durchgeführt habe. Im zweiten Teil des Kapitels gehe ich auf die Ergebnisse ein.

- **Kapitel 6:** Das letzte Kapitel fasst die in dieser Arbeit gefundenen Ergebnisse noch einmal zusammen und spricht einige weiterführende Fragen an, welche sich aus den Ergebnissen dieser Arbeit ergeben haben.

Kapitel 2

Verwandte Arbeiten

"I never have found the perfect quote. At best I have been able to find a string of quotations which merely circle the ineffable idea I seek to express."

—Caldwell O'Keefe

Wissen über den Entwurf von einfach zu bedienenden Benutzerschnittstellen zu sammeln und in komprimierter Form niederzuschreiben, ist kein neuer Ansatz. So existieren heutzutage eine Vielzahl an Richtlinienkatalogen und Heuristiken, sowie etliche Sammlungen an Entwurfsmustern.

In diesem Kapitel möchte ich einige der existierenden Regeln und Richtlinien vorstellen. Dabei unterscheide ich zwischen vier Grundarten:

Es existieren verschiedene Arten von Richtlinien

- Usability-Prinzipien (engl. usability rules, heuristics)
- Leitfäden (engl. UI guidelines, UI styleguides)
- Normen / Standards
- Entwurfsmuster zur Gestaltung von Benutzerschnittstellen (engl. HCI design patterns)

2.1 Usability-Prinzipien

Viele Design- und Usability-Experten haben Regel-Sammlungen veröffentlicht, welche die essentiellen Grundprinzipien einfach zu bedienender Benutzerschnittstellen darlegen. Einige der bekanntesten Regelwerke möchte ich in diesem Abschnitt vorstellen.

2.1.1 Wilfred J. Hansen, 1972

Hansen entwickelte eine der ersten Richtlinien-Sammlungen

Die ältesten Usability-Prinzipien, auf die ich bei meinen Recherchen gestoßen bin, sind jene von Wilfred J. Hansen aus dem Jahr [1971]. Er entwickelte während seiner Arbeit an „Emily“, einem Textverarbeitungssystem, folgenden Satz von Usability-Prinzipien:

1. **First principle: Know the user**
2. **Minimize memorization**
   (a) **Selection not entry**
   (b) **Names not Numbers**
   (c) **Predictable behavior**
   (d) **Access to system information**
3. **Optimize operations**
   (a) **Rapid execution of common operations**
   (b) **Display inertia**
   (c) **Muscle Memory**
   (d) **Reorganize command parameters**
4. **Engineer for errors**
   (a) **Good error messages**
   (b) **Engineer out the common errors**
   (c) **Reversible actions**
   (d) **Redundancy**
   (e) **Data structure integrity**

Hansen entwickelte diese Usability-Prinzipien basierend auf seinen Erfahrungen mit anderen Textverarbeitungssystemen und verfeinerte sie während des Verlaufs seiner Arbeit an „Emily“. Sein oberstes Grundprinzip für die Entwicklung von einfach zu bedienenden Programmoberflächen ist die Kenntnis über den Benutzer. Wünsche, Eigenschaften und Schwächen der potentiellen Benutzer zu kennen ist laut Hansen essentiell um einfach zu bedienenden Benutzerschnittstellen zu entwerfen. Tatsächlich ist dies eines der Usability-Grundprinzipien, welches noch heute zu den am häufigsten genannten Merkmalen einer benutzerorientierten Anwendungsentwicklung gehört.

Er entwickelte die Richtlinien währen der Arbeit an einem Textverarbeitungssystem

Drei weitere Prinzipien sollen dabei helfen diese oberste Grundregel zu erfüllen: *Minimiere die Gedächtnislast*, *Optimiere Abläufe* und *Entwickle mit Hinblick auf auftretende Fehler*. Hansen hat auch diese Prinzipien weiter aufgeschlüsselt und ihnen neue, konkretere Regeln untergeordnet, durch deren Berücksichtigung die einzelnen Prinzipien erfüllt werden können.

Interessant an Hansens Regel-Sammlung ist die hierarchische Struktur. Untergeordnete Regeln geben konkretere Auskunft darüber, wie ein bestimmtes Prinzip während dem Entwurf berücksichtigt werden kann. Die hierarchische Struktur ist ein Aspekt, der später in den sogenannten Pattern Languages wieder auftritt.

Die Richtlinien haben eine hierarchische Struktur

2.1.2 Donald A. Norman, 1988

In seinem Buch „The Psychology of Everyday Things“ [1988], präsentierte Donald A. Norman folgende sieben Usability-Prinzipien:

1. **Use knowledge in the world and in the head**
2. **Simplify task structures**
3. **Make things visible, bridge the gulfs of execution and evaluation**
4. **Use natural mappings**

5. **Use natural and artificial constraints**
6. **Design for error**
7. **When all else fails, standardize**

Normans Regeln beziehen sich auf Beispiele im Buch

In diesen Prinzipien greift Norman auf einige der im Buch vorgestellten Konzepte zurück. Normans Buch enthält eine Fülle von Beispielen, welche die vorgestellten Prinzipien verdeutlichen und Anregungen dazu geben, wie man diese umsetzen kann.

2.1.3 Ben Shneiderman, 1992

Die „8 goldenen Regeln des Interaktionsdesigns“ von Ben Shneiderman aus dem Jahr [1992] gehören heutzutage zu den am meisten referenzierten Usability Regeln.

1. **Strive for consistency**
2. **Enable frequent users to use short cuts**
3. **Offer informative feedback**
4. **Design dialogue to yield closure**
5. **Offer simple error handling**
6. **Permit easy reversal of actions**
7. **Support internal focus of control**
8. **Reduce short-term memory load**

Shneiderman erklärt, dass diese Regeln möglicherweise nicht in allen Situationen einsetzbar sind. Sie sollen aber eine hilfreiche Checkliste mit Kernaussagen aus dem Bereich der Mensch-Maschine Interaktion darstellen. Shneiderman bemerkt, dass Designer, welche diese Regeln beherzigen, bessere Systeme erstellen als diejenigen, die sie ignorieren.

2.1.4 Jakob Nielsen, 1994

Auch Jakob Nielsen präsentierte [1994] eine Zusammenstellung von Usability-Prinzipien (bzw. Usability-Heuristiken, wie er sie nannte). Die vorgestellten Heuristiken waren jedoch nicht neu. In den 90er Jahren gab es schon eine Vielzahl verschiedener Usability-Prinzipien. Nielsen wollte diejenigen Prinzipien zusammentragen, welche die meisten Usability-Probleme erklären.

Nielsens Ziel war es, Usability-Prinzipien herzuleiten, die für eine heuristische Evaluierung besonders geeignet sind. Die heuristische Evaluierung ist eine Methode zur Bewertung von Benutzerschnittstellen hinsichtlich ihrer Benutzbarkeit. Bei der heuristischen Evaluierung wird die Benutzschnittstelle mithilfe bekannter Usability-Prinzipien überprüft. Mögliche Verletzungen dieser Regeln und daraus resultierende Usability Probleme sollen dabei aufgedeckt werden.

Nielsen selektierte die relevantesten Regeln mit einer Faktorenanalyse

Welche Heuristiken die meisten real auftretenden Benutzbarkeits-Probleme beschreiben, wollte Nielsen in einer Studie herausfinden. Dazu hat er eine Datenbank mit 249 Benutzbarkeits-Problemen aus 11 seiner früheren Projekte mit insgesamt 101 Usability-Heuristiken verglichen. Die zusammengetragenen Heuristiken haben unterschiedliche Herkünfte und wurden ursprünglich für unterschiedliche Zwecke entwickelt. Nielsen wollte somit sicherstellen, dass möglichst viele Sichtweisen auf das Thema Benutzbarkeit berücksichtigt wurden. Mithilfe einer Faktorenanalyse leitete er die folgenden Usability-Prinzipien her:

1. **Visibility of system status**
2. **Match between system and the real world**
3. **User control and freedom**
4. **Consistency and standards**
5. **Error prevention**
6. **Recognition rather than recall**

7. **Flexibility and efficiency of use**
8. **Aesthetic and minimalist design**
9. **Help users recognize, diagnose, and recover from errors**

2.1.5 Usability-Prinzipien in der Praxis

Alle vorgestellten Usability-Prinzipien und Heuristiken scheinen inhaltlich und vor allem vom Umfang her relativ ähnlich. Die Regelwerke beschreiben kurz und prägnant, was eine gut bedienbare Benutzerschnittstelle ausmacht. Durch die Kürze dieser Regelwerke behält der Leser leicht den Überblick und es vereinfacht ihm, sich die vorgestellten Prinzipien zu merken.

Die Regeln geben keine Tipps für die Umsetzung

Allerdings geben diese Regelwerke keine Auskunft darüber, wie die einzelnen Prinzipien in der Praxis umsetzbar sind. Die Usability-Prinzipien von Hansen geben durch ihren hierarchischen Aufbau zwar konkretere Vorschläge, wie einzelne Regeln umsetzbar sind, bleiben aber durch ihre Kürze noch sehr allgemein.

Ein Beispiel: *Good error messages* erinnert zwar daran, dass gute Fehlermeldungen ein wichtiges Kriterium für eine einfach zu bedienende Programmoberfläche sind, jedoch wird keine Auskunft darüber gegebenen, wie gute Fehlermeldungen aussehen und wie sie sich von schlechten Fehlermeldungen unterscheiden.

Insgesamt betrachtet kann man sagen, dass die in diesem Abschnitt vorgestellten Usability-Prinzipien sich zwar möglicherweise für die Klassifizierung von Usability-Problemen eignen, als Hilfe für den Entwurf wahrscheinlich jedoch eher ungeeignet sind. Ben Shneiderman [2004] erkärte, dass seine 8 goldenen Regeln des Interaktionsdesigns eher als ein hilfreicher Startpunkt für die Bewertung von Benutzerschnittstellen angesehen sollen. Gespräche mit Experten ergaben, dass Entwickler und Produkt Manager für den Entwurf solcher Benutzerschnittstellen konkretere Regeln und Hilfestellungen benötigen.

2.2 User Interface Guidelines

Leitfäden sind konkreter und ausführlicher als Prinzipien

Leitfäden zur Gestaltung von Benutzeroberflächen (engl. user interface guidelines) geben konkrete Anweisungen, wie die Benutzeroberfläche gestaltet werden soll. Sie sollen den Entwicklern dabei helfen, benutzerfreundlichere und vor allem konsistentere Programmoberflächen zu gestalten. Solche Leitfäden gibt es inzwischen für nahezu alle größeren Softwareplattformen – neben Computer- und Handy-Betriebssystemen gehören dazu auch Softwareplattformen wie z.B. Eclipse.

Sie sollen ein konsistentes Look and Feel über Anwendungsgrenzen hinweg ermöglichen

Ein konsistentes Aussehen und Verhalten von Softwareanwendungen erleichtert den Benutzern die Bedienung und vereinfacht auch das Erlernen von neuen Programmen und Funktionalitäten. Da unterschiedliche Programme beziehungsweise deren Erweiterungen aber oft von verschiedenen Entwicklerteams und Unternehmen entwickelt werden, ist eine direkte Absprache nicht möglich. Hersteller der Plattformen versuchen deshalb mithilfe von Leitfäden über die Anwendungsgrenzen hinweg ein einheitlicheres Look and Feel zu erzeugen. Die Entscheidung ob und wie sich an diese Richtlinien gehalten wird, obliegt jedoch den einzelnen Entwicklerteams und Unternehmen selbst.

2.2.1 Apple Macintosh Guidelines

Apples Leitfäden beschreiben, wie die Komponenten von Aqua verwendet werden sollen

Apple veröffentlichte bereits [1992] für die Benutzeroberfläche des Macintosh Betriebssystems ausführliche user interface guidelines. Seit seiner Veröffentlichung wird dieser regelmäßig aktualisiert und auf die Benutzeroberflächen der neuen Betriebssysteme abgestimmt. Die heutige Version umfasst knapp 400 Seiten und beschreibt das äußere Erscheinungsbild und Verhalten von Aqua, der Benutzeroberfläche von Mac OS X.

Der erste Teil des Leitfadens beschreibt, worauf es bei der Entwicklung von Benutzerschnittstellen ankommt. Er erklärt die grundlegenden Usability Prinzipien und beschreibt weshalb es wichtig ist, zu wissen, wer die Benutzer

sind und wie man diese in den Entwicklungsprozess einbeziehen kann.

Der zweite Teil beschäftigt sich mit den Technologien und Konzepten von Mac OS X. So werden hier elementare Komponenten der Benutzeroberfläche beschrieben. Dazu gehören zum Beispiel die Dock (die Schnellstartleiste für Anwendungen unter Mac OS X, die auch alle aktiven Programme beherbergt) und die Netztwerktechnologie Bonjour. Des weiteren wird beispielsweise auf Sicherheitsmerkmale von Mac OS X eingegangen und beschrieben wie Animationen das Nutzungserlebnis verbessern können.

Der dritte Teil beschreibt Aqua und gibt konkrete Tipps für den Entwurf von Benutzeroberflächen. So wird zum Beispiel beschreiben, worauf es beim Entwurf von Menüs ankommt *(siehe Abbildung 2.1)* und wie groß die Pixelabstände zwischen verschiedenen Schaltflächen sein sollen *(siehe Abbildung 2.2)*.

Für viele andere Betriebssysteme (z.B. Microsoft Windows) und Plattformen (z.B. Symbian) gibt es entsprechende Guidelines [Microsoft, 2005],[Nokia, 2005].

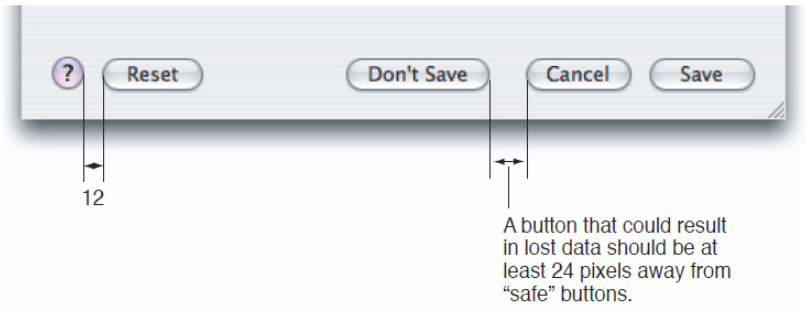


Abbildung 2.1: Auszug aus den Apple Human Interface Guidelines

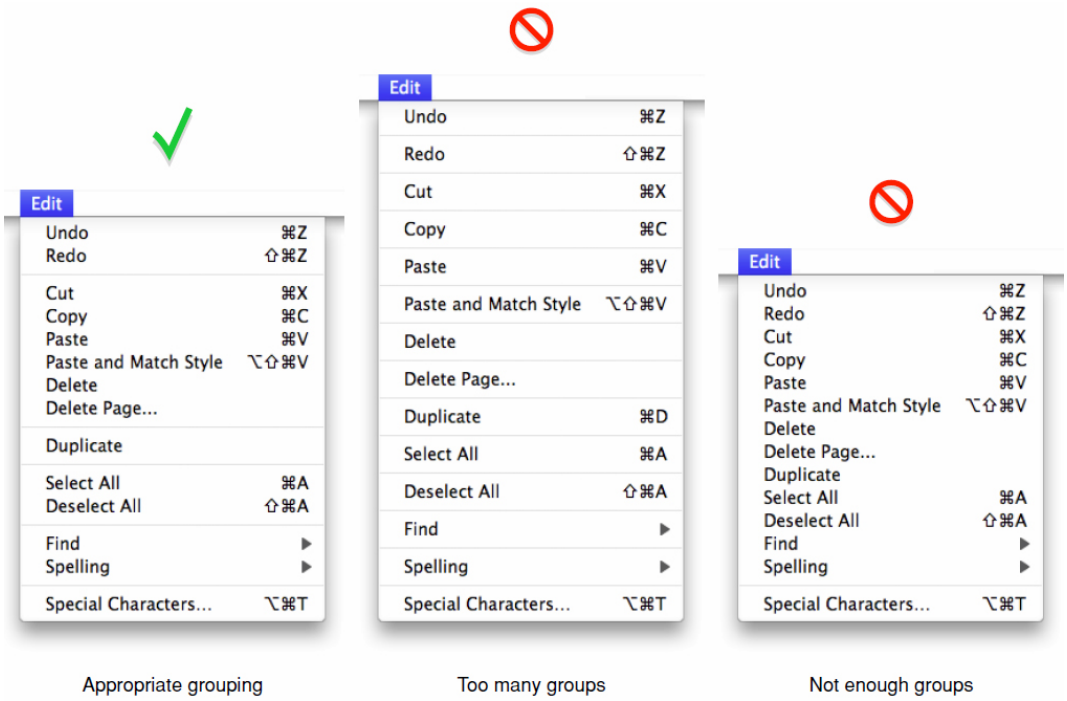


Abbildung 2.2: Auszug aus den Apple Human Interface Guidelines

2.2.2 Eclipse Guidelines

Eclipse verwandelte sich zur Plattform. Das machte Leitfäden notwendig

Neben UI guidelines für Handy- und Computer-Betriebssysteme, existieren auch einige Anwendungsbezogenen Leitfäden. Dies kommt vor allem dann vor, wenn sich die Anwendungen zu Plattformen weiterentwickeln. Eclipse ist ein gutes Beispiel für eine solche Softwareanwendung. Software-Plattformen leiden unter dem gleichen Problem wie Betriebsystem-Plattformen: Tausende Entwickler rund um den Globus arbeiten parallel und ohne Abstimmung an verschiedenen Produkterweiterungen. Um der daraus resultierenden Inkonsistenz der Benutzeroberfläche entgegen zu wirken, können Leitfäden entwickelt werden, die ein einheitliches Look and Feel vorschreiben.

Eclipse ist modular aufgebaut

Eclipse ist eine Software-Entwicklungs-Plattform mit einer offenen, plug-in basierten Struktur. Bis auf einen kleinen kernel ist alles in Eclipse ein plug-in. Primär wird Eclipse als integrierte Entwlicklungsplattform (IDE) für Java be-

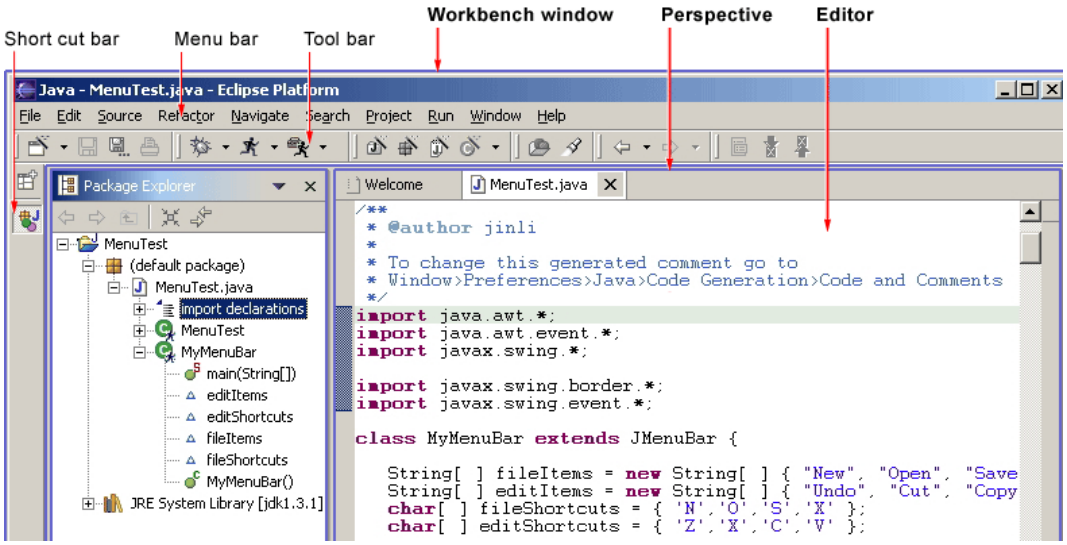


Abbildung 2.3: Auszug aus den Eclipse User Interface Guidelines

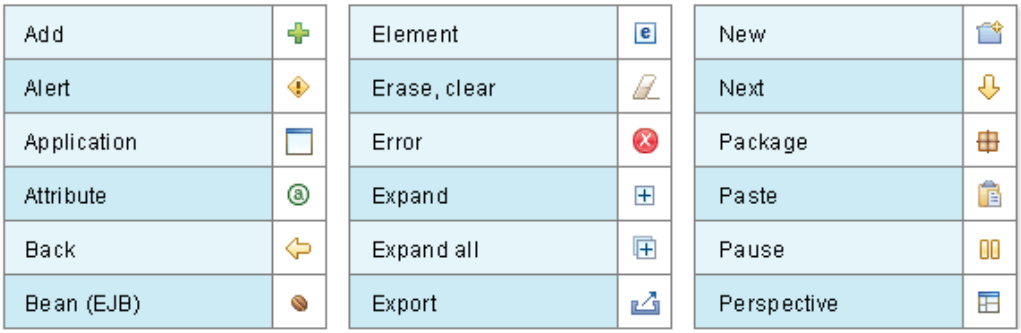


Abbildung 2.4: Auszug aus den Eclipse User Interface Guidelines

nutzt. Durch seinen modularen Aufbau und die daraus resultierende Erweiterbarkeit, kann Eclipse aber auch für die Entwicklung von Software in andere Sprachen benutzt werden. Neben den plug-ins für konkrete Programmiersprachen, gibt es aber auch solche, die andere Funktionalitäten abdecken (z.B. IDE Erweiterungen oder Hilfsmittel zur Quellcode-Analyse).

Die Eclipse User Interface Guidelines [Edgar et al., 2001] beschreiben den Aufbau der Benutzeroberfläche von Eclipse *(siehe Abbildung 2.3)* und erklären sehr ausführlich wie und wann die Standardsymbole (engl. icons) eingesetzt werden

sollen *(siehe Abbildung 2.4)*. Vorgaben zur einheitlichen Verwendung von sogenannten „shortcuts“ und Standardeinträge für Kontextmenüs in Eclipse-Texteditoren sind nur zwei weitere Beispiele aus der Vielzahl an Richtlinien, die in diesem Leitfaden festgehalten wurden.

2.3 Entwurfsmuster

2.3.1 Entwurfsmuster in der Architektur

Entwurfsmuster kommen aus dem Gebiet der Architektur

Entwurfsmuster (engl. design patterns) kommen ursprünglich aus dem Bereich der Architektur. Bereits in der Renaissance begannen Architekten ihr Wissen über gut gelungene Entwürfe niederzuschreiben. In den 70er Jahren des vorigen Jahrhunderts entwickelte der Architekt Christopher Alexander daraus die Idee der Entwurfsmuster und Mustersprachen (engl. pattern languages).

Alexanders Entwurfsmuster sind eine Sammlung von Lösungsvorschlägen zu bekannten Entwurfsproblemen aus dem Bereich der Häuser- und Städte-Architektur. Ein Muster umfasst meist nicht mehr als drei bis vier Seiten Prosatext und besteht immer aus folgenden Bestandteilen:

- **Name:** Jedes Entwurfsmuster besitzt einen eindeutigen Namen, der den Inhalt in ein bis zwei Wörtern beschreibt.
- **Bewertung:** Die Bewertung beschreibt, wie sicher sich der Autor ist, dass es sich bei der vorgestellten Lösung um die einzig wahre Lösung handelt. Hier können je nach Zuversicht entweder kein, ein oder zwei Sterne vergeben werden.
- **Bild:** Ein Bild (zumeist eine Fotografie) zeigt ein Beispiel einer Anwendung des Musters.
- **Kontext:** Der Kontext gibt an, in welchem Zusammenhang das Problem auftreten kann. Anhand des Kontexts kann der Leser des entscheiden, ob das Muster für die aktuelle Entwurfsphase relevant ist. In

diesem Abschnitt kann auf andere Muster der Mustersprache referenziert werden, die den Kontext für dieses Muster bilden (So kann zum Beispiel im Kontext eines Musters für die Gestaltung der Terrasse eines Straßencafés auf das übergeordnete Muster des Straßencafés verwiesen werden).

- **Problem:** Hier wird das Problem beschrieben, welches es zu lösen gilt. Oftmals handelt es sich hierbei um einen Zielkonflikt zwischen zwei oder mehreren Eigenschafts-Ausprägungen (z.B. Lichteinfall, Fensterfläche versus Privatsphäre).

- **Empirischer Hintergrund / Beispiele:** Hier wird das Problem (bzw. der Zielkonflikt) noch einmal näher beschrieben und es wird auf mögliche Lösungsansätze eingegangen

- **Lösung:** In diesem Absatz wird kurz und prägnant eine verallgemeinerte Lösung dargelegt.

- **Skizze:** Unter der Lösung in Textform wird die Lösungsidee noch einmal in Form einer kleinen Skizze dargestellt.

- **Referenzen:** In diesem Abschnitt wird auf weitere Muster referenziert, die in diesem Zusammenhang relevant sein könnten. Dies können beispielsweise weitere Probleme sein, die sich unmittelbar aus der beschriebenen Lösung ergeben. Das Entwurfsmuster *Straßencafé* verweist zum Beispiel auf das Entwurfsmuster *Sitzwand*, welches beschreibt, wie man die Terrasse von der Straße trennen kann.

Die Bewohner sollen in den Entwurfsprozess mit einbezogen werden

Alexanders Ziel war es, mithilfe der Entwurfsmuster die Bewohner der Häuser und Siedlungen in den Entwurfsprozess mit einzubeziehen. In seiner Arbeit „The Timeless Way of Building“ [1979] erklärt er, dass eigentlich nur die Bewohner selber entscheiden können, wie der optimale Aufbau der Gebäude aussieht, in denen sie später leben werden. Dies kommt neueren Ansätzen zum Einbezug der Benutzer in die Entwicklung von Computerprogrammen ziemlich nahe (engl. participatory design, [Schuler and Namioka, 1993]).

2.3.2 Entwurfsmuster in der Softwareentwicklung

Obwohl Alexanders Ideen viele Parallelen mit den Ideologien aus dem Bereich der Benutzerschnittstellenentwicklung ausweisen, fand der Entwurfsmuster-Ansatz seinen Weg in die Softwareentwicklung über die Konstruktion objektorientierter Software.

Entwurfsmuster vermitteln Konzepten aus der Objektorientierung

Kent Beck und Ward Cunningham [1987] berichteten auf der OOPSLA Konferenz für Objektorientierung von einem Experiment, in dem Entwickler ohne Smalltalk Erfahrung relativ erfolgreich eine Smalltalk Programmoberfläche entwickelt haben. Die Konzepte der Smalltalk Benutzerschnittstelle wurden den Probanden mithilfe von fünf Entwurfsmustern vermittelt.

Aus diesem Experiment entwickelte sich in den nachfolgenden Jahren die Idee der Software Entwurfsmuster. 1995 veröffentlichte die sogenannte „Gang of Four“ das Buch „Design Patterns – Elements of Reusable Object-Oriented Software“ [1995]. Das Buch vermittelt in Form von Entwurfsmustern Methoden der objektorientierten Softwareentwicklung und ist heute eines der Standardwerke der Softwarekonstruktion.

Diese Muster entsprechen nicht mehr der Grundidee von Alexander

Obwohl das Konzept der Entwurfsmuster auf diesem Weg Einzug in die Welt der Softwareentwicklung fand, entsprechen die von der Gang of Four vorgestellten Entwurfsmuster nicht mehr ganz der Ursprungsidee von Alexander. Zum einen unterscheiden sich die Software Entwurfsmuster in ihrem Aufbau maßgeblich von Alexanders Architektur-Mustern. Das Prinzip der Pattern Language, welches bei Alexander durch den starken Gebrauch von Referenzen aufgebaut wurde, wurde von der Gang of Four nicht übernommen. So handelt es sich hierbei mehr um eine Pattern-Sammlung als um eine Pattern Language. Zum anderen verfolgen die Entwurfsmuster aber auch ein anderes Ziel: Sie sind nicht mehr dazu bestimmt, Benutzer in den Entwicklungsprozess einzubeziehen sondern vielmehr dazu, Fachleuten neue Konzepte zu vermitteln.

2.3.3 Entwurfsmuster für Benutzerschnittstellen

Borchers brachte Entwurfsmuster in die Welt des Interaktionsdesigns

Jan Borchers führte das Prinzip der Entwurfsmuster schließlich im Bereich des Interaktionsdesigns ein. In seinem Buch „A Pattern Approach to Interaction Design“ [2001] greift er die ursprüngliche Idee von Alexander auf und stellt einen Ansatz zur Entwicklung von benutzerorientierter Software vor, welcher starken Gebrauch von Entwurfsmustern macht.

Er präsentierte ein Rahmenwerk zur integrierten Entwicklung von Software und Benutzerschnittstelle mithilfe von drei Pattern Languages *(siehe Abbildung 2.5)*:

- **Application Domain Pattern Language:** Konzepte aus dem Anwendungsbereich können unter Verwendung einer Pattern Language festgehalten werden. Dies umfasst das Wissen, welches gewöhnlich während der Initiierungsphase eines Projekts durch Befragungen und Beobachtungen der potentiellen Anwender gewonnen wird. Das gesammelte Wissen kann später von den UI Designern aufgegriffen werden, um eine Benutzerschnittstelle zu entwickeln, die den Bedürfnissen der Anwender gerecht wird.

- **Human-Computer-Interaction Pattern Language:** Die sogenannte HCI Pattern Language komprimiert Wissen über die Entwicklung von benutzerfreundlichen Benutzerschnittstellen. Dieses Wissen kann sowohl UI Designern als auch Anwendungsentwicklern zugute kommen. UI Designer können die Entwurfsmuster zur Entwicklung der Programmoberfläche nutzen. Anwendungsentwickler können durch die gewonnene Kenntnis die Softwarearchitektur so auslegen, dass sie die in den Entwurfsmustern beschriebenen Interaktionskonzepte ermöglicht.

- **Software Engineering Pattern Language:** In einer dritten Pattern Language werden Konzepte der Softwareentwicklung festgehalten. Diese Patterns richten sich an die Anwendungsentwickler und können zusammen mit den HCI Design Patterns für die Ent-

wicklung der Softwarearchitektur und des Softwaredesigns verwendet werden.

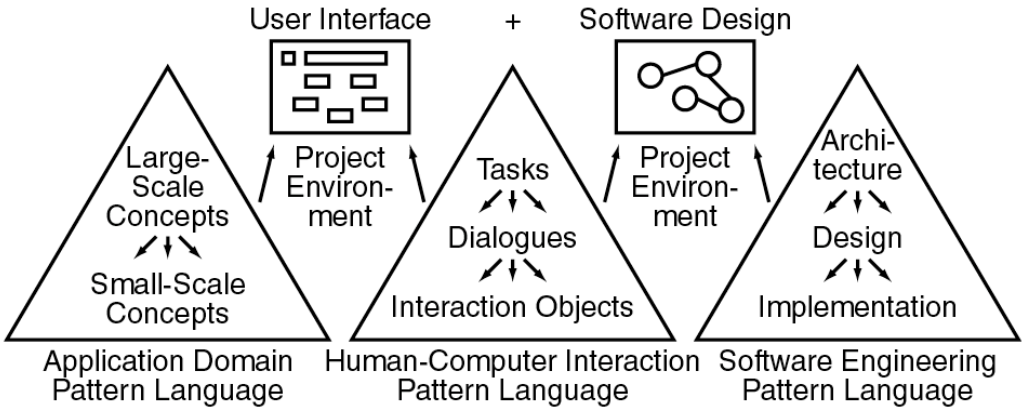


Abbildung 2.5: Der Pattern-basierte Ansatz von Jan Borchers [2001]

Borchers exerziert den beschriebenen Ansatz an dem Beispiel interaktiver Musikexponate für Museen durch. In seinem Buch präsentiert er dazu drei Pattern Languages, welche er aus seiner Erfahrung mit der Errichtung von interaktiven Exponaten entwickelt hat.

Borchers hält sich stark an Alexanders Vorgaben

Im Gegensatz zur „Gang of Four“ hat Borchers stärker an Alexanders Grundidee festgehalten. So sind seine Entwurfsmuster auch für Außenstehende verständlich. Das sollen sie auch: Borchers möchte mit seinen Entwurfsmustern eine Basis für einen Entwicklungsprozess schaffen, in dem Benutzer, UI Designer und Anwendungsentwickler Hand in Hand arbeiten. Die Entwurfsmuster der „Gang of Four“ richten sich hingegen nur an die Anwendungsentwickler und sind für Außenstehende eher unverständlich.

2.4 ISO 9241

Neben Usability Experten und Betriebssystemherstellern verfassen auch Standardisierungsorganisationen Richtlinien zur Entwicklung einfach zu bedienender Software. Die

Internationale Organisation für Normung (engl. International Organization for Standardization) – kurz ISO – hat in diesem Bereich verschiedene Standards veröffentlicht. Einer der für den Usability-Bereich am interessantesten Standards, ist der mit der Bezeichnung ISO 9241 [1998].

ISO 9241 umfasst Richtlinien für Hard- und Software

Der Standard ISO 9241 unter dem Namen „Ergonomie der Mensch-System Interaktion“ enthält eine Vielzahl an Richtlinien zur Benutzerschnittstellenentwicklung. Aktuell umfasst der Standard 28 Teile (stand Mai 2009), welche auf unterschiedliche Aspekte der Mensch-Maschine Interaktion eingehen. Einige Teile beziehen sich dabei mehr auf die ergonomische Ausrichtung der Hardware, während sich andere Teile mit der benutzerfreundlichen Gestaltung von Programmoberflächen beschäftigen.

So beschreibt ISO 9421-110 [2006] zum Beispiel die Grundsätze der Dialoggestaltung. Die ISO spezifizierte folgende sieben Grundsätze der Dialoggestaltung:

- **Aufgabenangemessenheit**
- **Selbstbeschreibungsfähigkeit**
- **Erwartungskonformität**
- **Lernförderlichkeit**
- **Steuerbarkeit**
- **Fehlertoleranz**
- **Individualisierbarkeit**

ISO Standards sind sehr technisch und abstrakt

Was genau mit diesen Prinzipien gemeint ist, wird in dem Standard ausführlich beschrieben und anhand von kleinen Textbeispielen erklärt. Trotz der Beispiele sind die Richtlinien relativ technisch und abstrakt formuliert *(siehe Abbildung 2.6)*. Dies liegt daran, dass die beschriebenen Normen auf eine möglichst breite Masse an Systemen anwendbar sein sollen.

4.3 Aufgabenangemessenheit

Ein interaktives System ist aufgabenangemessen, wenn es den Benutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h., wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Aufgabenerledigung eingesetzten Technologie.

4.3.1 Der Dialog sollte dem Benutzer solche Informationen anzeigen, die im Zusammenhang mit der erfolgreichen Erledigung der Arbeitsaufgabe stehen.

ANMERKUNG Die Aufgabenerfordernisse bestimmen die geforderte Qualität, Quantität und Art der angezeigten Informationen.

BEISPIEL 1 In einem Nutzungskontext, in dem die Verarbeitung einiger eingehender Mitteilungen zeitkritisch ist, zeigt das Dialogsystem die einzuhaltenden Termine an, die von Bedeutung sind.

Abbildung 2.6: Auszug aus der ISO 9421-110

ISO Standards sind System-unabhängig

Die ISO weist darauf hin, dass es sich bei den vorgestellten Richtlinien nicht um so genannte Styleguides (Plattformspezifische Leitfäden). ISO 9241 betrachtet weder ein spezifisches Betriebssystem noch eine spezifische Anwendungsumgebung. Die Empfehlungen sollen vielmehr bei der Entwicklung solcher Styleguides angewendet werden.

Neben dem ISO 9241 hat die Internationale Standardisierungsorganisation noch weitere Normen für die Entwicklung benutzerfreundlicher Geräte und Software veröffentlicht. So beschreibt ISO 14915 [2002] zum Beispiel Grundsätze für die Gestaltung von Multimedia-Anwendungen. ISO 11064 [2000] behandelt ergonomische Grundsätze für den Gestaltungsprozess von Leitzentralen. Leitzentralen sind Stellen, an denen eine Vielzahl von Informationen zusammen kommen und von wo aus komplexe automatisierte Systeme, Fuhrparks, oder sogar ganze Personaleinheiten gesteuert werden.

2.5 Scott Henningers GUIDE-System

Scott Henninger entwickelte eine Reihe von Systemen zur Verwaltung und Weiterentwicklung firmenspezifischer User Interface Guidelines.

Henninger argumentiert, dass sowohl neue Prozesse als auch spezialisierte Technologien benötigt werden, um User

GUIDE soll firmenspezifisches Usability Wissen verwalten

Interface Guidelines zu einem adäquaten Hilfsmittel für Entwickler zu machen. Mit seinem GUIDE-System [1997] stellt er eine Umgebung vor, die Entwicklern helfen soll, Regeln zu finden und diese im Laufe der Zeit an neue Gegebenheiten anzupassen. Dieses System soll Unternehmen dabei helfen, gesammeltes Wissen über den Entwurf von Benutzerschnittstellen zu wahren und weiterzuentwickeln.

GUIDE stellt ein Web-Interface bereit, über welches auf eine Datenbank mit spezifischen UI Guidelines zurückgegriffen werden kann. Neben der Möglichkeit, in GUIDE nach geeigneten Regeln für Entwurfsentscheidungen zu suchen, stellt Henninger eine Art Überprüfungs-Prozess vor. Die Überprüfung kann jederzeit während des Projektes durchführen und kann zu drei verschiedenen Ergebnissen führen: (1) GUIDE beinhaltet passende Regeln und der Entwurf ist zu diesen konform. (2) Die vom System vorgeschlagenen Regeln sind nicht geeignet, es existieren aber andere Regeln, die befolgt werden sollen. (3) Es existieren keine geeigneten Regeln in GUIDE.

Im ersten Fall wird der konkrete Projekt-Fall der Richtlinie zugeordnet, um so als Beispiel und Hilfsmittel für spätere Projekte zur Verfügung zu stehen. In den anderen beiden Fällen muss ein Änderungsprozess in Gang geworfen werden, welcher es erlaubt, die in GUIDE vorhanden Richtlinien zu verändern. Über die Zeit hinweg wird der Leitfaden so auf die Bedürfnisse des Unternehmens und die aktuellen Technologien angepasst.

Neben diesen Möglichkeiten zur Weiterentwicklung der Datenbasis, verfügt das GUIDE-System über einen wissensbasierten Frage-Antwort-Mechanismus, mit welchem halbautomatisch geeignete Regeln für das aktuelle Projekt selektiert werden können. Dem Entwickler werden dabei eine Reihe an Fragen über die Projekteigenschaften gestellt, nach deren Beantwortung das System die vermutlich passendste Regel vorschlägt. Das System arbeitet bei der Suche nach geeigneten Regeln ähnlich wie das amerikanische Rechtssystem (das sogenannte case law / Fallrecht): Entscheidungen werden auf Basis sogenannter Präzedenzfälle (Musterfälle aus der Vergangenheit) getroffen.

2.6 Abschließende Betrachtung

Erfahrungsberichte über den Einsatz der Leitfäden sind rar

In der Literatur existiert eine Vielzahl verschiedener Regeln und Leitfäden zur Erstellung von Benutzerschnittstellen. Die verschiedenen Regelwerke weisen dabei unterschiedliche Dataillierungsgrade auf. Trotz dieser Vielzahl an Regelwerken gibt es nur wenige Erfahrungsberichte aus der Praxis, die die Nützlichkeit solcher Prinzipien, Leitfäden und Entwurfsmuster untersuchen. Rar sind auch Berichte über die Anpassung solcher Leitfäden auf die bestimmten Bedürfnisse eines einzelnen Unternehmens.

In dieser Arbeit möchte ich untersuchen, inwiefern auf ein Unternehmen abgestimmte Entwurfsmuster den verantwortlichen Entwicklern und Produkt Managern beim Entwerfen neuer Benutzerschnittstellen behilflich sein können. Ich habe mich dabei für den Ansatz der Entwurfsmuster entschieden, weil dieses Konzept einen Mittelweg zwischen den äußerst umfangreichen Leitfäden und den sehr knappen Usability-Prinzipien einschlägt und somit vielversprechend für die Weitergabe von fachbezogenem Usability-Wissen ist.

Kapitel 3

Feldstudie

"Users are perfectly capable of expressing their latent needs. They just can't do it verbally. That's why we do ethnography and empathic reasearch."

—Rich Sheridan

3.1 Motivation

Usability Experten sind sich einig, dass die Kenntnis über die potentiellen Benutzer eine der wichtigsten Eigenschaften einer Anwender-orientierten Entwicklung ist. Der Designer muss die Bedürfnisse, Arbeitsaufgaben und den Arbeitsplatz der Benutzer kennen, um letzten Endes Anwendungen zu entwickeln, die den Benutzer bestmöglich bei seiner Arbeit unterstützen.

Richtlinien müssen auf das Anwendungsgebiet abgestimmt sein

Usability Richtlinien und Entwurfsmuster sollen die Entwickler beim Entwurf einfach zu bedienender Software unterstützen, indem sie elegante Lösungen zu häufig auftretenden Interaktions-Problemen liefern. Um jedoch heraus zu finden, welche Probleme in einem bestimmten Anwendungsgebiet häufig auftreten und wie man diese im Sinne der Benutzer lösen kann, ist auch für die Erstellung von solchen Richtlinien die Kenntnis über die Benutzer und das Anwendungsgebiet unerlässlich.

In diesem Kapitel beschreibe ich die Anwenderstudie, welche ich im Bereich der Prüfstandsautomatisierungssysteme durchgeführt habe. Im ersten Abschnitt wird auf die dazu verwendete Methodik eingegangen. Danach beschreibe ich zwei potentielle Benutzergruppen eines Prüfstandsautomatisierungssystems. Im letzten Teil des Kapitels gehe ich auf die in diesem Gebiet vorherrschenden Interaktionsparadigmen ein.

3.2 Methodik

Die Contextual Inquiry [Beyer and Holzblatt, 1998] ist eine Feldbefragungsmethode, welche im Bereich der benutzerorientierten Software-Entwicklung weit verbreitet ist. Diese Methode vereint Eigenschaften des Interviews mit denen einer Beobachtung und ermöglicht so, effektiv an Informationen über Benutzer, ihre Arbeitsaufgaben und deren Arbeitsumgebung zu gelangen.

Es findet ein Interview am Arbeitsplatz statt

Bei einer Contextual Inquiry besucht der Interviewer den Benutzer an seinem Arbeitsplatz. Ziel ist es, eine Art Meister/Lehrjunge-Beziehung zwischen Anwender und Interviewer aufzubauen. Der Benutzer soll den Interviewer in sein Arbeitsgebiet einführen. Dazu beobachtet der Interviewer den Benutzer, während dieser seiner täglichen Arbeit nachgeht. Zwischendurch stellt der Interviewer dem Benutzer Fragen um herauszufinden, warum der Benutzer dies oder jenes gemacht hat und wie sein mentales Modell des Systems aussieht.

Die Contextual Inquiry ist eine Hybrid-Methode

Wie schon einleitend angedeutet, ist die Contextual Inquiry eine Hybrid-Methode aus Interview und Beobachtung. Während Interviews in der Regel sehr durch den Interviewenden und seine Fragestellungen beeinflusst werden, ist es bei der passiven Beobachtung genau umgekehrt: hier hat der Beobachter relativ wenig Einfluss auf den Prozess. Das Idealverhältnis liegt, wie so oft, in der Mitte. Genau dort ist die Contextual Inquiry anzusiedeln. Bei dieser Methode wird ein natürlicheres Verhältnis zwischen Fragendem und Befragtem aufgebaut, was dem Interviewer hilft, möglichst realistische Daten zu gewinnen *(siehe Abbildung 3.1)*.

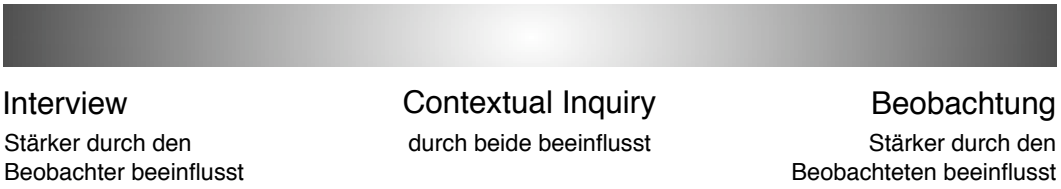


Abbildung 3.1: Die Contextual Inquiry als Hybrid-Methode [Ogle, 2009]

3.3 TCM Benutzer

Zu Beginn der Feldstudie habe ich folgende Benutzergruppen des TCM Systems identifiziert:

- **Projekt Ingenieure:** Die Projekt Ingenieure sind für den Entwurf der Testabläufe zuständig. Dazu arbeiten sie eng mit den Kunden zusammen. Die Kunden besprechen mit den Projekt Ingenieuren ihre Ziele, worauf die Projekt Ingenieure einen Durchführungsplan erarbeiten. Die Projekt Ingenieure auf den Motor abgestimmte Testablaufpläne an, welche schließlich den Prüfstandsfahrern übergeben werden, damit diese den Test durchführen.

  Projekt Ingenieure bereiten Tests vor

- **Prüfstandsfahrer:** Die zweite Benutzergruppe sind die sogenannten Prüfstandsfahrer. Die Prüfstandsfahrer bedienen und überwachen das System während des eigentlichen Testlaufs. Sie sind dafür verantwortlich, dass die geplanten Tests korrekt durchgeführt und die Messresultate gesichert werden.

  Prüfstandsfahrer führen die Tests durch

  Desweiteren müssen sie darauf achten, dass der Motor und die Test-Apparaturen während des gesamten Testlaufs in einem sicheren Bereich betrieben werden. Dazu muss der Prüfstandsfahrer ständig eine Vielzahl von Messgrößen unter Beobachtung haben. Sollte sich der Motor oder das Testsystem anders als erwartet verhalten, so muss der Prüfstandsfahrer jederzeit bereit sein, Gegenmaßnahmen einzuleiten. Im Notfall muss der Testlauf gestoppt werden.

Prüfstandsfahrer haben in der Regel ein sehr tiefgründiges Wissen über den Motor und sein Verhalten. Sie wissen genau, welche Parameter sie verändern müssen, damit sich der Motor wie gewünscht verhält. Tiefergehende Computerkenntnisse haben die meisten Prüfstandsfahrer nicht.

Applikations Ingenieure konfigurieren TCM

- **Applikations Ingenieure:** TCM ist ein sehr flexibles Automatisierungssystem. Neben der internen Nutzung wird TCM auch an Kunden verkauft. Durch die vielen verschiedenen Arten von Motorenprüfständen und die unterschiedlichen Kundenwünsche, muss TCM an die einzelnen Bedürfnisse angepasst werden. Applikations-Ingenieure besprechen mit den betreffenden Personen die Anforderungen, untersuchen diese auf Machbarkeit und entwickeln daraufhin eine maßgeschneiderte TCM Konfiguration.

Inbetriebnehmer sind für Installation und Wartung verantwortlich

- **Inbetriebnehmer:** Die Inbetriebnehmer sind für die Wartung und Inbetriebnahme der Prüfstände verantwortlich. Bevor die eigentlichen Tests starten können, müssen die Prüfstände vorbereitet werden. Dazu gehört das installieren des Motors und der Messapparaturen aber auch die Konfiguration von TCM bezüglich der neuen Ausstattung. Neben dieser Vorbereitungsarbeit bei Projektstarts werden die Inbetriebnehmer aber auch zwischendurch häufiger konsultiert. Sie verfügen über ein sehr tiefgehendes Wissen bezogen auf das TCM System und können den Prüfstandsfahrern bei akuten Problemen behilflich sein.

Bei dieser Arbeit lag der Fokus auf den beiden erst genannten Benutzergruppen. Im Rahmen einer Contextual Inquiry habe ich fünf Prüfstandsfahrer und einen Projekt Ingenieur bei ihrer Arbeit begleitet. Die Arbeit der Applikations-Ingenieure und Inbetriebnehmer wurde nicht weiter untersucht.

3.4 Prüfstandsfahrer

3.4.1 Ablauf der Contextual Inquiry

Während meiner Arbeit bei der FEV Motorentechnik habe ich an fünf Tagen drei verschiedene Prüfstände besucht. Dabei lernte ich fünf verschiedene Prüfstandsfahrer kennen, welche mir von ihrer Arbeit erzählen konnten. Drei von Ihnen habe ich ein bis zwei Tage bei Ihrer Arbeit begleitet. Zwei der Prüfstände wurden mit TCM betrieben und einer mit dem älteren Adapt-System.

Eingangs habe ich den Prüfstandsfahrern erklärt, dass ich neu in der Firma bin und gerne etwas über ihre Arbeit erfahren möchte. Desweiteren würde ich mich für das TCM System und seine Mängel interessieren. Das Erfahrene habe ich in Form handschriftlicher Notizen festgehalten. Fotografieren sowie das Anfertigen von Audio/Video-Aufnahmen sind an den Prüfständen nicht erlaubt.

3.4.2 Der Arbeitsplatz der Prüfstandsfahrer

Ein Prüfstand besteht aus einer Test-Zelle und dem Kontrollraum

Der Arbeitsplatz des Prüfstandsfahrers ist der Motoren-Prüfstand. Ein Prüfstand besteht aus einem Kontrollraum und der eigentlichen Test-Zelle *(siehe Abbildung 3.2)*. Die Test-Zelle ist ein abgeschlossener Raum, in dem sich der zu testende Motor und die Testapparaturen befinden. Der Kontrollraum ist mit der Test-Zelle durch eine Tür verbunden, welche aber während des Testlaufs aus Sicherheitsgründen verschlossen bleiben muss.

Die meiste Zeit sitzen die Prüfstandsfahrer an ihrem Kontrollpult im Kontrollraum*(siehe Abbildung 3.3)*. Durch eine große Glasscheibe können sie in die Test-Zelle hinein schauen. Je nach Prüfstand kann das Kontrollpult unterschiedliche ausgestattet sein. Zur Standardausstattung gehören zwei Bildschirme, eine Tastatur und ein separates Steuerungspult, welches zusätzliche Knöpfe und Drehregler bereitstellt. Die meisten Prüfstandsfahrer haben zusätz-

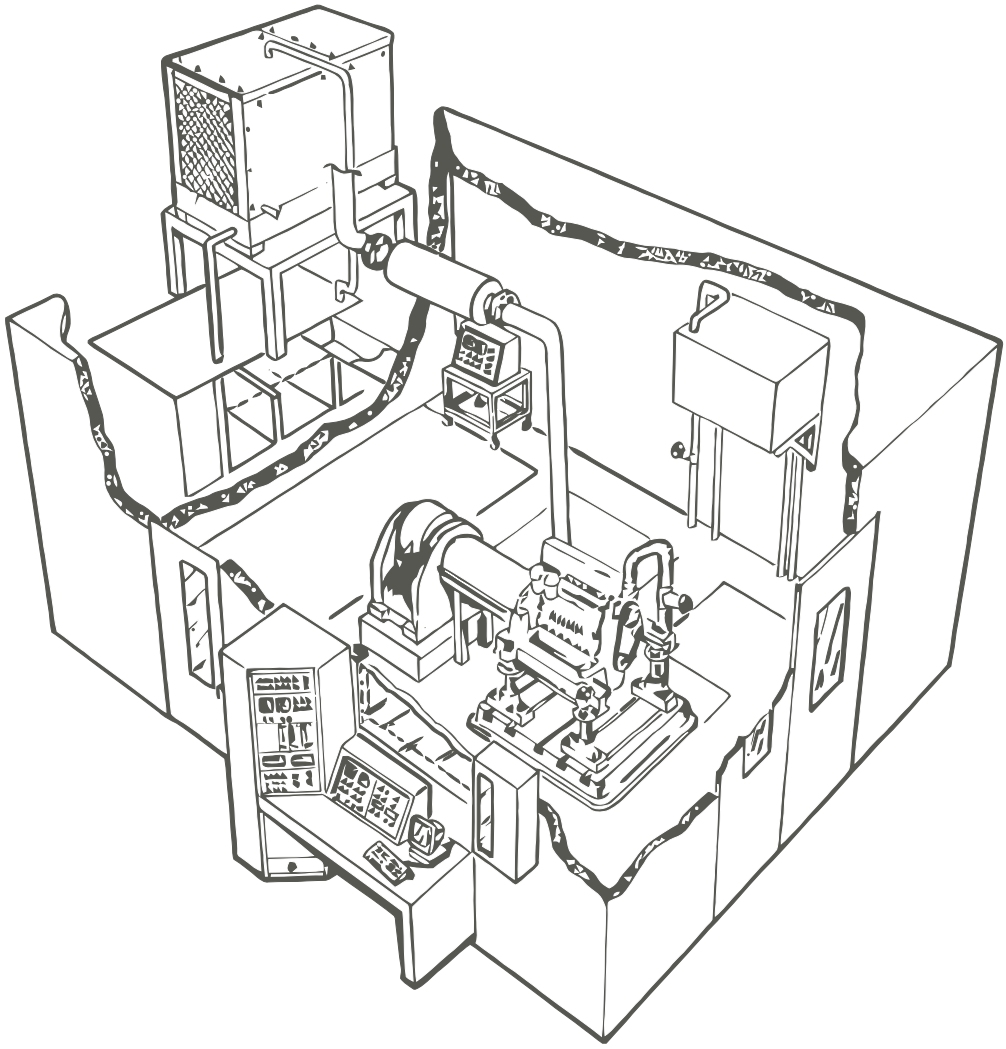


Abbildung 3.2: Eine Skizze eines Motoren-Prüfstands [Plint and Martyr, 1998]

lich noch einen Laptop auf dem Tisch stehen, auf dem sie Loglisteneinträge vornehmen und ihre e-Mails empfangen.

Im Kontrollraum ist es relativ laut

Obwohl eine dicke Wand das innere der Test-Zelle von dem Kontrollraum trennt, wird es während der Tests im Kontrollraum relativ laut – vor allem wenn der Motor bei einer hohen Drehzahl betrieben wird. Neben den Motorengeräuschen prägen Kraftstoff- und Ölgeruch das Arbeitsumfeld der Prüfstandsfahrer.

Abbildung 3.3: Der Arbeitsplatz der Prüfstandsfahrer (Foto: FEV Motorentechnik)

Im Kontrollraum sind die Prüfstandsfahrer meist nicht allein. Oft werden von einem Kontrollraum aus mehrere Prüfstände gesteuert *(siehe Abbildung 3.3)*. Neben anderen Prüfstandsfahrern befinden sich dort oft auch Inbetriebnehmer sowie Projekt Ingenieure und Kunden. Manchmal sitzen die Kunden mit am Kontrollpult. So können sie die Messergebnisse „live“ mitverfolgen und kontrollieren ob die Messungen mit den vereinbarten Randbedingungen (Umgebungstemperatur, Kraftstofftemperatur, ...) durchgeführt werden.

3.4.3 Anforderungen der Prüfstandsfahrer

Basierend auf den Beobachtungen und Interviewergebnissen habe ich eine Liste derjenigen Eigenschaften herausgearbeitet, die den Prüfstandsfahrern besonders wichtig sind:

- **Alles im Blick:** Obwohl die Hauptanzeige des Adapt und TCM-Automatisierungssystems mit seiner Vielzahl an angezeigten Mess- und Stellgrößen dem Laien möglicherweise als extrem überladen vorkommt, ist es für den Prüfstandsfahrer wichtig, alle relevanten Größen direkt im Blickfeld zu haben. Die Prüfstandsfahrer wissen genau, wo sie welche Größe auf dem Bildschirm finden und können so je nach Situation die kritischen Größen im Auge behalten.

- **Hervorheben von kritischen Größen:** Sowohl Adapt, als auch TCM ermöglichen es, die Anzeige so zu konfigurieren, dass Werte ihr visuelles Erscheinungsbild verändern, je nachdem in welchem Wertebereich sie sich aktuell befinden. So können Messgrößen, die sich in einem kritischen Bereich aufhalten besonders hervorgehoben werden (z.B. indem sie in fetter, roter Schrift dargestellt werden). Die Systeme ermöglichen die Definition mehrere solcher Wertebereiche pro Messgröße. So kann zum Beispiel zwischen drei Intervallen unterschieden werden: *(1) Temperatur zu niedrig*, *(2) Temperatur OK*, *(3) Temperatur zu hoch*.

  Von manchen Prüfstandsfahrern wird diese Funktion auch benutzt, um unkritische und weniger wichtige Messgrößen auszublenden. Der Messwert wird dann in der Hintergrundfarbe der Benutzeroberfläche dargestellt und ist somit unsichtbar für den Benutzer.

  Obwohl diese Funktionalität von allen Prüfstandsfahrern sehr geschätzt wird, erzählten drei Prüfstandsfahrer, dass sie Schwierigkeiten haben, die Wertebereiche in TCM gemäß ihren eigenen Wünschen anzupassen.

- **Konfigurierbarkeit:** Obwohl die Hauptanzeige bereits viele Messgrößen standardmäßig anzeigt, kommt es immer wieder vor, dass eine zusätzliche Mess- oder Steuergröße auf dem Bildschirm angezeigt werden soll. Dies ist zum Beispiel der Fall, wenn für einen Test neue Messgeräte angebracht wurden, deren Messwerte jetzt zusätzlich angezeigt werden sollen. Das System sollte den Prüfstandsfahrern ermöglichen, Anzeigefelder für diese neuen Messgrößen auf der Anzeige zu platzieren.

Darüber hinaus sollte das System die Möglichkeit bieten, die vorhandenen Anzeigefelder zu verschieben und neu zu gruppieren. Meine Beobachtungen haben gezeigt, dass viele Prüfstandsfahrer Schwierigkeiten damit haben und so häufig 10-20 Minuten Arbeitszeit für eine Aufgabe verwendet werden, die eigentlich in einer Minute durchführbar wäre.

- **Grafiken:** Grafische Darstellungen von Messwertverläufen über die Zeit bieten wesentlich mehr Informationsgehalt als die bloße Anzeige des aktuellen Messwertes. Mithilfe solcher Grafiken können die Prüfstandsfahrer auf einen Blick sehen, wie sich ein Wert im Verlauf der Zeit verhält. So kann festgestellt werden, ob die Messgröße sich instabil verhält und stark schwankt, ob sie konstant verläuft oder ob sie sich langsam dem gewünschten Sollwert nähert.

  Das Verhalten über die Zeit ist für die Prüfstandsfahrer ein wichtiges Indiz, welches auch Auskunft über die Qualität einer Messung gibt. So ist es meistens erwünscht, dass sich bestimmte Werte (z.B. die Drehzahl) möglichst konstant verhalten. Durch einen falsch eingestellten Regler kann es aber auch vorkommen, dass die Regelgröße anfängt zu schwingen. Grafiken geben Auskunft über ein solches Verhalten und helfen dabei, den Regler richtig einzustellen.

- **Volle Kontrolle:** Jederzeit die volle Kontrolle über den Motor und den gesamten Prüfstand zu haben ist eine der wichtigsten Anforderungen an das System. Sollte eine kritische Situation auftreten, muss der Prüfstandsfahrer innerhalb weniger Sekunden handeln. Das System muss dafür sorgen, dass seine Befehle direkt an den Motor, die Belastungsmaschine und die sonstigen Messapparaturen weitergegeben werden. Damit der Benutzer dem System vertraut, ist es auch in unkritischen Situationen wichtig, dass er jederzeit die Kontrolle über das gesamte System hat.

  Während meiner Beobachtungen trat mehrmals der Fall auf, dass gewünschte Aktionen nicht ausgeführt wurden, weil das System die entsprechenden Befehle gar nicht oder falsch weitergegeben hat. Diese Probleme traten vor allem im Zusammenhang mit den verschiedenen Modi auf, die TCM bereitstellt. Die Be-

nutzer versuchten dabei Befehle auszuführen, die im eingestellten Modus nicht funktionieren.

3.5 Projekt Ingenieure

Projekt Ingenieure sind keine direkten Benutzer des Systems

Während des Verlaufs der Feld-Analysen stellte sich heraus, dass die Projekt Ingenieure derzeit keine direkten Benutzer der Prüfstandsysteme sind. Die Entwicklung der Systeme soll allerdings derart weitergetrieben werden, dass die Projekt Ingenieure in Zukunft Testablaufpläne direkt in das Automatisierungssystem eingeben können.

Heutzutage werden die Testablaufpläne noch in getrennten Programmen erzeugt und dann als Tabelle (z.B. in Form einer CSV-Datei) an den Prüfstandsfahrer geschickt.

Im Rahmen meiner Arbeit habe ich einem Projekt Ingenieur beim Erstellen eines solchen Testablaufplans zugeschaut und Telefoninterviews mit drei weiteren Projekt Ingenieuren geführt.

3.5.1 Die Aufgaben der Projekt Ingenieure

Projekt Ingenieure erstellen Testpläne

Die Hauptaufgabe der Projekt Ingenieure ist es, die Ablaufpläne für die Tests zu erstellen. Sie entscheiden also was, wie lange und in welcher Reihenfolge getestet wird. Dazu greifen sie entweder auf existierende Standardtests zurück oder sie erstellen Tests mithilfe der sogenannten statistischen Versuchsplanung (engl. design of experiments, DOE).

Prüfstandszyklen werden von Regierungen vorgeschrieben

Aufgrund der immer stärker auftretenden Emissionsbeschränkungen seitens der Regierungen (sowohl in Europa als auch in den Vereinigten Staaten), werden die sogenannten Prüfstandszyklen immer wichtiger. Bei dieser Methode werden vorgegebene Fahrmuster auf dem Prüfstand nachgefahren und die dabei entstehenden Schadstoffemissionen hochaufgelöst aufgezeichnet. Liegen die Emissionswerte über den maximal zulässigen Werten, so muss der

Motor hinsichtlich dieser Kriterien optimiert werden. Das Optimieren geschieht meistens durch verändern von Parametern im Motorsteuergerät (engl. Engine Control Unit, ECU).

Für die Fahrmuster existieren Tabellen, welche die entsprechenden Testabläufe in einem für TCM verständlichen Format beinhalten (das sogenannte FIH-Dateiformat). Manchmal entscheidet sich der Projekt Ingenieur dafür, nicht direkt das gesamte Fahrmuster abzufahren sondern nur kritische Teilstücke zu untersuchen und den Motor daraufhin zu optimieren. Erst danach wird das gesamte Fahrmuster getestet.

Aufwändiger in der Vorbereitung ist die statistische Versuchsplanung. Oft geht es bei den Tests darum, den Motor hinsichtlich gewisser Zielparameter zu optimieren. Minimierung der Emissionswerte, Minimierung des Kraftstoffverbrauchs und Maximierung des Drehmoments sind nur einige Beispiele aus der Vielzahl an möglichen Optimierungsdimensionen eines Verbrennungsmotors. Diese Zielparameter werden von vielen verschiedenen Größen beeinflusst (z.B. Einspritzmenge/-dauer/-zeitpunkt, Abgasrückführungsrate, Luft-Kraftstoff-Verhältnis, ...).

Beim DOE wird ein Modell des Motorverhaltens erstellt

Um das tatsächliche Optimum bezüglich eines Zielparameters zu finden, müsste man pro Drehzahlstufe alle Kombinationen der berücksichtigten Einflussgrößen vermessen. Eine statistische Versuchsplanung ermöglicht es jedoch, den Testaufwand drastisch zu reduzieren. Bei diesem Testverfahren müssen nur wenige Kombinationen der Einflussgrößen (= Testpunkte) am Prüfstand getestet werden. Aus den gewonnenen Messdaten kann im Nachhinein ein Modell des Motorverhaltens erstellt werden, welches es ermöglicht, die notwendigen Optimierungsschritte öfflineäm PC vorzunehmen.

Die Erstellung solcher Versuchspläne (sprich: die Suche nach geeigneten Testpunkten) ist relativ komplex und auch Aufgabe der Projekt Ingenieure. Die Liste mit den Testpunkten wird auch hier wieder in Form einer Tabelle dem Prüfstandsfahrer weitergegeben, welcher dann die einzelnen Testpunkte auf dem Prüfstand anfährt und misst.

3.5.2 Anforderungen der Projekt Ingenieure

In Zukunft sollen die Projekt Ingenieure ihre Versuchspläne direkt in das Prüfstandssystem einpflegen. Daraus resultieren folgende Anforderungen an das System:

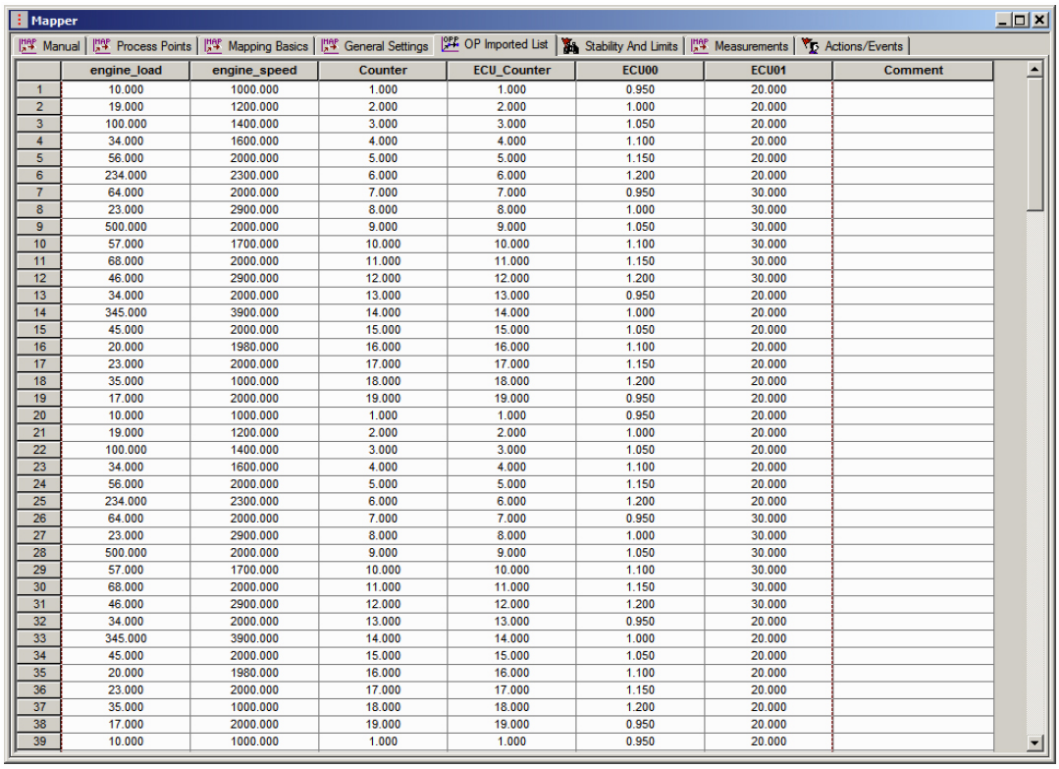
- **Einfaches Erstellen von Tests:** Hauptkriterium für die Projekt Ingenieure ist es, Testabläufe möglichst einfach und effizient erstellen und bearbeiten zu können. Heute werden die Testabläufe meist in Excel-Tabellen vorgefertigt und dann unter Verwendung bestimmter Hilfsprogramme in das TCM FIH-Dateiformat transformiert. Die dadurch entstehenden Dateien sind sehr unübersichtlich und dadurch für die Benutzer schwer zu durchschauen (Der 30-minütige European Transient Cycle (ETC) beansprucht zum Beispiel 3602 Zeilen in der FIH-Datei). Dies kann zu Komplikationen führen, wenn kurzfristig kleine Änderungen in den Testabläufen vorgenommen werden müssen.
- **Integration in die Toollandschaft:** Bei der Erstellung von Versuchsplänen im Rahmen der statistischen Versuchsplanung, greifen die Projekt Ingenieure auf Standardanwendungen wie z.B. Matlab und Excel zurück. Die von diesen Programmen generierten Ablaufpläne müssen auch hier wieder anhand von Hilfsprogrammen in das TCM-Format übersetzt werden. Eine Integration von TCM in die vorhandene Toollandschaft könnte diesen Prozess vereinfachen.

3.6 TCM Interaktionsparadigmen

Bei der Erstellung von UI Richtlinien und Entwurfsmustern müssen neben den Benutzeranforderungen auch die vertrauten Interaktionsprinzipien aus der vorhandenen Softwarelandschaft berücksichtigt werden. Ziel ist es nicht, die vorhandenen Interaktionsparadigmen über Bord zu werfen, sondern auf deren Basis neue Design-Ideen einzubringen, welche die Bedienung des Systems vereinfachen.

Auf einige Hauptmerkmale der TCM Benutzerschnittstelle möchte ich etwas näher eingehen:

- **Listen und Tabellen:** Listen und Tabellen dominieren die Benutzeroberfläche von TCM und sind Hauptdarstellungsform für Daten. Kanäle, Events, Loglisten und Grenzwerte sind nur einige Beispiele der Daten, welche in TCM in Tabellenform dargestellt werden. Meistens beinhalten diese Tabellen eine Vielzahl an Zeilen, was die Suche innerhalb dieser Tabellen erschwert *(siehe Abbildung 3.4)*.



	engine_load	engine_speed	Counter	ECU_Counter	ECU00	ECU01	Comment
1	10.000	1000.000	1.000	1.000	0.950	20.000	
2	19.000	1200.000	2.000	2.000	1.000	20.000	
3	100.000	1400.000	3.000	3.000	1.050	20.000	
4	34.000	1600.000	4.000	4.000	1.100	20.000	
5	56.000	2000.000	5.000	5.000	1.150	20.000	
6	234.000	2300.000	6.000	6.000	1.200	20.000	
7	64.000	2000.000	7.000	7.000	0.950	30.000	
8	23.000	2900.000	8.000	8.000	1.000	30.000	
9	500.000	2000.000	9.000	9.000	1.050	30.000	
10	57.000	1700.000	10.000	10.000	1.100	30.000	
11	68.000	2000.000	11.000	11.000	1.150	30.000	
12	46.000	2900.000	12.000	12.000	1.200	30.000	
13	34.000	2000.000	13.000	13.000	0.950	20.000	
14	345.000	3900.000	14.000	14.000	1.000	20.000	
15	45.000	2000.000	15.000	15.000	1.050	20.000	
16	20.000	1980.000	16.000	16.000	1.100	20.000	
17	23.000	2000.000	17.000	17.000	1.150	20.000	
18	35.000	1000.000	18.000	18.000	1.200	20.000	
19	17.000	2000.000	19.000	19.000	0.950	20.000	
20	10.000	1000.000	1.000	1.000	0.950	20.000	
21	19.000	1200.000	2.000	2.000	1.000	20.000	
22	100.000	1400.000	3.000	3.000	1.050	20.000	
23	34.000	1600.000	4.000	4.000	1.100	20.000	
24	56.000	2000.000	5.000	5.000	1.150	20.000	
25	234.000	2300.000	6.000	6.000	1.200	20.000	
26	64.000	2000.000	7.000	7.000	0.950	30.000	
27	23.000	2900.000	8.000	8.000	1.000	30.000	
28	500.000	2000.000	9.000	9.000	1.050	30.000	
29	57.000	1700.000	10.000	10.000	1.100	30.000	
30	68.000	2000.000	11.000	11.000	1.150	30.000	
31	46.000	2900.000	12.000	12.000	1.200	30.000	
32	34.000	2000.000	13.000	13.000	0.950	20.000	
33	345.000	3900.000	14.000	14.000	1.000	20.000	
34	45.000	2000.000	15.000	15.000	1.050	20.000	
35	20.000	1980.000	16.000	16.000	1.100	20.000	
36	23.000	2000.000	17.000	17.000	1.150	20.000	
37	35.000	1000.000	18.000	18.000	1.200	20.000	
38	17.000	2000.000	19.000	19.000	0.950	20.000	
39	10.000	1000.000	1.000	1.000	0.950	20.000	

Abbildung 3.4: Beispiel einer Tabelle in TCM

- **Baumstruktur:** TCM basiert auf einer Modulstruktur. Verschiedenen System-Funktionalitäten sind unterschiedlichen Modulen zugeordnet. Jedes Modul besitzt eine klar definierte Aufgabe innerhalb der TCM Systemlandschaft. Die Module sind mit dem sogenannten TCM Framework verbunden, welches eine Reihe an Standardfunktionalitäten bereitstellt. Jedes

Modul besitzt seine eigene, an die Aufgabe angepasste, Benutzerschnittstelle.

Die Module werden in TCM am linken Bildschirmrand als Baumstruktur dargstellt. In dieser Baumstruktur sind jedem Modul Kanäle, Events und sonstige Modul-spezifische Funktionalitäten untergeordnet *(siehe Abbildung 3.5)*. Der Modul-Baum ist Hauptnavigationselement in TCM. Der Benutzer kann Module oder Teilfunktionen der Module mit der Maus auswählen, woraufhin sich in der rechten Hälfte der Programmoberfläche die entsprechende Benutzeroberfläche öffnet.

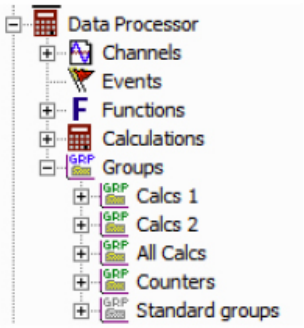


Abbildung 3.5: Die Baumstruktur des Data Processsor-Moduls

- **User Defined Screens:** Die sogenannten User Defined Screens (UDS) *(siehe Abbildung 3.6)* ermöglichen dem Benutzer, angepasste Benutzeroberflächen zu erstellen. UDS bilden den Teil der Benutzerschnittstelle, welcher von den Prüfstandsfahrern während der Tests verwendet wird. Während des Betriebs kann zwischen mehreren UDS hin- und hergeschaltet werden.

  Die User Defined Screens beinhalten Anzeigen für die Messwerte sowie Steuerelemente für den Motor und die Messapparaturen. Die einzelnen Bedienelemente können auf den UDS frei platziert werden. TCM bietet von Haus aus eine Vielzahl an Bedien- und Anzeigeelementen (z.B. Schieberegler, Knöpfe, Thermometer-Anzeigen, Graphen, ...) *(siehe Abbildung 3.6)*.

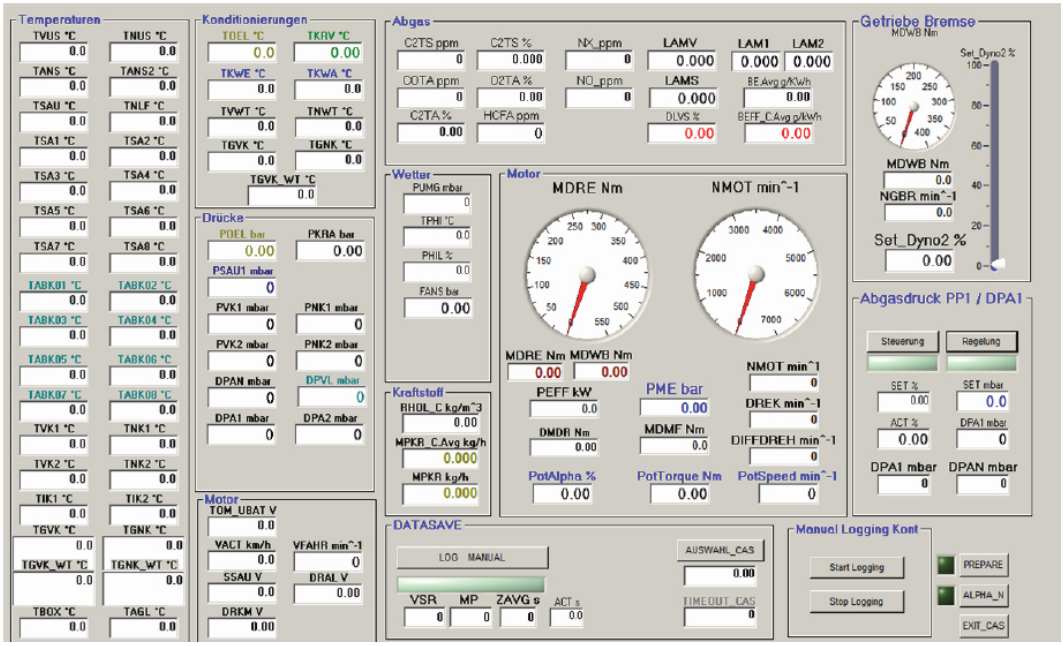


Abbildung 3.6: Beispiel eines User Defined Screens

Kapitel 4

Auswahl der Muster

"Adequate information about the existing environment and about the types of place that it is desirable to make cannot be kept inside one brain."

—Christopher Alexander

Nachdem ich mithilfe der Contextual Inquiry das Anwendungsgebiet kennengelernt habe und die Bedürfnisse der einzelnen Benutzergruppen sowie die vorherrschenden Interaktionsprinzipien der vorhandenen Software analysieren konnte, werde ich nun für das Anwendungsgebiet geeignete HCI Design Patterns suchen. Ziel ist es, solche Entwurfsmuster zu finden, die den Produkt Managern und Entwicklern beim Entwerfen der Benutzerschnittstellen behilflich sein können. Es sollen also Patterns gefunden werden, durch deren Anwendung Schwachstellen in den Programmoberflächen des TCM-Systems ausgemerzt werden können.

Im Rahmen einer Literaturrecherche habe ich verschiedene Pattern Sammlungen durchforstet. Insgesamt wurden über 600 Entwurfsmuster verschiedener Autoren hinsichtlich ihrer Relevanz für das TCM-Prüfstandsautomatisierungssystem untersucht. Da viele Entwurfsmuster in mehreren Sammlungen vorkommen, habe ich mich vor allem auf die Sammlung von Jenifer Tidwell [2005] konzentriert.

Die folgende Tabelle beschreibt kurz die unterschiedlichen Sammlungen, welche ich bei meiner Auswahl berücksichtigt habe. Im darauffolgenden Abschnitt wird jedes der 18 ausgewählten Entwurfsmuster *(siehe Anhang A)* kurz beschrieben und erläutert, warum es relevant für TCM und das Gebiet der Prüfstandsautomatisierungssysteme ist.

Autor / Name der Sammlung	Anzahl der Patterns	Kurze Beschreibung
Jenifer Tidwell Designing Interfaces	94	Umfangreiche Sammlung allgemeiner Interaktionsprinzipien grafischer Benutzeroberflächen. Sowohl auf Desktop- als auch auf Webanwendungen bezogen.
Martijn van Welie A Pattern Library for Interaction Design	ca. 140	Umfangreiche Sammlung für den Entwurf von Websites. Welie unterteilt seine Sammlung in Entwurfsmuster für Benutzer-Bedürfnisse, Anwendungs-Bedürfnisse und den Kontext.
Douglas K. van Duyne The Design of Sites	ca. 110	Standardwerk für den Entwurf von Websites.
Diverse Autoren UI-Patterns	47	UI-Patterns ist eine offene Plattform für HCI Design Patterns. Benutzer können neue Patterns erstellen und vorhandene kommentieren. Zur Zeit ist diese Sammlung noch eher spärlich.
The Usability Group Brighton The Usability Pattern Collection	10	Eine der ältesten HCI Design Pattern Sammlungen. Nicht sehr umfangreich - zuletzt aktualisiert im Jahr 1998.
Janne Lammi The UI Pattern Factory	18	Kleine, kommentierbare, Web-basierte Pattern Sammlung für Ajax/Web 2.0 Interaktionskonzepte. Teilweise auch auf Desktop-Anwendungen übertragbar.
Yahoo Inc. Design Pattern Collection	47	Yahoo ist eines der wenigen Unternehmen, die eine eigene HCI Design Pattern Sammlung entwickelt und veröffentlicht haben. Die Patterns beziehen sich auf den Entwurf von Websites.
Ian Graham Wu	79	Wu ist eine Pattern Sammlung, die Interaktionsprinzipien für Webanwendungen beinhaltet. Die Entwurfsmuster sind in einem ähnlichen Format geschrieben, wie die von C. Alexander.
Mark L. Irons Patterns for Personal Websites	36	Diese Entwurfsmuster sollen angehenden Webdesignern beim Entwurf ihrer persönlichen Website helfen. Auch hier wird ein Alexander-ähnliches Format verwendet.
Saari A. Laasko User Interface Design Pattern	21	Die vorgestellten Entwurfsmuster sind in erster Linie keine Patterns, wie sie in heutiger Software vorkommen, sondern beschreiben das, was die Autoren unter einem guten Design verstehen.
Dep. of Georgraphy, Uni Oregon Patterns for Data Graphics	5	Fünf Patterns für die Visualisierung von Daten. Die Entwurfsmuster kommen aus dem Bereich der Geographie, sind aber auch in anderen Gebieten einsetzbar.
Todd Coram and Jim Lee Experiences	9	Experiences ist eine der wenigen HCI Design Pattern Languages. Wie bei Alexander und Borchers sind die Entwurfsmuster hierarchich angeordnet. Leider sind nur neun der 26 Patterns ausformuliert
Diverse Autoren Quince	90	Von Infragistics entwickelter Pattern Explorer. Benutzer können Patterns kommentieren und über ihren Einsatz in der Praxis berichten. Keine neuen Patterns - Hauptsächlich Tidwell-Patterns in der Bibliothek

Tabelle 4.1: Liste der betrachteten Pattern-Sammlungen

Die Relevanz der Entwurfsmuster wurde in einem mehrstufigen Auswahlverfahren bestimmt. Den einzelnen Patterns wurde dabei eine Zahl auf der Skala zwischen 1 und 5 zugewiesen, die die Relevanz bezüglich des Anwendungsgebiets bewertete. Ähnliche Muster wurden nicht doppelt berücksichtigt sondern zu einem zusammengefasst. Zwischendurch wurde die Auswahl mit einem Usability-Experten aus dem Fachbereich diskutiert.

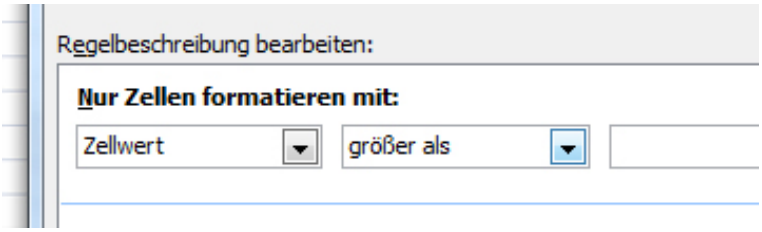
4.1 Edit in Place

Textänderungen direkt an Ort und Stelle vornehmen

Das erste Entwurfsmuster stammt aus der Sammlung von Jenifer Tidwell [2005] und beschreibt ein Konzept zum Editieren von Text-Elementen. Das Pattern beschreibt, dass es in vielen Fällen benutzerfreundlicher ist, einen Text direkt an der Stelle bearbeiten zu können, an welcher er auf der Programmoberfläche auftaucht. Anwendung findet dieses Entwurfsmuster unter anderem in Dateiverwaltungsprogrammen von modernen Betriebssystemen, wie zum Beispiel im Explorer von Microsoft Windows: Beim Klick auf einen Datei- oder Ordnernamen öffnet sich ein kleiner Texteditor, welcher das Ändern des Namens ermöglicht.

Auf der TCM-Programmoberfläche erscheinen viele verschiedene Text-Elemente (z.B. Kanal-Namen, Grenzwerte, Event-Namen, ...). Oft sind diese aber nur in einem zusätzlichen Dialog editierbar. Der Dialog lässt sich meistens durch ein Kontextmenü öffnen. Durch das in dem Entwurfsmuster beschriebene Konzept, könnte man die Bearbeitung solcher Text-Elemente vereinfachen.

4.2 Fill in the Blanks

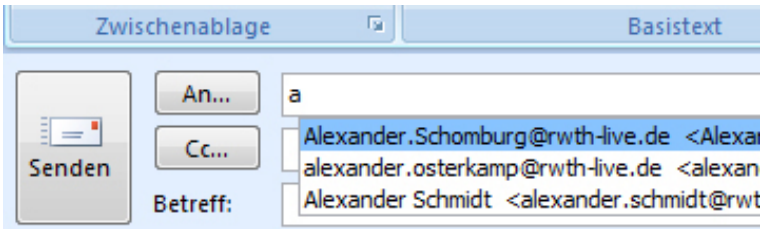


Dialoge sollen das Ausdrücken, was das System macht

Benutzereinegaben werden meistens mithilfe von Kontrollelementen (z.B. Textfelder, Kontrollhäkchen,...) abgefragt. Jedes dieser Kontrollelemente besitzt in der Regel eine Bezeichnung, die angibt, welche Eingabe erwartet wird (z.B. „Name“ oder „Adresse“). Diese deklarative Art der Benutzereingabe ist in manchen Fällen aber nicht klar genug. „Fill in the Blanks“ Tidwell [2005] schlägt vor, eine Art Lückentext mit den Kontrollelementen zu erstellen. Der Text oder Satz soll – wenn er ausgefüllt ist - das ausdrücken, was das System aufgrund der Eingabe macht.

TCM enthält viele Konfigurationsfenster, die eine Fülle an Kontrollelementen beherbergen. Oft ist der Zusammenhang der Kontrollelemente, sowie die Folge der Eingabe für den Benutzer nicht direkt ersichtlich. „Fill in the Blanks“ kann dabei helfen, die Konfiguration verständlicher zu gestalten.

4.3 Autocompletion



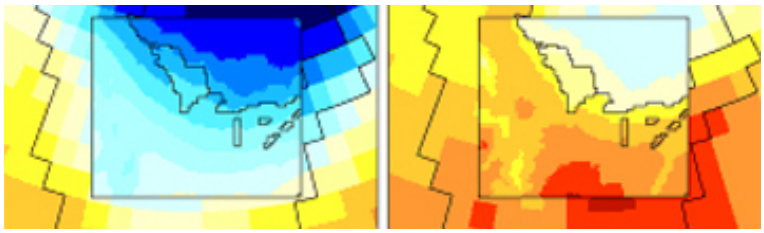
„Autocompletion“ Tidwell [2005] beschreibt ein Konzept, welches dem Benutzer bei der Eingabe von schwierig zu merkenden Texten hilft. Dieses Entwurfsmuster kann

immer dann angewendet werden, wenn eine Texteingabe erforderlich ist, bei der das System eine Liste an möglichen Eingaben antizipieren kann. Diese Liste kann zum Beispiel auf vorherigen Eingaben basieren. Webbrowser benutzen dieses Konzept, um dem Benutzer die Eingabe der schwer zu merkenden Web-Adressen zu erleichtern.

Eingabemöglichkeiten sollen vom System vorgeschlagen werden

In TCM müssen an vielen Stellen Eingaben von schwer merkbaren Textstücken getätigt werden. Ein Beispiel ist die Eingabe von kryptischen Kanalnamen. Da das System die Namen aller verfügbaren Kanäle kennt, könnte hier Autocompletion zum Einsatz kommen. Sobald der Benutzer anfängt einen Kanalnamen einzugeben, kann das System in einer Liste unter dem Eingabefeld alle zulässigen Kanalnamen anzeigen, die die bereits eingegebene Buchstabenfolge beinhalten.

4.4 Resolution Indicates Data Quality



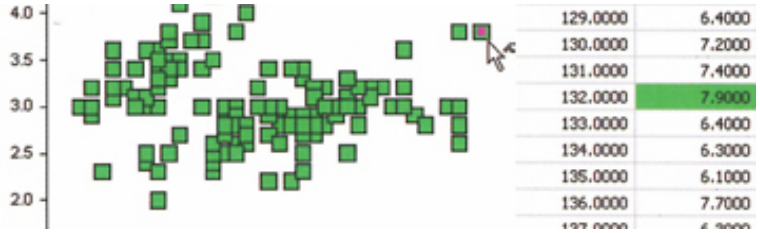
Das Entwurfsmuster „Resolution Indicatetes Data Quality“ University of Oregon [2007] kommt ursprünglich aus dem Bereich der Meteorologie und beschreibt, dass die Auflösung der Messdaten für den Benutzer ein wichtiges Merkmal zum Bestimmen der Messqualität darstellt. Aus ästhetischen Gründen interpolieren viele Softwareanwendungen die Messergebnisse, bevor sie grafisch dargestellt werden. Dadurch ist für den Benutzer aber nicht mehr ersichtlich, welche Genauigkeit die Daten haben. Das Entwurfsmuster empfiehlt, Messdaten nicht zu interpolieren, sondern in der Auflösung der tatsächlichen Messung darzustellen.

Auflösung ist ein wichtiges Indiz für die Messqualität

Obwohl das Muster ursprünglich aus einem grundverschiedenen Anwendungsgebiet kommt, passt es doch recht

gut zum Gebiet der Prüfstandsautomatisierungssysteme. TCM beinhaltet an vielen Stellen der Benutzeroberfläche grafische Repräsentationen von Messdaten – überall dort kann das Entwurfsmuster angewendet werden.

4.5 Data Brushing

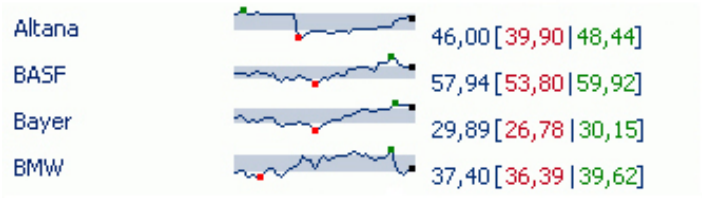


Der Zusammenhang zwischen Daten in verschiedenen Ansichten soll sichtbar sein

„Data Brushing“ Tidwell [2005] kann überall dort angewendet werden, wo Daten in verschiedenen Ansichten visualisiert werden. Das Entwurfsmuster beschreibt ein Konzept, bei dem der Benutzer Datenpunkte in einer Grafik auswählt, worauf die selektierten Datenpunkte in allen zugehörenden Ansichten hervorgehoben werden. Beispielsweise können Daten gleichzeitig in einer Tabelle und einer Grafik visualisiert werden. Beim Selektieren eines Datenpunktes in der Grafik wird die entsprechende Zelle in der Tabelle hervorgehoben – dort kann der Benutzer den genauen Wert des Punktes ablesen.

Auch bei Prüfstandssystemen macht es Sinn, verschiedene Sichtweisen auf Daten bereitzustellen. Das in diesem Entwurfsmuster vorgeschlagene Konzept kann dabei helfen, den Nutzen dieser Grafiken und Tabellen noch zu erweitern, indem dem Benutzer ersichtlich gemacht wird, welche Daten die gleichen Messpunkte darstellen.

4.6 Sparklines



Grafiken beinhalten wichtige Informationen über den Verlauf eines Wertes

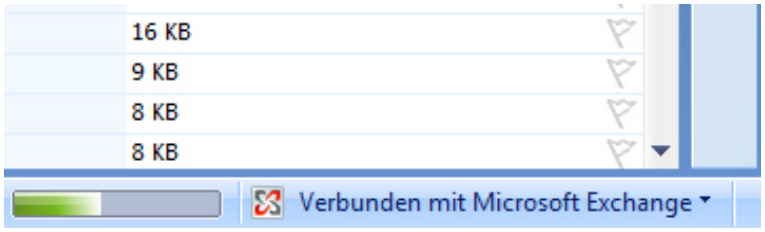
Sparklines ist eine von Edward Tufte [2006] entwickelte Idee, herkömmliche 2-dimensionale Graphen auf ihr Wesentlichstes zu reduzieren. Graphen beinhalten wichtige Informationen bezüglich der historischen Entwicklung eines Wertes. So gibt ein Graph unter anderem Auskunft darüber, wie regelmäßig oder unregelmäßig ein Wert verläuft, wie seine Tendenz ist und wo die Maxima bzw. Minima liegen. Der Nachteil von Graphen ist jedoch, dass sie viel Platz auf der Programmoberfläche in Anspruch nehmen.

Sparklines beschränken sich auf das Wesentliche

Sparklines verzichten auf grafische Dekorationen, Legenden und Achsen und stellen lediglich die Veränderung eines Wertes über die Zeit dar. Genaue Werte aus der Vergangenheit lassen sich mit dieser Darstellungsform zwar nicht mehr ermitteln aber die kleine Grafik genügt, um Aussagen über die Tendenz und das Schwingungsverhalten eines Wertes zu treffen.

Sparklines ist eines der Entwurfsmuster, in denen ich am meisten Potential für das TCM-System sehe. Der TCM Hauptbildschirm zeigt den Prüfstandsfahrern eine Fülle an Momentan-Werten an. Diese Werte durch Sparklines zu ersetzen würde nur unwesentlich mehr Platz in Anspruch nehmen, den Prüfstandsfahrern aber wichtige zusätzliche Informationen bieten.

4.7 Status Panel

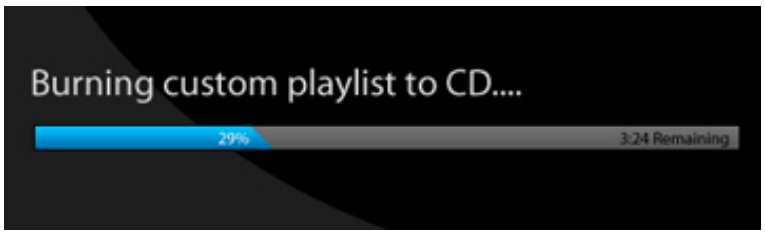


Manche Informationen müssen permanent sichtbar sein

Rückmeldung auf Benutzereingaben oder Auskünfte über den Systemsstatus sind wichtige Informationen, die eine Benutzerschnittstelle kommunizieren muss. Dieses Entwurfsmuster beschreibt das Konzept der Statusleiste, welches in vielen Software-Anwendungen verwendet wird. Manche Informationen sollen zwar jederzeit sichtbar sein, dürfen aber den Benutzer nicht beim Erledigen seiner aktuellen Aufgabe stören. Die sogenannte Statusleiste stellt einen Bereich dar, welcher für die permanente Anzeige solcher Informationen reserviert ist Tidwell [2005].

TCM muss den Benutzer darüber informieren, in welchem Betriebsmodus sich das System befindet. Des Weiteren muss für den Benutzer jederzeit ersichtlich sein, ob sich das System in einem sicheren Zustand befindet. Diese Informationen können anhand einer solchen Statusleiste kommuniziert werden, ohne den Benutzer durch aufspringende Hinweisfenster in seinem Arbeitsfluss zu unterbrechen.

4.8 Progress Indicator

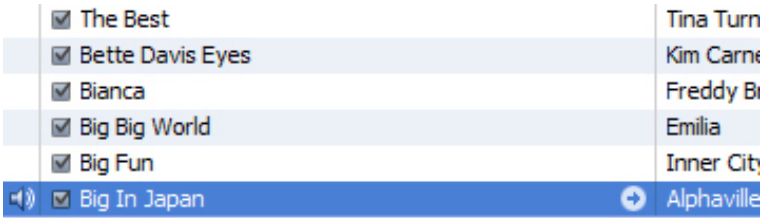


Auf länger dauernde Operationen muss hingewiesen werden

Das Entwurfsmuster mit den Namen „Progress Indicator“ Tidwell [2005] beschreibt das Konzept der sogenannten Fortschrittsanzeige. Oft kommt es vor, dass ein Computer zum Erledigen einer Operation mehr als nur ein paar Bruchteile einer Sekunde benötigt. In diesen Fällen muss der Benutzer darauf hingewiesen werden, dass das System sein Kommando erhalten hat und noch mit der Ausführung der Operation beschäftigt ist. Das Entwurfsmuster beschreibt des Weiteren, dass es besser ist, keine Informationen über Fortschritt und Restzeit zu geben, als unzuverlässige Schätzungen abzugeben.

Auch in Prüfstandsautomatisierungssystemen nehmen manche Operationen mehr Zeit in Anspruch als andere. Der Benutzer soll mithilfe von Fortschrittsanzeigen darauf hingewiesen werden. In TCM kommt es manchmal vor, dass die Benutzerschnittstelle einfriert und erst nach wenigen Sekunden weiter arbeitet. Anhand einer Fortschrittsanzeige kann dem Benutzer signalisiert werden, dass das System nicht abgestürzt sondern beschäftigt ist.

4.9 Row Striping



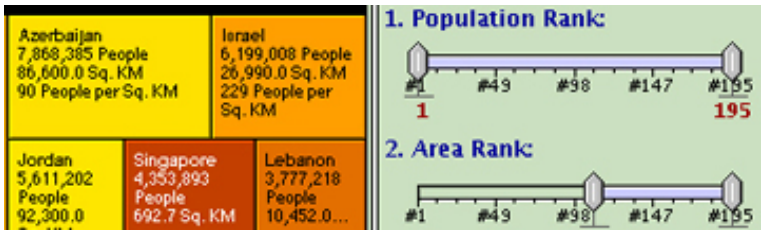
Tabellenzeilen sind oft schwer zu unterscheiden

Große Tabellen zu durchsuchen ist oftmals keine einfache Aufgabe. Erschwert wird die Sache dadurch, dass es – bedingt durch die dargestellte Datenmasse – häufig schwer ist, den einzelnen Tabellenzeilen zu folgen. Manchmal kommt es vor, dass man beim Lesen einer Zeile aus versehen in die darüber oder darunter liegende Zeile rutscht. Größere Abstände zwischen den Zeilen einzufügen ist oftmals aus Platzgründen nicht möglich. Auch Trennlinien lösen das Problem nicht – im Gegenteil: sie sind zusätzliche visuelle Stördaten, welche von den eigentlichen Daten ablenken.

„Row Striping“ Tidwell [2005] ist ein Gestaltungsprinzip, bei dem zwei aufeinanderfolgende Zeilen eine unterschiedliche Hintergrundfarbe haben. Dadurch sinkt die Gefahr, diese Zeilen miteinander zu verwechseln, beziehungsweise beim Lesen von der einen in die andere Zeile zu rutschen.

Wie schon im vorigen Kapitel erwähnt, sind Tabellen in TCM allgegenwärtig. Von Prüfstandsfahrern weiß ich, dass es oft mühselig ist, die gesuchten Einträge in diesen Tabellen zu finden. Das vorgestellte Konzept kann dazu beitragen, die Lesbarkeit und Benutzbarkeit solcher Tabellen zu verbessern.

4.10 Dynamic Queries



In Datensammlungen interessieren oft nur Elemente mit bestimmten Eigenschaften

„Dynamic Queries“ ermöglicht den Benutzern eine Datensammlung entsprechend bestimmter Kriterien zu durchsuchen. Das Entwurfsmuster schlägt vor, Steuerelemente wie Schieberegler und Kontrollkästchen bereitzustellen, mit denen der Benutzer Filterkriterien für eine Datensammlung anpassen kann. Beim Verstellen dieser Steuerelemente soll sich die Anzeige direkt entsprechend der neuen Filterkriterien ändern.

Oftmals interessieren den Benutzer eines Prüfstandsautomatisierungssystems nicht alle Messdaten sondern nur solche, die gewisse Eigenschaften aufweisen. Denkbar ist zum Beispiel ein Szenario, in dem sich der Prüfstandsfahrer für alle Messungen interessiert, die bei einer mindest-Drehzahl von 5000 aufgenommen wurden. Mithilfe eines Schiebereglers könnte er die Filterkriterien festlegen, worauf hin alle Messdaten mit einer kleineren Drehzahl direkt ausgeblendet werden.

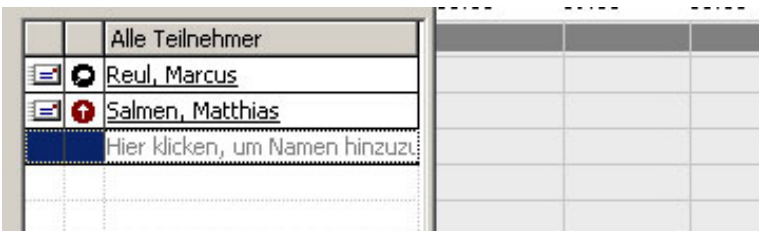
4.11 List Builder

Sogenannte List Builder Tidwell [2005] können überall dort eingesetzt werden, wo die Benutzer aus einer vorhandenen Menge an Elementen eine Teilmenge auswählen sollen. Ein List Builder besteht aus zwei Listen: der Quell- und der Zielliste. Der Benutzer kann anhand von Schaltflächen oder per Drag-and-Drop Elemente aus der Quellliste in die Zielliste verschieben. Ja nachdem, ob man ein und das selbe Element mehrmals hinzufügen kann, verschwinden dabei die entsprechenden Elemente aus der Quellliste oder nicht.

In TCM müssen oft Listen erstellt werden

In TCM kann dieses Prinzip an verschiedenen Stellen angewendet werden. Ein Beispiel hierfür ist die Auswahl der aufzuzeichnenden Kanäle. Vor der Messung muss der Prüfstandsfahrer festlegen, welche der TCM-Kanäle während der Messung aufgezeichnet und abgespeichert werden sollen. Da diese aus der Gesamtmenge der Kanäle ausgewählt werden müssen, ist dies ein typisches Beispiel für den Einsatz eines List Builders.

4.12 New Item Row



„New Item Row“ Tidwell [2005] beschreibt ein Konzept zum Erstellen neuer Einträge in Listen und Tabellen. An-

Elemente werden direkt in der Liste selber erstellt

stelle neue Listenelemente in gesonderten Dialogen zu kreieren, soll die letzte Zeile besondere Editier-Funktionalität besitzen. So kann man neue Einträge direkt in der Liste, beziehungsweise Tabelle erstellen. Dies kann ein gewöhnliches Textfeld sein, aber auch alternative Eingabeelemente wie zum Beispiel Drop-Down-Listen sind denkbar.

Wie schon vorhin erwähnt, besitzt die TCM-Benutzeroberfläche viele Listen. Manche der Listen bieten die Möglichkeit, neue Elemente zu erzeugen. Dies geschieht in der Regel über einen zusätzlichen Dialog, welcher durch den Benutzer aufgerufen werden muss. „New Item Row“ könnte diesen Prozess einfacher und effizienter gestalten.

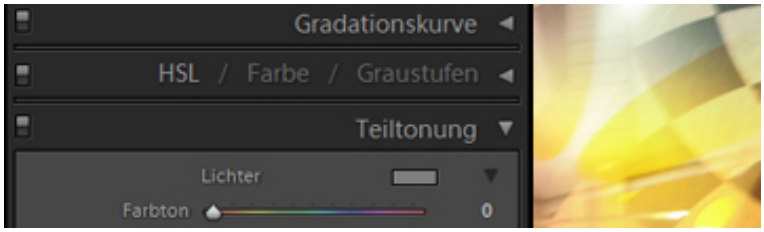
4.13 Jump to Item

Springen macht die Suche in Listen effizienter

Navigiert wird durch Listen meist mithilfe sogenannter Scrollbars oder den Pfeiltasten der Tastatur. Allerdings kann diese Vorgehensweise sehr mühselig sein – vor allem bei langen Listen. „Jump to Item“ Tidwell [2005] beschreibt ein Konzept, bei dem per Tastendruck direkt zu einem bestimmten Bereich innerhalb der Liste gesprungen wird. So kann zum Beispiel in einer alphabetisch sortierten Liste beim Drücken der P-Taste auf der Tastatur direkt zu dem Bereich der Liste gesprungen werden, dessen Einträge mit „P“ beginnen.

Mit seiner Vielzahl an langen Listen ist TCM der ideale Einsatzort für das vorgeschlagene Prinzip. Zur Zeit ist diese Funktionalität noch nicht implementiert.

4.14 Closable Panels

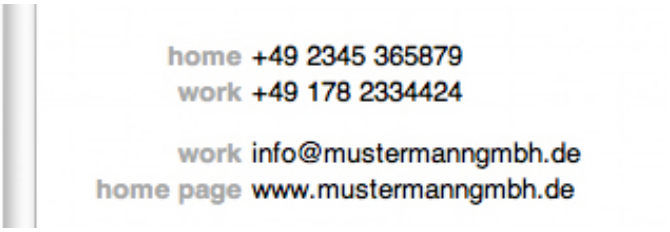


Manchmal reicht der Platz auf dem Bildschirm nicht aus

In manchen Fällen reicht der Platz auf dem Bildschirm nicht aus, um alle Steuerelemente zu platzieren. „Closable Panels“ Tidwell [2005] beschreibt eine Methode, bei der ähnliche Kontrollelemente gruppiert werden. Die einzelnen Gruppen kann der Benutzer durch einen Klick ein- und ausblenden. So müssen nicht immer alle Steuerelemente sichtbar sein, sondern nur diejenigen, die der Benutzer zum Erledigen der Aufgabe benötigt.

Im TCM System könnte dieses Prinzip zum Beispiel innerhalb des Test-Schedulers angewendet werden. Die Test-Scheduler-Bibliothek muss eine Fülle an Methoden bereitstellen, aus welcher der Benutzer die benötigten Elemente auswählt, um so einen Testlauf zu erstellen. Diese Bibliothek könnte entsprechend diesem Entwurfsmuster entwickelt werden.

4.15 Right Left Alignment



„Right Left Alignment“ Tidwell [2005] ist ein Entwurfsmuster, welches sich mit der Ausrichtung von Steuerelementen und ihren Beschriftungen (engl. labels) auseinan-

dersetzt. Fenster bestehen häufig aus mehreren Kontrollelementen (Textfelder, Kontrollkästchen, ...), welche meist untereinander angeordnet sind. Labels beschreiben die Funktion der Kontrollelemente. Oft werden sowohl labels als auch Kontrollelemente linksseitig ausgerichtet. Problematisch wird die Ausrichtung, wenn die Labels eine unterschiedliche Länge haben (Abbildung 4.1). Beim Right Left Alignment werden die Labels rechtsseitig und die Kontrollelemente linksseitig ausgerichtet. Dadurch sieht die Programmoberfläche aufgeräumter aus. Zusätzlich liegen die Labels näher an den zugehörigen Kontrollelementen, was nach dem Gestalt-Prinzip der Nähe Chang et al. [2002] die Zusammengehörigkeit unterstreicht.

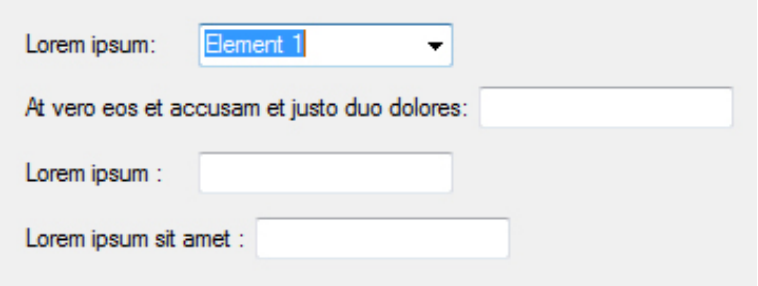
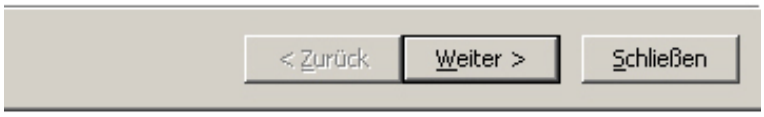


Abbildung 4.1: Schlecht ausgerichtete Kontrollelemente

Kontrollelemente und Beschriftungen sind in TCM oft wahllos platziert. Ein einheitliches Ausrichtungsschema würde nicht nur das optische Erscheinungsbild der Programmoberfläche verbessern, sondern den Benutzern auch helfen, sich in den Anwendungsfenstern besser zurecht zu finden.

4.16 Button Groups

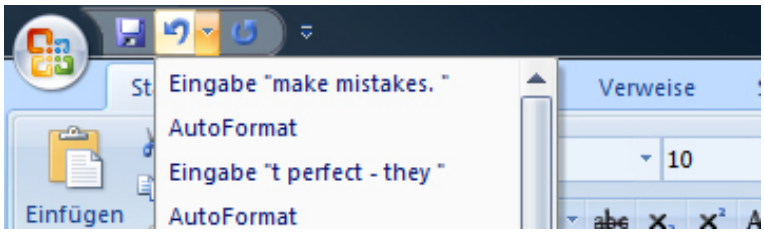


„Button Groups“ Tidwell [2005] beschreibt, dass Schaltflächen (engl. buttons) in Form kleiner Gruppen angeordnet werden sollen. Schaltflächen mit ähnlichen Funktionen sollen dabei eine Gruppe bilden (z.B. „Weiter“ und „Zurück“). Eine Gruppe soll dabei nicht mehr als drei bis vier Schaltflächen beinhalten.

Zusammengehörende Funktionen sollen gebündelt werden

Genauso, wie es keine Regeln für die Ausrichtung für Kontrollelemente gibt, fehlen für TCM auch klare Regeln, zur Ausrichtung von Schaltflächen. Dieses Entwurfsmuster soll die Entwickler daran erinnern, zusammengehörende Funktionen in kleinen Gruppen zu bündeln und diese auch so auf der Programmoberfläche darzustellen.

4.17 Undo

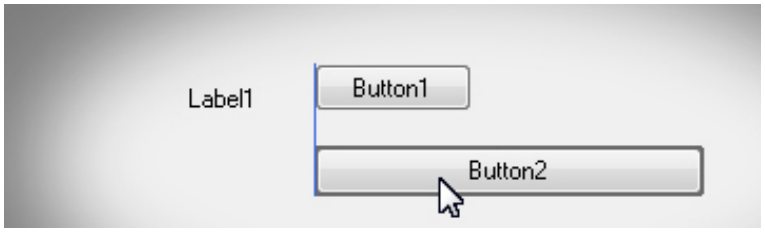


Menschen sind nicht perfekt und machen Fehler – auch beim Bedienen von Computer-Programmen. Aus diesem Grund sollen Software-Systeme (soweit dies machbar ist) dem Benutzer die Möglichkeit geben, vorangegangene Aktionen rückgängig zu machen. Das Entwurfsmuster beschreibt den sogenannten Rückgängig-Befehl (engl. undo), welcher in vielen Programmen zum Einsatz kommt Tidwell [2005]. Nahezu jedes Grafik- und Textverarbeitungsprogramm bietet eine solche Funktionalität.

Benutzer machen Fehler

In TCM fehlt an vielen Stellen noch die Möglichkeit, Eingaben und Aktionen rückgängig zu machen. Eingabefehler sind aus diesem Grund besonders ärgerlich. Die Implementierung einer Rückgängig-Funktion würde den Benutzern eine Menge Ärger ersparen und ihnen helfen, Vertrauen in das System zu gewinnen.

4.18 Magnetism



Das Ausrichten von Elementen ist oft mühselig

Magnetismus (engl. magnetism) Tidwell [2005] hilft dem Benutzer beim Ausrichten von Elementen auf dem Bildschirm. Dieses Prinzip kommt häufig in Grafikprogrammen zum Einsatz. Objekte exakt auszurichten kann eine schwierige und frustrierende Aufgabe für den Benutzer darstellen. Magnetismus unterstützt den Benutzer dabei, indem um potentielle Zielpositionen ein virtuelles „Magnetfeld“ aufgebaut wird. Zieht der Benutzer das Objekt in dieses Magnetfeld, so springt das Objekt direkt an die exakte Zielposition.

Während der Contextual Inquiry ist mir aufgefallen, dass den Prüfstandsfahrern das Ausrichten von neuen Kontrollelementen auf den sogenannten User Defined Screens sehr schwer fällt. Neben genervten Benutzern sind unaufgeräumte Programmoberflächen Folge dieses Problems. Magnetismus würde den Benutzern helfen, die Objekte schneller und präziser zu platzieren.

Kapitel 5

Workshop

"Perfect as the wing of a bird may be, it will never enable the bird to fly if unsupported by the air. Facts are the air of science. Without them a man of science can never rise."

—Ivan Pavlov

Nachdem ich nun einen geeigneten Satz an Entwurfsmustern selektiert habe, gilt es noch den Ansatz hinsichtlich seiner Praxistauglichkeit zu überprüfen.

EIn Workshop soll die Praxistauglichkeit überprüfen

Um eine direkte Rückmeldung von Produkt Managern und Entwicklern zu bekommen, haben wir uns dafür entschiedenen, einen Workshop zu organisieren. Der Workshop sollte Entwicklern und Produkt Managern die Möglichkeit bieten, den vorgeschlagenen Ansatz auszuprobieren und anschließend ihre persönliche Meinung abzugeben.

Im ersten Abschnitt dieses Kapitels werde ich erläutern, welche Ergebnisse ich mir von dem Workshop erhoffte und welche Methoden für die Evaluierung verwendet wurden. Danach werde ich den Ablauf des Workshops beschreiben. Die Evaluierung der Ergebnisse ist in zwei geteilt. Im ersten Teil werden die von den Teilnehmern entwickelten Prototypen evaluiert. Der zweite Teil befasst sich mit der Auswertung des Fragebogens.

5.1 Ziele und Methodik

Hauptziel des Workshops ist es, belegbare Fakten zu gewinnen, die darauf hinweisen, dass HCI Design Patterns Entwicklern und Produkt Managern beim Entwerfen von Benutzerschnittstellen behilflich sein können.

Die Teilnehmer sollen eine Design Aufgabe lösen

Die Workshop-Situation soll die Entwurfsphase einer Programmoberfläche so realistisch wie möglich wieder spiegeln. Anhand einer groben, funktionalen Software-Spezifikation werden die Teilnehmer Prototypen einer Programmoberfläche erstellen. Der erste Prototyp wird ohne die Hilfe der HCI Design Patterns erstellt. Im zweiten Teil des Workshops werden die Teilnehmer einen weiteren Prototyp erstellen – diesmal werden ihnen dazu die 18 ausgewählten HCI Design Patterns an die Hand gegeben.

Eine qualitative und eine quantitative Analyse wurden durchgeführt

Im ersten Teil der Evaluierung werden die Prototypen betrachtet. Im Rahmen einer qualitativen Analyse soll dokumentiert werden, inwiefern sich die Prototypen, welche unter Verwendung der HCI Design Patterns entwickelt wurden, von den ohne Zuhilfenahme der Entwurfsmuster erstellten unterscheiden. In einer quantitativen Analyse wird überprüft, ob in den letztgenannten Prototypen tatsächlich mehr der in den HCI Design Patterns vorgestellten Interaktionskonzepte berücksichtigt wurden. Ein t-Test soll entscheiden, ob das Ergebnis statistisch signifikant ist.

Um eine direkte Einschätzung der Teilnehmer darüber zu bekommen, inwiefern sich der vorgestellte Ansatz auch in späteren Entwicklungsprojekten anwenden lässt, wird am Ende des Workshops ein Fragebogen ausgeteilt *(siehe Anhang B)*. Der Fragebogen beinhaltet mehrere Aussagen, deren Gültigkeit die Teilnehmer auf einer Likert-Skala bewerten sollen. Am Ende des Kapitels werden die gewonnenen Resultate dargelegt und diskutiert.

5.2 Die Workshop Teilnehmer

7 Fachleute haben teilgenommen

An dem Workshop haben fünf Entwickler sowie zwei Produkt Manager teilgenommen. Alle Teilnehmer sind Mitarbeiter der „Test-Systems“-Abteilung der FEV Motorentechnik. Die Entwickler haben unterschiedliche Vorkenntnisse und arbeiten innerhalb der FEV an unterschiedlichen Softwareentwicklungsprojekten – alle jedoch im Bereich Prüfstandssysteme. Auch die beiden Produkt Manager betreuen innerhalb der FEV unterschiedliche Softwareentwicklungsprojekte. Einer der beiden ist Produkt Manager des schon vorher erwähnten und näher betrachteten TCM-Systems.

5.3 Der Ablauf des Workshops

Prototyping Methoden wurden vorgestellt

Der Workshop erstreckte sich über einen Arbeitstag und startete mit einem 30-minütigen Briefing. In diesem Briefing wurde kurz der Tagesablauf besprochen. Daraufhin wurden verschiedene Prototying-Ansätze vorgestellt. Den Teilnehmern wurde erklärt, dass es eine Vielzahl an Methoden zum Erstellen von Benutzerschnittstellen-Prototypen gibt. Auf Papier-Prototypen (engl. Paper-Prototypes) und das Erstellen von Prototypen mithilfe sogenannter User Interface Builder (wie zum Beispiel der in Visual Studio integrierte) wurde näher eingegangen. Auch ein Hybrid Ansatz aus Paper-Prototyping und Software-Prototyping wurde vorgestellt.

Die Design-Aufgaben kamen aus dem Fachgebiet

Anschließend wurde die erste Entwurfsaufgabe vorgestellt: Jeder Teilnehmer sollte einen Prototypen einer Programmoberfläche für eine kombinierte Alarm- und Logliste des TCM-Systems erstellen. In der Aufgabenstellung waren verschiedene Funktionen spezifiziert, für die es eine Benutzerschnittstelle zu entwerfen galt. So war unter anderem vorgeschrieben, welche Eigenschaften von Alarm- und Logeinträgen dargestellt werden sollen und welche Möglichkeiten zum Durchsuchen der Liste die Programmoberfläche bereitstellen soll.

Die Teilnehmer durften selber die Prototyping-Methode wählen

Für die Bearbeitung dieser Aufgabe hatten die Teilnehmer ca. 2,5 Stunden Zeit. Die Entwickler haben die Aufgabe an ihrem gewöhnlichen Arbeitsplatz ausgeführt. Die Produkt Manager hingegen haben den Prototyp in einem Konferenzraum erstellt – die Ablenkung am Arbeitsplatz durch Telefonanrufe und andere Störfaktoren wäre zu groß gewesen. Welche Prototyping-Technik die Teilnehmer hierfür verwenden wurde ihnen selber überlassen. Abschließend wurde den Teilnehmern ein einseitiges Arbeitsblatt ausgeteilt *(siehe Anhang C)*, auf dem die Aufgabe noch einmal genau spezifiziert war.

Die Prototypen wurden präsentiert

Vor der Mittagspause gab es noch eine Zusammenkunft mit allen Teilnehmern, in der die einzelnen Prototypen präsentiert wurden. Jeder Teilnehmer hat seinen Prototyp in einer zweiminütigen Kurzpräsentation vorgestellt. Daraufhin konnten die anderen Teilnehmer Fragen stellen und das vorgestellte Design wurde kurz in der Runde besprochen.

Nachmittags wurden die Patterns vorgestellt

Der zweite Teil des Tages startete wieder mit einem Briefing. Hier wurde das Konzept der HCI Design Patterns vorgestellt und die zweite Entwurfsaufgabe erläutert. Die zweite Aufgabe griff eine aktuelle TCM-Thematik auf. Die Teilnehmer sollten einen Prototyp für die Benutzerschnittstelle eines sogenannten „Test-Schedulers“ entwerfen.

Der Test-Scheduler sollte die Möglichkeit bieten, mithilfe einer grafischen Benutzeroberfläche einen Testablauf zu erstellen. Dies soll durch Verknüpfung bereits vorhandener, elementarer Test-Schritte von statten gehen. Die Benutzeroberfläche soll eine Art Bibliothek bereitstellen, welche die einzelnen Test-Schritte beinhaltet. Dem Benutzer muss nun die Möglichkeit gegeben werden, diese elementaren Einheiten aneinanderzureihen, um so eine Abfolge für einen Testlauf zu erstellen.

Die Aufgabenstellung *(siehe Anhang C)* beinhaltete neben den Anforderungen an die Bibliothek und das verketten von elementaren Einheiten (Methoden) auch Anforderungen an eine Benutzerschnittstelle zum Konfigurieren einzelner Einheiten.

Für die Bearbeitung der zweiten Entwurfsaufgabe waren 2 Stunden vorgesehen. Am Ende des Tages wurden wieder

kurz die einzelnen Prototypen vorgestellt und in der Runde diskutiert. Zum Schluss haben die Workshop Teilnehmer noch den ausgehändigten Fragebogen *(siehe Anhang B)* ausgefüllt.

Ein Lerneffekt spielt keine Rolle

Weil das Erstellen von Prototypen für Benutzerschnittstellen für die meisten Teilnehmer neu war, entstand ein Lerneffekt. Bei der zweiten Aufgabe waren die Teilnehmer schon etwas vertrauter mit der kurz vorher vorgestellten Paper-Prototyping Methode. Dieser Effekt hat jedoch keinen Einfluss auf die für mich relevanten Ergebnisse. In der Evaluierung geht es vor allem um den Einsatz der Entwurfsmuster und nicht um die Güte der Prototypen. Da die Workshop Teilnehmer für die zweite Aufgabe etwas weniger Zeit hatten, ist durch die Vertrautheit mit der Prototyping-Methode auch kein Zeitvorteil entstanden.

5.4 Evaluierung

5.4.1 Auswertung der Prototypen

Qualitative Analyse der Prototypen

Auf Patterns wurde referenziert

Beim Betrachten der Prototypen der zweiten Entwurfsaufgabe fällt direkt auf, dass die meisten Teilnehmer die in den Entwurfsmustern vorgestellten Konzepte verwendet haben. Die beiden Produkt Manager haben sogar in ihren Papier-Prototypen auf Entwurfsmuster referenziert *(siehe Abbildung 5.1)*.

Positive Resonanz bezüglich des Paper-Prototypings

Eine weitere Erkenntnis aus dem Workshop ist die überaus positive Resonanz auf das Paper-Prototyping. Zu Beginn der ersten Entwurfsaufgabe haben sechs der sieben Teilnehmer versucht einen Software-Prototyp zu erstellen (mithilfe von Power Point, Visual Studio, Visio, ...). Gegen Ende der Bearbeitungszeit der ersten Aufgabe wurde einigen klar, dass die Zeit wahrscheinlich nicht reicht und sie schwenkten noch kurzfristig auf Papier-Prototypen um. Bei der zweiten Entwurfsaufgabe haben sechs der sieben Teilnehmer einen Papier-Prototypen erstellt. In nachfolgenden

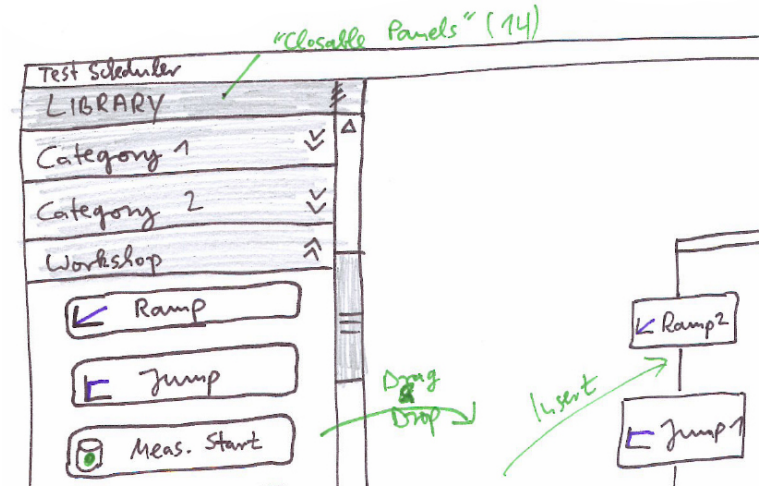


Abbildung 5.1: Referenz auf das Entwurfsmuster „Closable Panels“ in einem Papier-Prototyp

Gesprächen erzählten mir die Teilnehmer, dass sie verblüfft sind, wie schnell und einfach man Prototypen mithilfe von Papier und Post-Its erstellen kann. Mehrere Teilnehmer bekundeten, dass sie diese Methode auch in zukünftigen Projekten einsetzen möchten. Ein Produkt Manager verwendet die in diesem Workshop gelernte Prototyping Methode noch heute.

Ein Produkt Manager konnte die Muster am besten einsetzen

Am besten konnten die Patterns von dem Produkt Manager eingesetzt werden, der sich bereits im Vorfeld mit der Test-Scheduler-Thematik auseinander gesetzt hat. Zur Zeit des Workshops lief bei der FEV ein Projekt an, in dem es um die Entwicklung eines Test-Schedulers geht. Der Produkt Manger, welcher in diesem Projekt involviert ist, hatte sich schon im Vorfeld des Workshops damit beschäftigt, wie eine entsprechende Benutzeroberfläche aussehen kann. Mithilfe der Patterns konnte er seine Ideen verfeinern. Zum Beispiel zog er in dem von ihm erstellten Prototyp in Betracht, neue Listenelemente mit dem in der Pattern-Sammlung gefundenen Konzept „New Item Row“ zu erstellen *(siehe Abbildung 5.2)*.

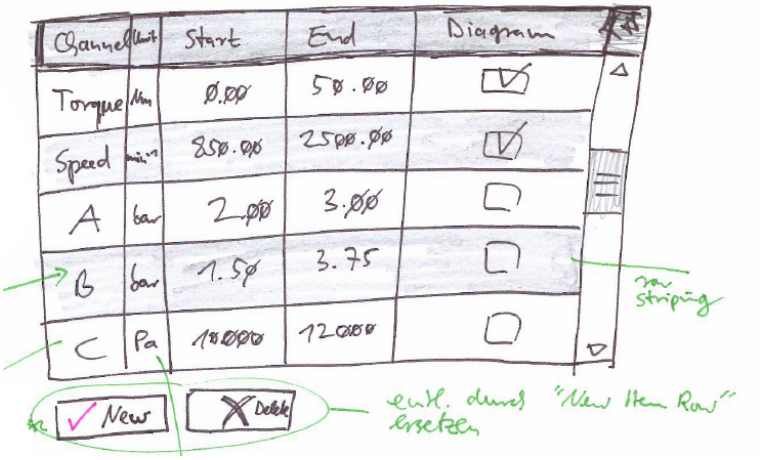


Abbildung 5.2: Referenz auf das Entwurfsmuster „Row Striping“ und „New Item Row“ in einem Papier-Prototyp

Quantitative Analyse der Prototypen

In einem weiteren Teil der Evaluierung habe ich analysiert, wie viele der in den Entwurfsmustern vorgestellten Konzepte in den Prototypen angewendet wurden. Die Ergebnisse habe ich in einer Tabelle zusammengefasst.

Bei der zweiten Aufgabe wurden die Konzepte wesentlich häufiger verwendet

Schon auf den ersten Blick sieht man, dass bei der zweiten Aufgabe wesentlich mehr der in den Entwurfsmustern vorgestellten Konzepte angewendet wurden*(siehe Tabelle 5.1)*. Da die Entwurfsmuster erst vor der zweiten Aufgabe vorgestellt wurden, mag dieses Ergebnis nicht verwundern. Es zeigt jedoch, dass die Entwurfsmuster nicht nur wohlbekannte Konzepte beinhalten, sondern vor allem solche, die die Teilnehmer noch nicht kannten oder an die sie beim Entwurf nicht denken. Verstärkt wird dieser Befund noch durch die Tatsache, dass die Entwurfsmuster – bedingt durch die vielen Listen-bezogenen Konzepte (New Item Row, Row Striping, ...) – eigentlich viel einfacher in der ersten Aufgabe hätten angewendet werden können

Das Ergebnis ist statistisch relevant

Um schließlich noch zu zeigen, dass das Ergebnis auch statistisch signifikant ist (sprich dass der Mittelwert der zweiten Reihe signifikant größer ist als der der ersten Reihe), habe ich einen einseitigen gepaarten t-Test durchgeführt.

Die aus dem Test resultierende Irrtumswahrscheinlichkeit beträgt 0,007, womit das Ergebnis statistisch signifikant ist (unter Berücksichtigung eines Signifikanz-Levels von 0,01).

Teilnehmer Nr.	1	2	3	4	5	6	7	Durchschnitt
Aufgabe 1	1	1	0	0	0	0	0	0,3
Aufgabe 2	4	3	5	3	0	0	3	2,6

Tabelle 5.1: Anzahl der verwendeten Patterns

5.4.2 Auswertung der Fragebögen

Der ausgehändigte Fragebogen *(siehe Anhang B)* ist wichtiger Bestandteil der Evaluierung. Auf einer fünf-teiligen Likert-Skala sollten die Teilnehmer unter anderem beurteilen, in wieweit sie sich vorstellen können, die präsentierten Konzepte in realen Entwicklungsprojekten einzusetzen.

In den ersten Fragen des Fragebogens wurde abgefragt, welche Vorkenntnisse die verschiedenen Teilnehmer hatten und wie häufig sie an der Konzeption und Entwicklung von Programmoberflächen beteiligt sind. Die Ergebnisse zeigen, dass das Konzept der HCI Design Patterns für alle Workshop Teilnehmer neu war. Sechs der sieben Teilnehmer haben schon einmal Prototypen für User Interfaces entwickelt. Allerdings ist der Prozentsatz der Arbeitszeit, welchen sie für den Entwurf oder die Implementierung von Benutzerschnittstellen verwenden eher gering. Dies kann dadurch erklärt werden, dass es sich bei der FEV um eine relativ kleine Softwareentwicklungsabteilung handelt und die Entwickler sehr vielfältige Aufgaben bewältigen müssen. Spezialisten, die sich überwiegend mit der Entwicklung von Programmoberflächen beschäftigen, gibt es nicht.

Im zweiten Abschnitt des Fragbogens geht es darum, ob die Entwurfsmuster die verschiedenen Konzepte klar und verständlich rüber bringen konnten. Die Teilnehmer stuften die Patterns als verständlich und übersichtlich ein. Der Umfang der Patterns scheint genau richtig zu sein. Das

Inhaltsverzeichnis hat den Teilnehmern eher weniger geholfen. Bei der Frage, ob die Entwurfsmuster inhaltlichen Bezug zum Thema „Prüfstandsautomatisierungssysteme“ haben, waren sich die meisten Teilnehmer jedoch eher unschlüssig.

Folgende Grafiken veranschaulicht zu jeder Frage den Arithmetischen Mittelwert der Antworten, sowie das Intervall, in dem sich der tatsächliche Mittelwert mit einer 95-prozentigen Wahrscheinlichkeit aufhält.

Die Likert-Skala ist wie folgt normiert: (1) Dem stimme ich gar nicht zu, (2) Dem stimme ich eher nicht zu, (3) Unentschieden, (4) Dem stimme ich eher zu, (5) Dem stimme ich voll zu.

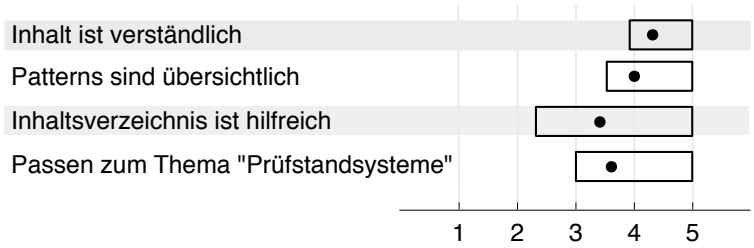


Abbildung 5.3: Ergebnisse des Fragebogens

Der nächste Fragenblock bezieht sich auf die Fragestellung, inwieweit den Teilnehmern die vorgestellten Interaktionskonzepte und Probleme neu waren. Nur für drei der sieben Teilnehmern waren die in den Entwurfsmustern vorgestellten Konzepte neu. Sechs der Teilnehmer gaben an, dass die Patterns sie auf bekannte Lösungen aufmerksam gemacht haben, an die sie ohne die Entwurfsmuster nicht gedacht hätten.

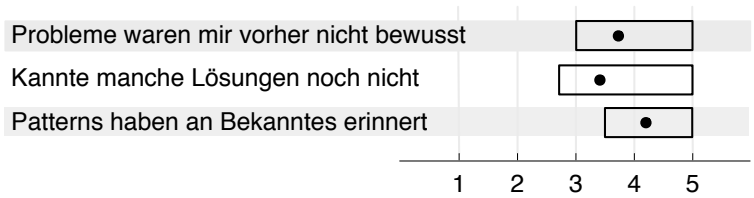


Abbildung 5.4: Ergebnisse des Fragebogens

Eins der Ziele des letzten Fragenblocks war es, herauszufinden in welcher Phase des Entwicklungsprozesses die Teilnehmer am meisten Potential für den Einsatz von HCI Design Patterns sehen. Die Anwendbarkeit der Patterns in den drei verschiedenen Phasen des Entwicklungsprozesses (Anforderungsanalyse, Spezifikation des UI, Implementierung) wurde dabei durchweg positiv bewertet.

Patterns wären hilfreich in realen Projekten

Die Ergebnisse lassen allerdings keine Rückschlüsse darauf ziehen, dass eine der Entwicklungsphasen besser geeignet ist als eine andere. Die Antworten zu der Frage, ob die Teilnehmer sich vorstellen können, die Patterns in eigenen Entwicklungsprojekten einzusetzen, fiel sehr positiv aus:

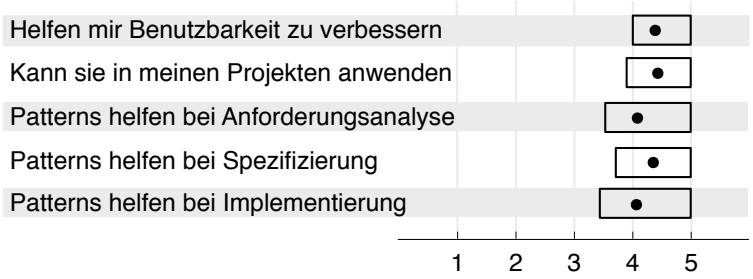


Abbildung 5.5: Ergebnisse des Fragebogens

5.4.3 Zusammenfassung der Ergebnisse

Die Workshop-Resultate lieferten uns starke Anzeichen dafür, dass HCI Design Patterns Produkt Managern und Entwicklern beim Entwurf von benutzerfreundlichen Programmoberflächen behilflich sein können.

Entwickler und Produkt Manager ohne vorherige Usability-Erfahrung konnten die Entwurfsmuster in kürzester Zeit anwenden. Alle Teilnehmer waren der Meinung, dass die in den Patterns beschriebenen Konzepte ihnen dabei helfen können, die im Haus entwickelte Software benutzerfreundlicher zu gestalten. Alle Teilnehmer können sich vorstellen, die Patterns in laufenden Entwicklungsprojekten anzuwenden.

Kapitel 6

Ergebnisse und weiterführende Arbeiten

"Creativity is allowing yourself to make mistakes. Design is knowing which ones to keep."

—Scott Adams

6.1 Zusammenfassung und Ergebnisse

Im Rahmen dieser Arbeit wurden Anforderungen und Bedürfnisse der Benutzer von Motorenprüfständen erforscht. Aus den daraus gewonnenen Ergebnissen wurde eine Sammlung an Entwurfsmustern zur Erstellung von Benutzerschnittstellen zusammengestellt. Diese Entwurfsmuster sollen Produkt Managern und Softwareentwicklern helfen, Prüfstandssysteme zu entwickeln, die anwenderfreundlicher und effizienter benutzbar sind.

Als Erstes habe ich eine Feldstudie durchgeführt, die mir geholfen hat, das für mich neue Gebiet der Motorenprüfstände genauer kennen zu lernen. Des Weiteren gab mir die Feldstudie die Möglichkeit, die Benutzer des Systems, ihre täglichen Arbeitsaufgaben, sowie ihre Arbeitsumgebung kennen zu lernen. Aus den Ergebnissen dieser Studie konnte ich Anforderungen und Bedürfnisse der Be-

nutzer ableiten, sowie Usability-Probleme in der aktuellen Prüfstandssoftware identifizieren.

In dem nächsten Schritt meiner Arbeit habe ich 13 verschiedene Pattern-Sammlungen mit insgesamt über 600 Entwurfsmustern für Benutzerschnittstellen durchsucht. Aus dieser Fülle an Entwurfsmustern, habe ich 18 ausgewählt, die meiner Meinung nach auf die aktuellen Usability-Probleme der Prüfstandssoftware eingehen und dabei die Bedürfnisse der Benutzer berücksichtigen.

Mithilfe eines Workshops wollte ich untersuchen, ob diese Entwurfsmuster tatsächlich Entwicklern und Produkt Managern beim Entwurf der Benutzeroberflächen behilflich sein können.

Die Ergebnisse des Workshops sind sehr positiv

Die Ergebnisse des Workshops waren durchweg positiv. Entwickler und Produkt Manager ohne vorherige Erfahrung mit HCI Design Patterns haben in kürzester Zeit die in den Entwurfsmustern vorgestellten Interaktionskonzepte aufgegriffen und konnten diese in ihre Entwürfe einfließen lassen. Die Ergebnisse des ausgehändigten Fragebogens untermauern dies noch einmal. Die Workshop Teilnehmer bestätigten, dass die ausgehändigten Entwurfsmuster ihnen dabei helfen können, die Benutzbarkeit der entwickelten Software zu verbessern und können sich gut vorstellen, die Patterns in kommenden Softwareentwicklungsprojekten anzuwenden.

6.2 Weiterführende Arbeiten

Obwohl die Ergebnisse dieser Arbeit stark darauf hindeuten, dass der vorgestellte Ansatz funktioniert, bleiben noch einige Frage offen. In diesem Abschnitt möchte ich auf diejenigen Fragen eingehen, welche mir am interessantesten und lohnenswertesten für weiterführende Untersuchungen in diesem Bereich erscheinen.

6.2.1 Tests in realen Entwicklungsprojekten

Als wichtigsten nächsten Schritt sehe ich die Erprobung des Ansatzes in einem realen Softwareentwicklungsprojekt. Obwohl ich bemüht war, die Workshop-Aufgaben so nah wie möglich an der Realität eines richtigen Entwicklungsprojektes zu orientieren, konnte an dem einen Tag natürlich kein gesamter Entwicklungsprozess mit all seinen Phasen durchlaufen werden.

In welcher Phase sollen die Patterns benutzt werden?

In diesem Rahmen wäre es interessant zu untersuchen, in welcher Phase des Entwicklungsprozesses die Patterns zum Einsatz kommen können. Des Weiteren könnte ermittelt werden, inwiefern die Entwurfsmuster zur Diskussion verschiedener Entwürfe einer Benutzerschnittstelle zum Einsatz kommen können - beispielsweise während der Meetings in der Phase der Anforderungsspezifizierung.

Patterns als Anforderungen für die Beschaffung?

Des Weiteren könnte beleuchtet werden, inwiefern sich die Patterns zur Formulierung von Anforderungen gegenüber Lieferanten eignen. Wie in der Einleitung bereits beschrieben, werden im industriellen Sektor oft Teile der Entwicklung aus dem Unternehmen ausgelagert, beziehungsweise es werden Komponenten eingekauft. Bei der FEV, wie auch in anderen Unternehmen, werden Komponenten für die Programmoberflächen (sogenannte widgets) häufig über Zulieferer bezogen. Die Entwurfsmuster könnten im Beschaffungsprozess Randbedingungen für die einzukaufenden Komponenten darstellen. So könnten beispielsweise alle Entwurfsmuster, die Interaktionsprinzipien mit Listen und Tabellen beschreiben, für den Einkauf eines neuen Tabellen-widgets zu Rate gezogen werden.

6.2.2 Entwicklungs-Unterstützung

Die meisten der ausgewählten Entwurfsmuster beschreiben eine Lösung zu einem allgemeinen Problem, welche an vielen verschiedenen Stellen der Benutzeroberfläche zum Einsatz kommen kann. Es ist naheliegend, für diese Standardlösungen vorgefertigte Code-Stücke bereitzustel-

len, welche dann nur noch an die verschiedenen Gegebenheiten angepasst werden müssen.

Patterns als UI Module?

Beispielsweise könnte eine erweiterte Liste als Benutzerschnittstellenkomponente entwickelt werden, welche die verschiedenen Entwurfsmuster für Listen und Tabellen berücksichtigt. So bräuchten die Softwareentwickler während der Entwicklung nur noch die richtigen Benutzerschnittstellenkomponenten einzubinden und anzupassen. Sie müssten sich keine Gedanken mehr um die eigentlichen Entwurfsmuster machen.

6.2.3 Verteil- und Weiterentwicklungsmethoden

Ein System zum Verwalten der Patterns ist denkbar

Die Entwurfsmuster müssen auf irgendeinem festgeschriebenen Weg zu den Entwicklern und Produkt Managern gelangen. Der in dieser Arbeit vorgestellte Ansatz macht keine Aussagen darüber, wie der Transport der Entwurfsmuster vonstatten gehen soll. In dem Workshop habe ich den Teilnehmern die Patterns in Form von Papier-Ausdrucken ausgehändigt. Wenn man bedenkt, dass die einzelnen Entwurfsmuster über die Zeit hinweg weiterentwickelt werden und die Sammlung mit neuen Entwurfsmustern erweitert werden kann, scheint eine elektronische Weitergabe der Patterns vernünftiger.

Interessant scheint die Frage, wie ein solches System aussehen soll, das die elektronische Weitergabe der Entwurfsmuster innerhalb eines Unternehmens ermöglicht. Obwohl Scott Henninger schon viel Forschung in dieser Richtung betrieben hat, besteht auch hier noch Raum für Weiterentwicklungen und Verbesserungen.

6.2.4 Inhaltsverzeichnis

Suchen in größeren Sammlungen ohne Inhaltsverzeichnis ist mühselig

Das von mir erstellte Inhaltsverzeichnis fand während des Workshops eher weniger Beachtung. Die meisten Teilnehmer haben sich durch Blättern in der Pattern-Sammlung zurecht gefunden. Bei 18 Entwurfsmustern und somit 18 Din-A4 Seiten mag dies noch eine geeignete Methode zur

Orientierung sein. Bei größeren Sammlungen kann diese Vorgehensweise jedoch recht schnell an ihre Grenzen stoßen. Daher wird ein gut strukturiertes Inhaltsverzeichnis unerlässlich. Das Inhaltsverzeichnis muss dem Benutzer die Möglichkeit geben, geeignete Entwurfsmuster für seine Entwurfsaufgabe zu finden, ohne die gesamte Sammlung zu durchsuchen.

Eine Kontext-basierte Suche ist möglicherweise hilfreich

Diese Aufgabe mag auf den ersten Blick trivialer erscheinen, als sie es in Wirklichkeit ist. Eine problemorientierte Suche ist wenig zielführend, da den Entwicklern die Probleme, welche die Patterns beschreiben, während dem Entwurf meist gar nicht bewusst sind (diese Erkenntnis wurde unter anderem aus dem Workshop gewonnen). Möglicherweise ist eine Kontext-basierte Suche eine gute Lösung zu diesem Problem. Diese und andere Fragen bezüglich des Aufbaus und der Struktur eines solchen Inhaltsverzeichnisses erachte ich als eine weitere interessante Fragestellung für zukünftige Arbeiten.

Anhang A

Die Entwurfsmuster

FEV User Interface Design Pattern Catalog | Table of Contents

Input / Editing Values and Strings

Output / Displaying Data

Output / Feedback

Working with Lists

UI Layout / Navigation

General

01 Edit in Place

Context

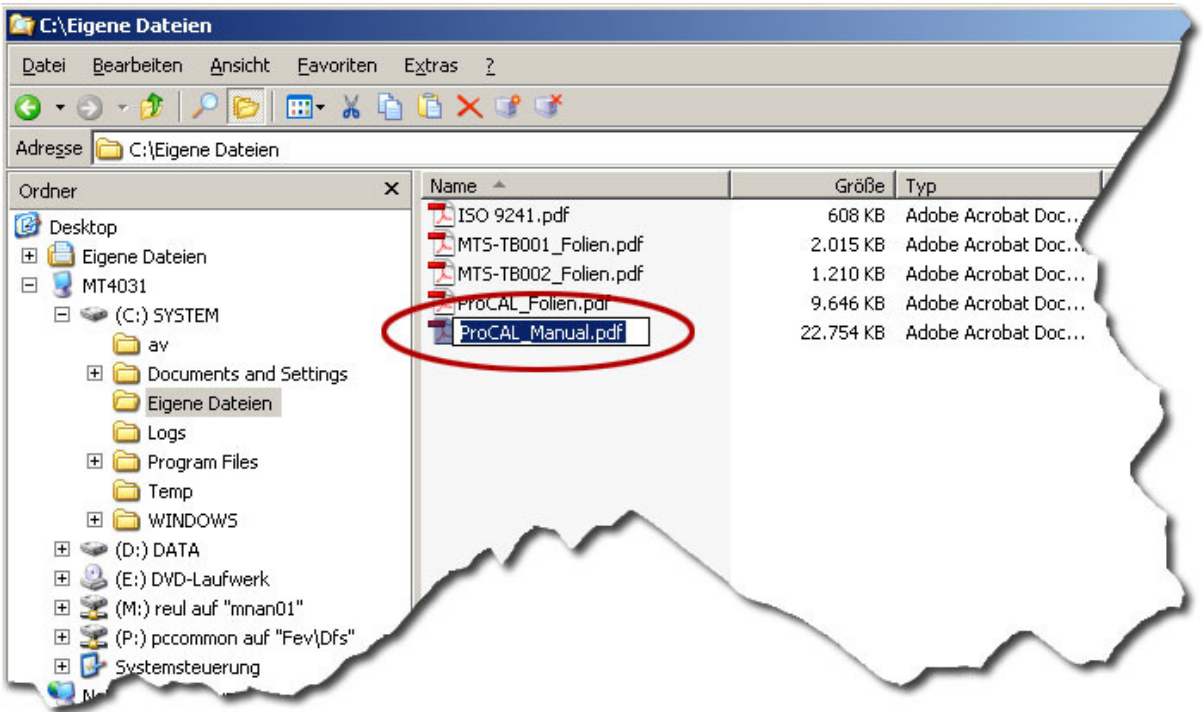
...your user interface contains some text that the user may want to change sometimes.

Problem

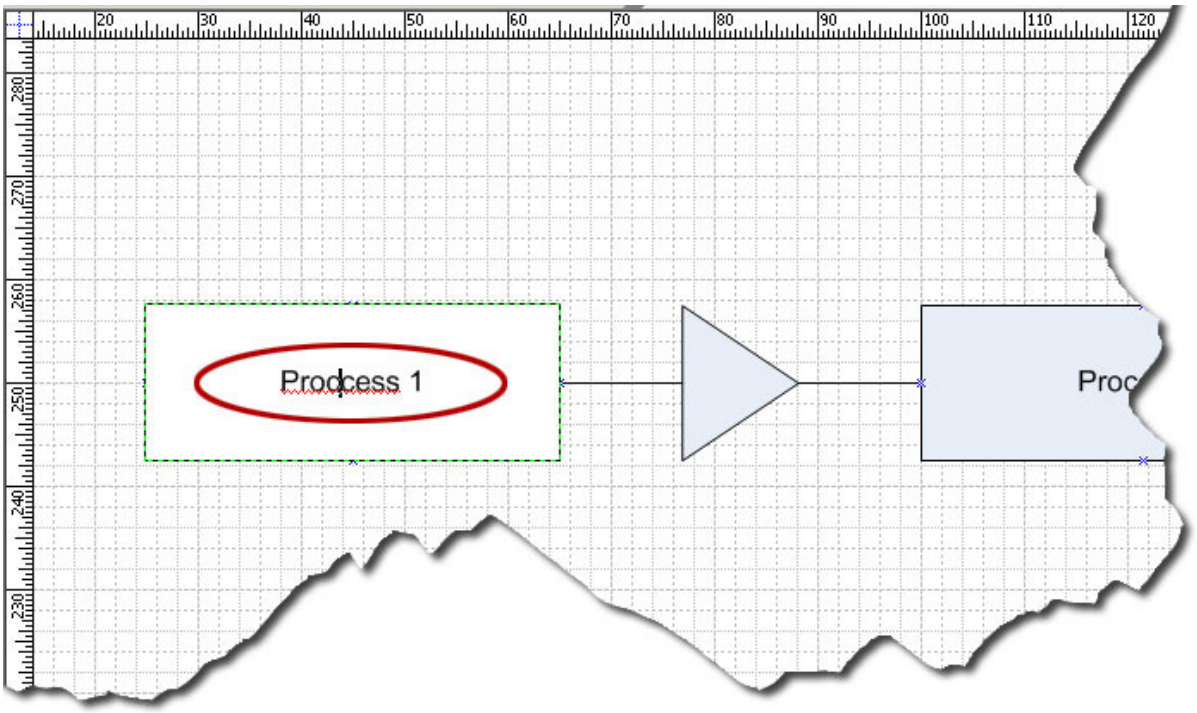
Making the user go somewhere else for editing a text or a value that is already shown somewhere on the screen usually isn't a good idea. The user may not find the place where he can edit the thing, he wants to change. It also takes time to switch the attention from one place to another. Finally, additional editors increase the complexity of the user interface unnecessarily.

Examples

The Windows Explorer lets the user edit filenames directly on the place where the filename stands. When the user clicks on a selected file, a little text editor opens and allows the user to change the name of the file.



Microsoft Visio lets the user change names of nodes and transitions by simply clicking at them.



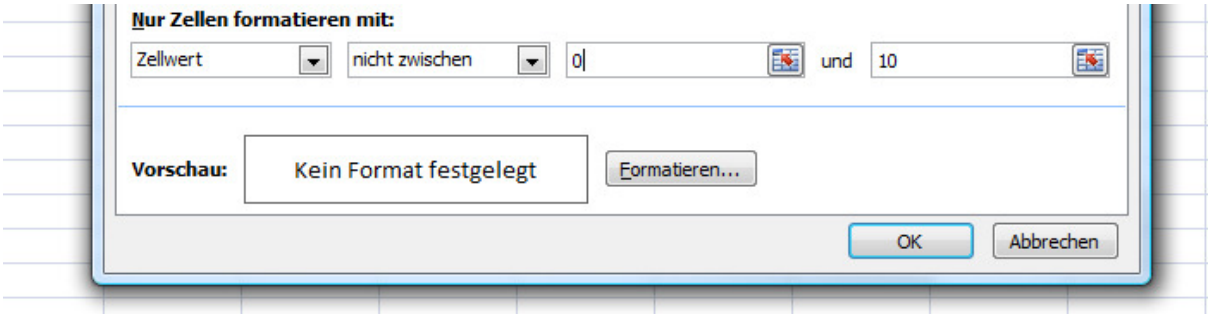
Solution

Use a small, dynamic editor to let the user change text right "in place". Position the editor directly over the original text, rather than using a separate panel or dialogue box.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

02 Fill in the Blanks



Context

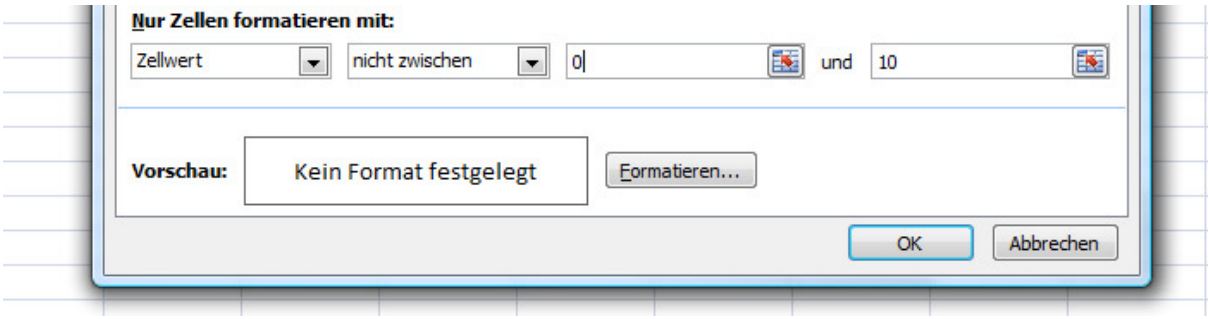
…you need to ask the user for input (usually one-line text, a number, or a choice from a dropdown list).

Problem

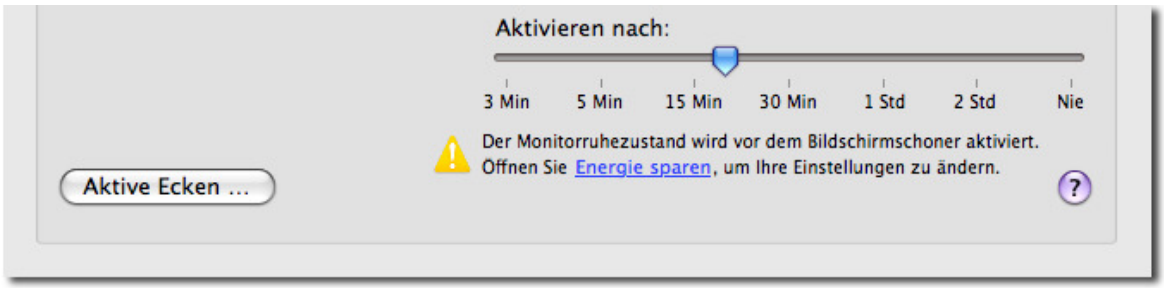
Input forms consist in most cases of a set of label/control pairs (e.g. a label and a dropdown box, a label and a text field, …). However, the labels' typical declarative style (such as "Name: ", "Min: ", "Max:", …) isn't clear enough for users to understand what's going on.

Examples

Some applications, however, verbally describe the action to be taken once everything is filled out, in an active-voice sentence. The following Excel cell-formatting dialog box lets you choose the phrases in this "sentence" from dropdown boxes. As the phrase change, the subsequent text fields (showing 0 and 10 in this example) might be replaced by other controls, such as a single text field for "equal to".



The Mac OS X System Preferences has several places that use a simple Fill-in-the-Blanks approach. Here's one:



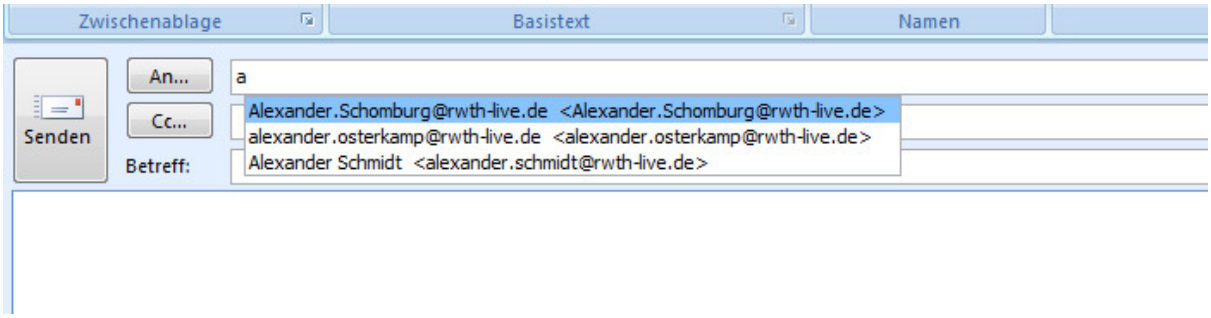
Solution

Arrange one or more fields in a form of a prose sentence or phrase, with the fields as "blanks" to be filled in by the user.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

03 AUTOCOMPLETION



Context

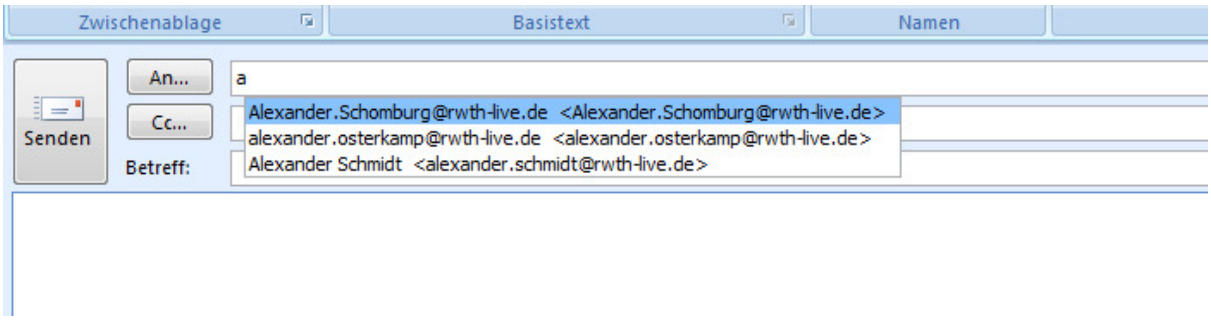
…the user types something predictable (such as a URL, a channel name, a filename, …).

Problem

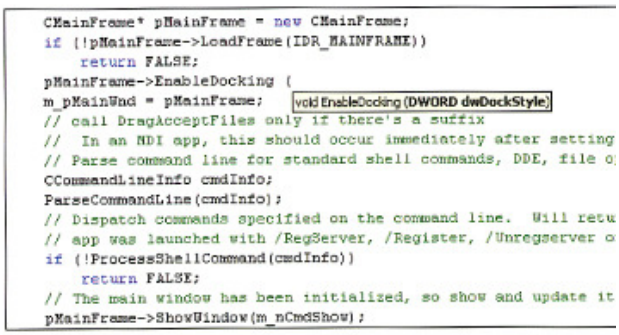
Entering long strings takes valuable time and is a frequent source of error. The longer and stranger the string that must be typed, the greater the possibility that the user makes a typographical error.

Examples

Many email clients use Autocompletion to help users fill in TO:, CC:, and BCC: fields. They generally draw on an address book, contacts list, or a list of addresses you've exchanged email with. This example from Microsoft Outlook shows a list of completions suggested upon typing the letter "a"; the first suggestion is automatically highlighted, so a single keystroke can get rid of it. You can thus type straight "through" the completion (if it's wrong).



Code editors such as Visual Studio invest in very complex Autocompletion mechanisms. Visual Studio's IntelliSense completes the built-in keywords of a programming language, of course, but it also draws on the functions, classes, and variable names defined by the user. It even can show the arguments to functions that you invoke (in the righthand screenshot). Autocompletion in Visual Studio thus serves as a typing aid, a memory aid, and a browser of context-appropriate functions and classes.



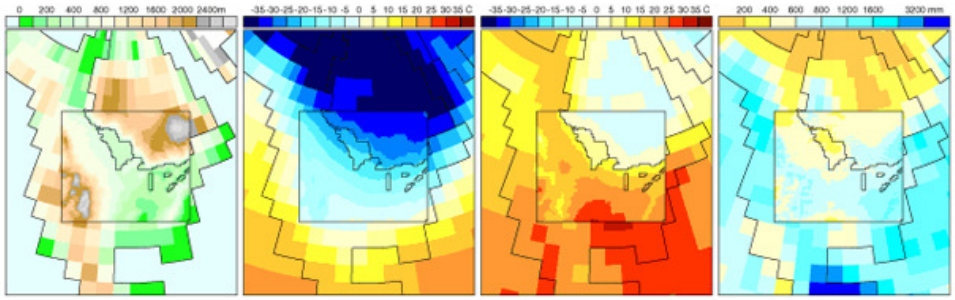
Solution

As the user types into a text field, anticipate the possible answers and automatically complete the entry when appropriate.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

04 Resolution Indicates Data Quality



Context

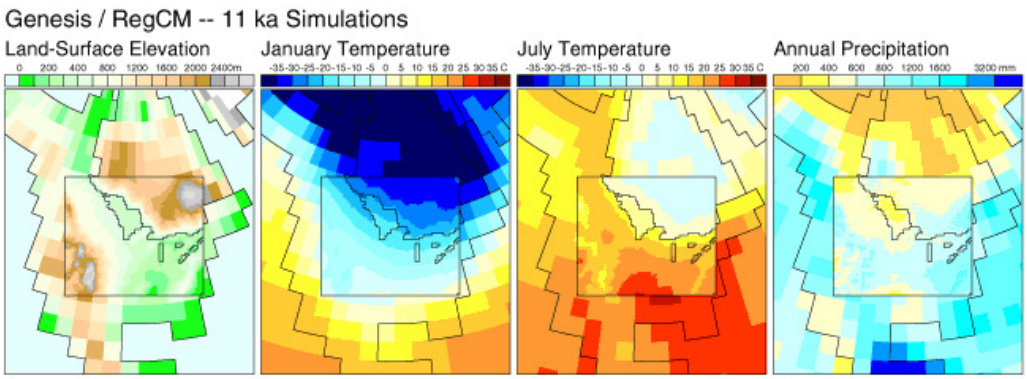
…you want to represent measured data in a graphical form.

Problem

The resolution of measured data is restricted by the performance of the measurement unit. To make the visual appearance more appealing, the underlying data is often smoothed by interpolation algorithms. However, the scale of measurement is an important characteristic of data and should not be obscured. High-resolution presentation of the data, or the use of high-resolution reference overlays (wire frames …), can obscure important limitations of the data and hide the nature of the underlying model.

Examples

This pattern comes originally from the field of meteorology. In meteorology, the measurement results often come from different measuring stations and are therefore delivered in various resolutions. Because maps with smooth gradients are more appealing to the eye, it is tempting to interpolate the data before presenting it to the user. However, interpolating the data takes away important information about the data quality and the actual measurement results. Therefore, meteorologic software often represents the data at the resolution, it was measured. In a map that contains measurement results from different measurement stations, the resolution can also vary from place to place. The following picture shows a map that combines the data from two distinct measuring apparatus – for the area in the middle of the map, the measurement results are available in a much higher resolution than they are at the margins.



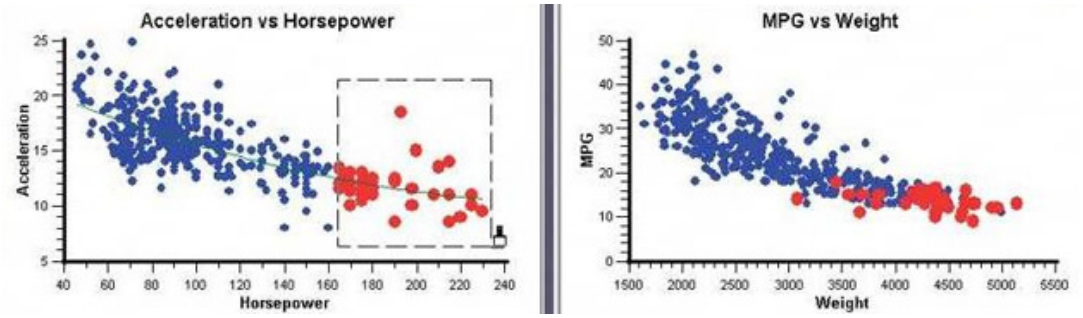
Solution

Do not use interpolation algorithms to display measured data. Indicate the resolution of the underlying data by showing grid-cells and reference maps at their measured or recorded resolution. If effective resolution varies over the time, vary the display accordingly. When the data is insufficiently dense to provide a good visual representation without smoothing measurement points, consider an alternate representation.

* * *

[1] Design Patterns for Data Graphics: http://geography.uoregon.edu/datagraphics/patterns/index.htm, 2007.

05 DATA BRUSHING



Context

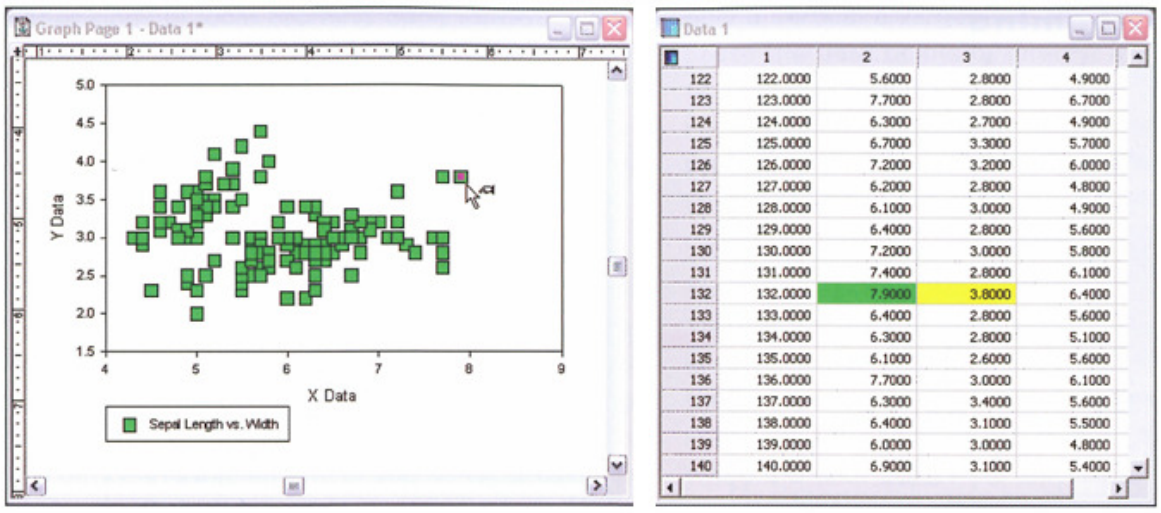
…you show two or more information graphics of a single data set at the same time. For example: you might have a scatter plot and a table, a scatter plot and a line plot, or two scatter plots with different axes, whatever – as long as each graphic shows the same data set.

Problem

When you use different views on the same data set, it is often hard to spot, which data points belong together. You might have several 2-dimensional graphs to display a set of data points of a higher dimension. In many cases it is important to know, which points of the different graphs represent the same measurement point.

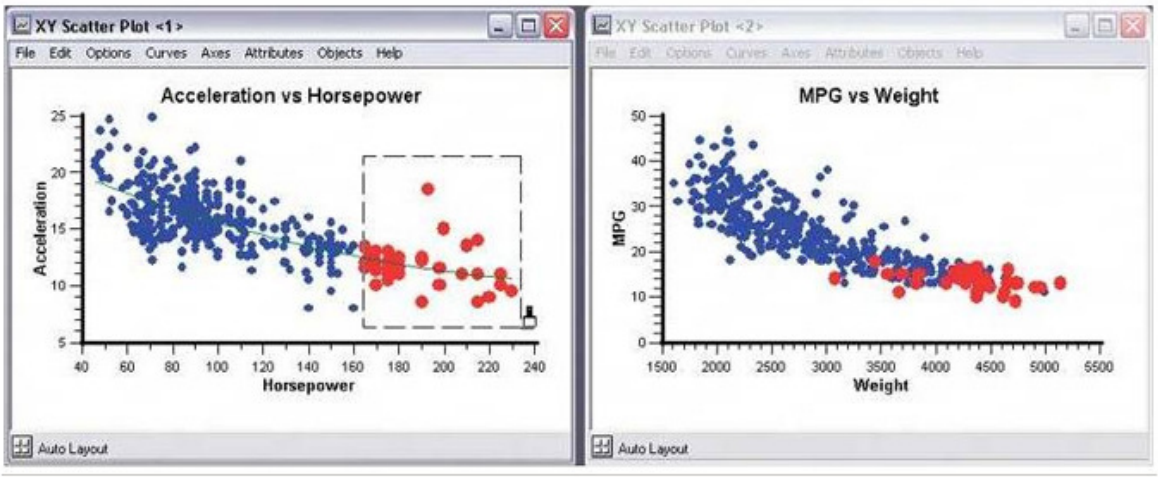
Examples

Sigmaplot permits the selection of single data points – both in its graph view and the table view. The selected point will be highlighted with a special color – in the table as well as in the graph.



	1	2	3	4
122	122.0000	5.6000	2.8000	4.9000
123	123.0000	7.7000	2.8000	6.7000
124	124.0000	6.3000	2.7000	4.9000
125	125.0000	6.7000	3.3000	5.7000
126	126.0000	7.2000	3.2000	6.0000
127	127.0000	6.2000	2.8000	4.8000
128	128.0000	6.1000	3.0000	4.9000
129	129.0000	6.4000	2.8000	5.6000
130	130.0000	7.2000	3.0000	5.8000
131	131.0000	7.4000	2.8000	6.1000
132	132.0000	7.9000	3.8000	6.4000
133	133.0000	6.4000	2.8000	5.6000
134	134.0000	6.3000	2.8000	5.1000
135	135.0000	6.1000	2.6000	5.6000
136	136.0000	7.7000	3.0000	6.1000
137	137.0000	6.3000	3.4000	5.6000
138	138.0000	6.4000	3.1000	5.5000
139	139.0000	6.0000	3.0000	4.8000
140	140.0000	6.9000	3.1000	5.4000

Cornerstone (a statistics and graphing package) provides the ability of selecting a set of data points. The selected points will be highlighted in all alternative views.



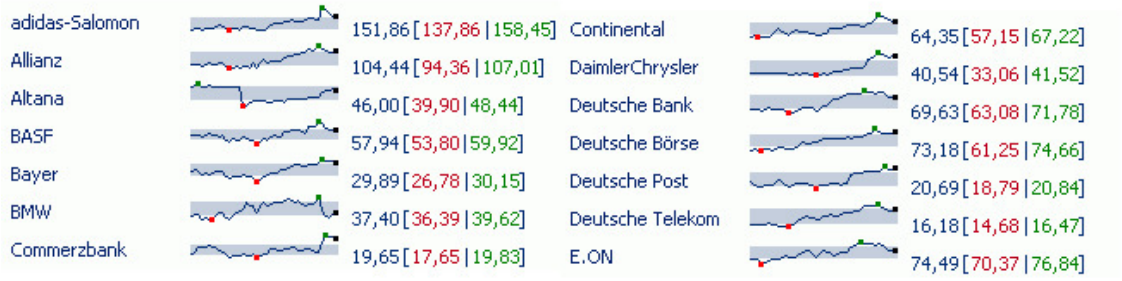
Solution

Let the user select data in one view; show the same data selected simultaneously in another view.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

06 SPARKLINES



Context

…your application must provide information about data that change over time (such as measurement data).

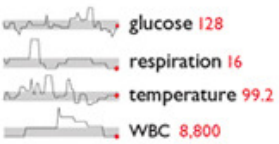
Problem

Measured data is often displayed as a pair of a label and its current value (e.g. POEL: 5,04 bar). However, the value of a measurement parameter changes in many cases several times a second. To better interpret the actual behaviour of the value, a graph would be useful. But, graphs take large amounts of screen space. If there are many measurement parameters to be visualized or screen space is very limited, graphs are not realizable. In addition, graphs provide often too detailed information, which distracts from the bare essentials.

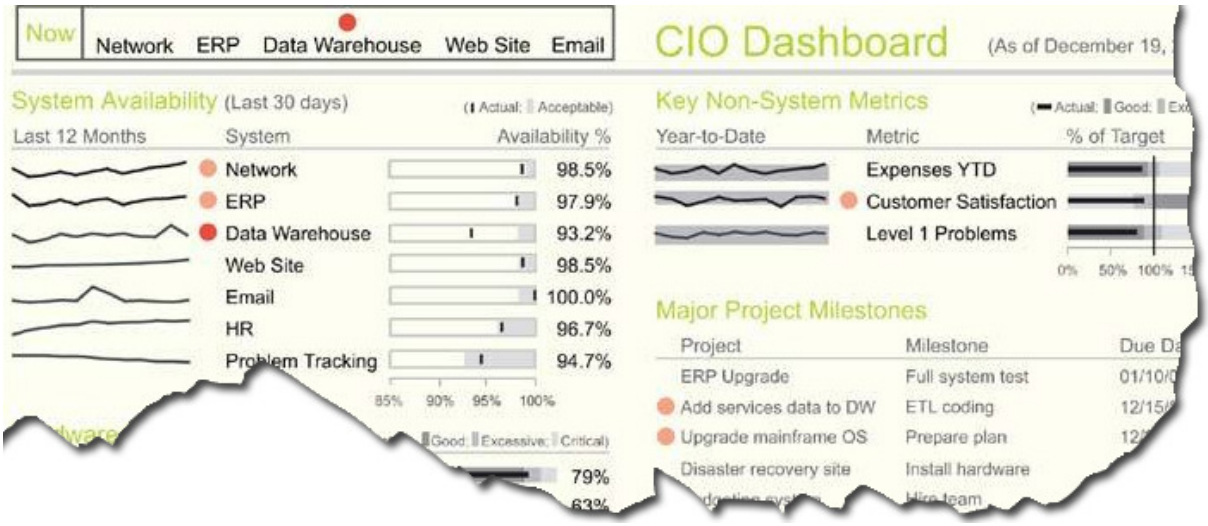
Examples

Edward Tufte presented a concept that is called “Sparklines”. He proposes to accompany each value with a tiny graph showing behaviour of a value. The graph must be very minimalistic and should concentrate on the fundamental: the trend of the measurement value. Axes, captions, frames, and other decorations distract from the essence. Because graphs are often unsettled and should not distract from the other information on the page, make sure that the contrast between graph and background is not too high (e.g. chose shades of gray instead of black if the background is white).

The concept can be enhanced by adding information about limits, highs, and lows:



On the following screenshot, you see a so called "dashboard" that makes heavy use of Sparklines:



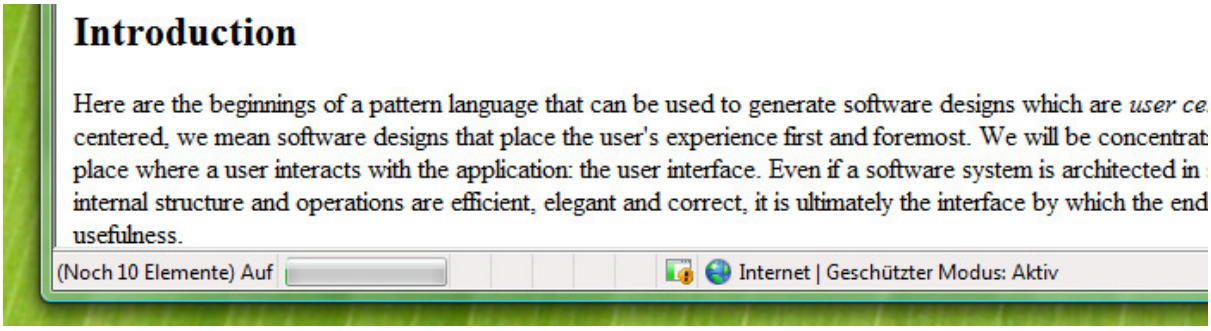
Solution

Use little, minimalistic graphs to display the trend of measurement parameters.

* * *

[1] Tufte, E. R. Beautiful Evidence. Graphis Pr., 2006.

07 STATUS PANEL



Context

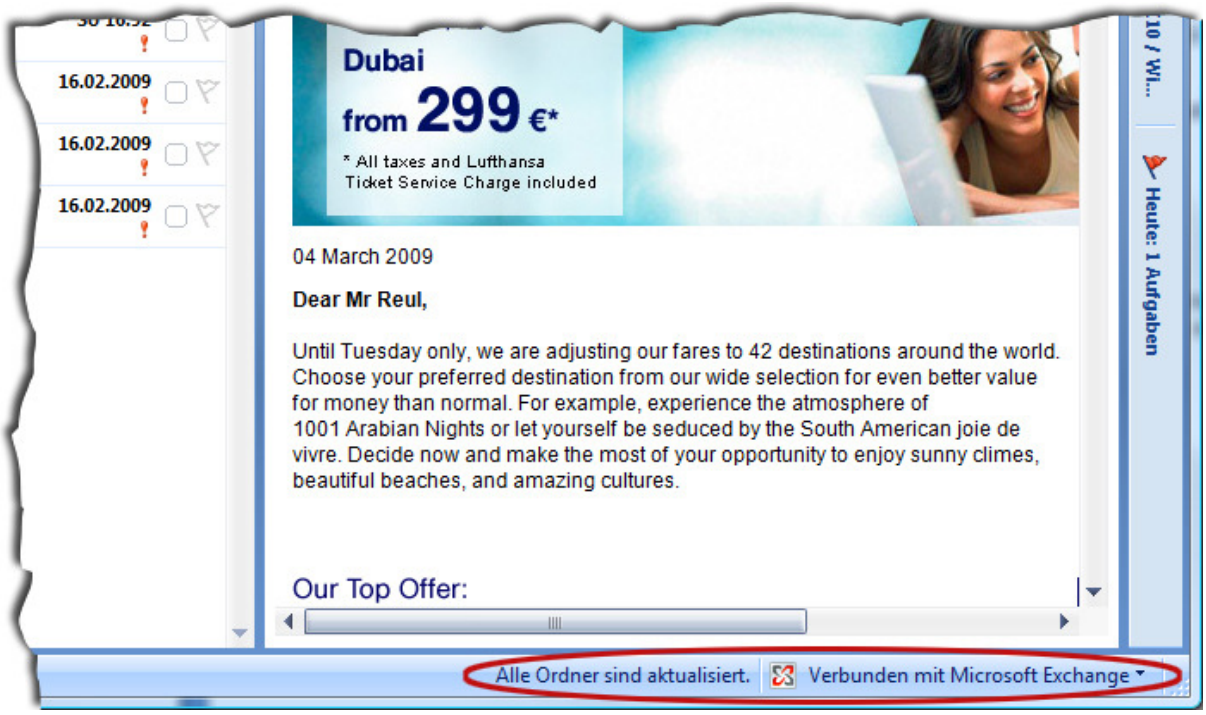
…you want to provide feedback to the user.

Problem

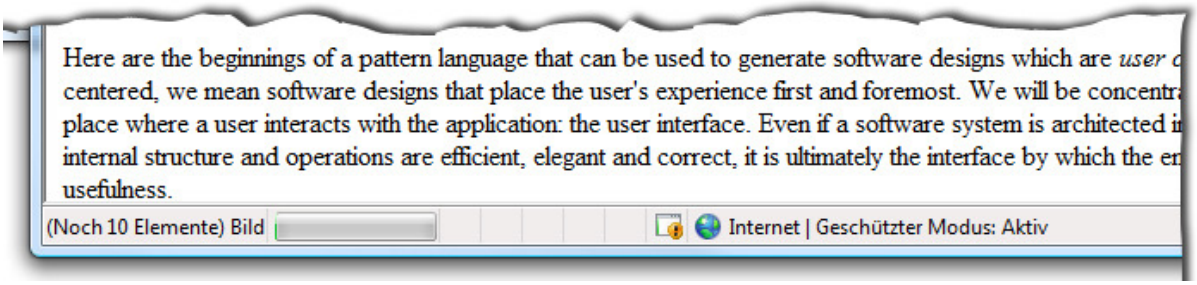
For every action the user takes, the system should provide some feedback that the program has accepted the command. In addition to that, the user should be notified when some change in the state of the system occurs. You may be tempted to deliver application warnings and information in popup windows or dialogs, but it can obscure the task window, the user is focusing on. The user should not be forced to interrupt his current task to respond to the informational feedback (this does not apply for safety-relevant information).

Examples

Microsoft Outlook 2007 uses a so called “status bar” to inform the user about the state of the system (“connected to exchange server”, ”synchronising folders”,…). In addition, Outlook 2007 allows users to customize what information is shown in the status area.



The Microsoft Internet Explorer uses a similar approach to inform the users whether a site was loaded correctly or not. The Status panel of the Internet Explorer also provides information about the security level of the visited website.



Visual Studio uses this pattern to display information about the build process.

Solution

Use a status Panel, rather than dialog boxes or pop up windows, to display information about the current state of the application. Place it near the main display area, so the user can view this information when she deems appropriate.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

08 Progress Indicator

Context

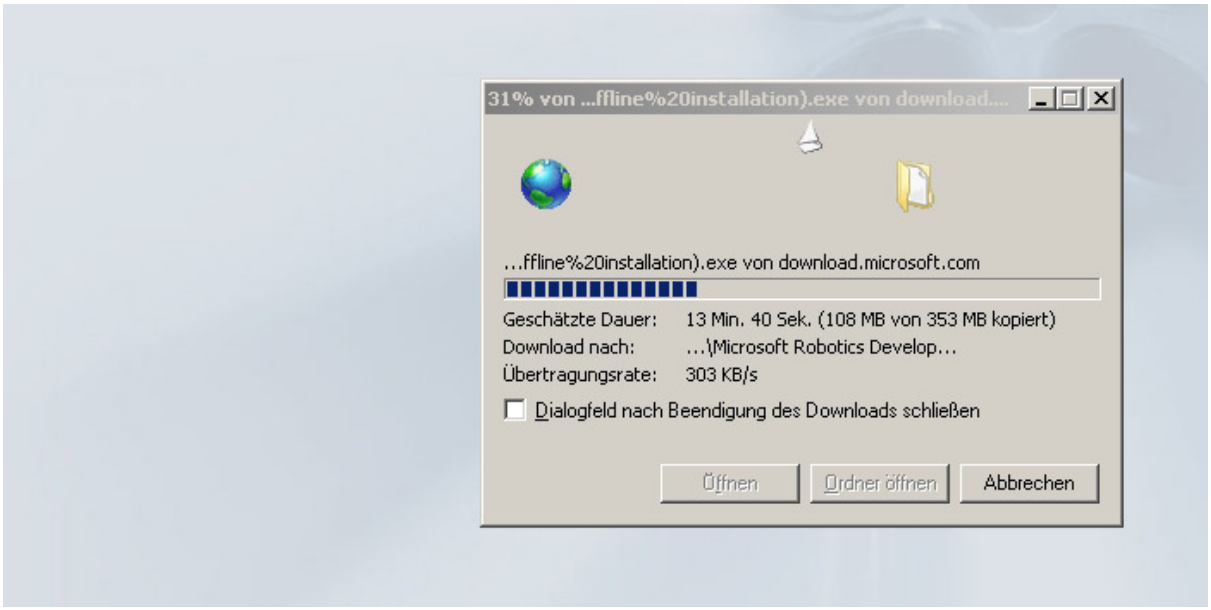
…your application performs time consuming operations (operations that last longer than a second).

Problem

If feedback does not come directly, the users get maybe antsy. Anxiety can build quickly when they don't know that something is going on, what is going on, how long it will take, and if they are unable to take control of their circumstances.

Examples

Many applications provide progress indicators to show that there is an operation in progress. Experiments show that if users see an indication that something is going on, they are much more patient, even if they have to wait longer than they would without a progress indicator. Microsoft Internet Explorer for example indicates the progress of a running download in form of a progress bar.



However, there exist situations in which the program can not tell how many percent or time of an operation is left. In this case, it is better to indicate only that something is in progress than making unreliable assumptions about the time that is left. Don't use progress bars that start again if they have reached the 100% mark. Users will be frustrated if the end is reached and the progress bar starts again. Firefox uses spinning wheels to indicate that a page is loading.

Solution

Display an indicator that shows the actual progress. If you can not give reliable information about the time that is left, indicate only that the system is working. Do not make unreliable assumptions about the length of the operation.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

09 Row Striping

Rank	Name	Citizenship	Age	Net Worth ($Bil)	Residence
1	Warren Buffett	United States	77	62.0	United States
2	Carlos Slim Helu & family	Mexico	68	60.0	Mexico
3	William Gates III	United States	52	58.0	United States
4	Lakshmi Mittal	India	57	45.0	United Kingdom
5	Mukesh Ambani	India	50	43.0	India
6	Anil Ambani	India	48	42.0	India
7	Ingvar Kamprad & family	Sweden	81	31.0	Switzerland
8	KP Singh	India	76	30.0	India

Context

…you have decided to display some of the information in form of a list or a table.

Problem

Large tables or lists suffer often the problem of being hard to read or scan. If the rows are placed close to each other, the user may have problems to identify to which row a certain cell belongs. Because screen size is often a constraining factor, making the gaps between the rows larger does not work in all situations.

Examples

Apple's music application "iTunes" uses tables to organize media files (such as music, videos, …). If the user possesses a large music collection, these tables can become very large and therefore hard to read. An example: the table that shows the entire music collection contains for each audio file the name of the artist, the track name, the album name, the duration,… The track name stands at the very left of the table – the duration of the track is displayed in the very right column of the table. If the user wants to know how long a certain piece of music will play, it can be very hard to detect which of the many cells with the duration indications belongs to the track of interest. Apple uses therefore a technique that is called "row striping": The backgrounds of two successive rows are coloured with different shades. This helps the user to follow a row from left to right and back again, without confusing the rows.

Titelname	Interpret	Dauer	Genre	Jahr
Believe	Cher	4:00	Pop	1999
Der Berg ruft	K 2	4:45		1994
The Best	Tina Turner	4:10		1989
Bette Davis Eyes	Kim Carnes	3:49	Oldies	1981
Bianca	Freddy Breck	2:33		1973
Big Big World	Emilia	3:23		1998
Big Fun	Inner City	3:26		1988
Big In Japan	Alphaville	3:53		1984
Billie Jean	Michael Jackson	4:54		1983
Bla Bla Bla	Gigi D'Agostino	3:14		2000
Black Is Black	Belle Epoque	5:09		1977
Black Or White	Michael Jackson	4:16		1991
Black Velvet	Alannah Myles	4:47		1990
Blame It On The Rain	Vanilli	4:19		1989
Blau blüht der Enzian		4:04		
Blockbuster		3:14	Oldies	
Blue (Da Ba Dee)			Pop	
Blue Monday				

Many Websites uses the same technique. You see here an example of www.aida.de:

Tag	Hafen	Land/Insel	Ankunft	Abfahrt	
09.11.2009 Montag	Bangkok	Thailand	-	-	Details zum Hafen
10.11.2009 Dienstag	Bangkok	Thailand	-	20:00 Uhr	Details zum Hafen
11.11.2009 Mittwoch	Koh Samui	Thailand	10:00 Uhr	18:00 Uhr	Details zum Hafen
14.11.2009 Samstag	Penang	Malaysia	8:00 Uhr	18:00 Uhr	Details zum Hafen
15.11.2009 Sonntag	Kuala Lumpur	Malaysia	7:00 Uhr	18:00 Uhr	Details zum Hafen
16.11.2009 Montag	Singapur	Singapur	9:00 Uhr	-	Details zum Hafen
17.11.2009 Dienstag	Singapur	Singapur	-	13:00 Uhr	Details zum Hafen
19.11.2009 Donnerstag	Muara	Brunei	8:00 Uhr	17:00 Uhr	Details zum Hafen

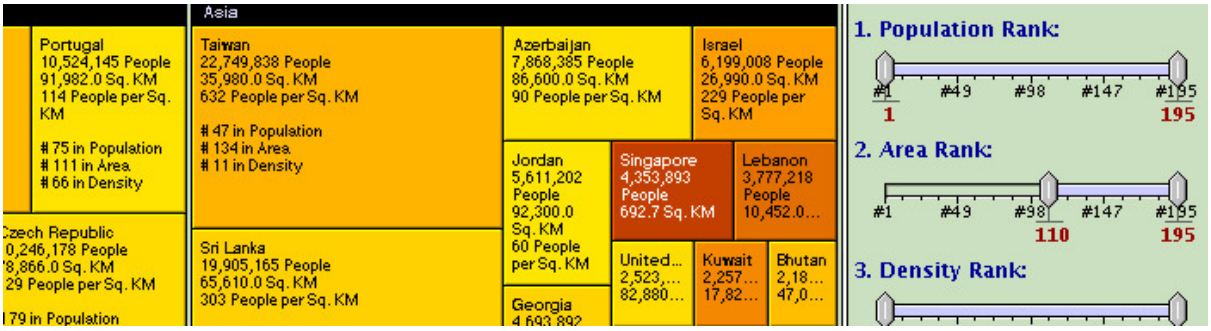
Solution

Pick a pair of quiet, low saturation colors that are similar in value, but not identical (One needs to be a little bit darker than the other). Use these colors alternately to colorize the background of the table rows. This pattern virtually eliminates the need for horizontal lines between the rows.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

10 Dynamic Queries



Context

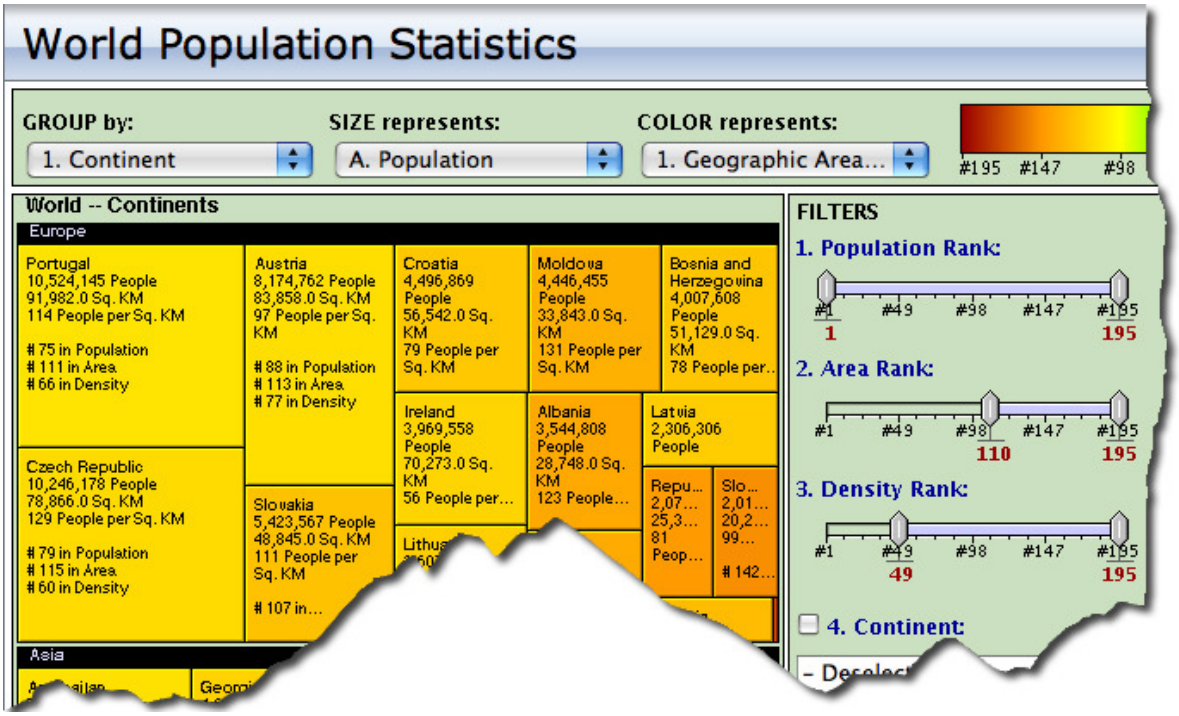
...you show the user a large, multi-dimensional data set, of any shape, with any presentation. User need to filter out some of the data to accomplish a task.

Problem

Exploring large data sets can be very tedious. In many cases the user is looking for data items with specific attributes. Browsing through the entire list or table isn't here very efficient. If the user knows the attributes, he is looking for, why not let the computer search the data?

Examples

The first example for this pattern is from The Hive Group and shows a treemap of the world's population. The panel on the right lets the user filter the data with three interval sliders by Population Rank, Area Rank, and Density Rank. Each time, a control's value changes, the treemap is refreshed. In this example, the treemap on the left shows only those countries that are in the area ranking between place 110 and 195 and are between place 49 and 195 concerning the density ranking.



Ebay uses Dynamic Queries for the filtering of search results. By clicking on the corresponding checkboxes, the user can filter the serach results by condition (new/used) and other properties (buying format,...).

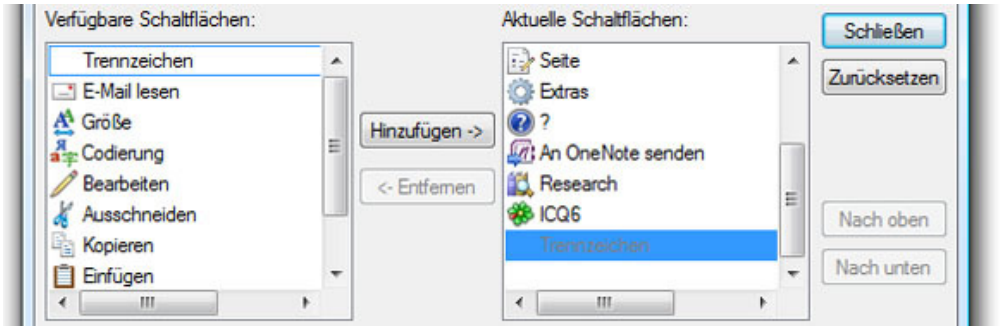
Solution

Provide ways to filter the data set immediately and interactively. Employ easy-to-use standard controls, such as sliders and checkbox, to define which parts of the data set get shown. As soon as the user adjusts those controls, the results appear on the data display.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

11 List Builder



Context

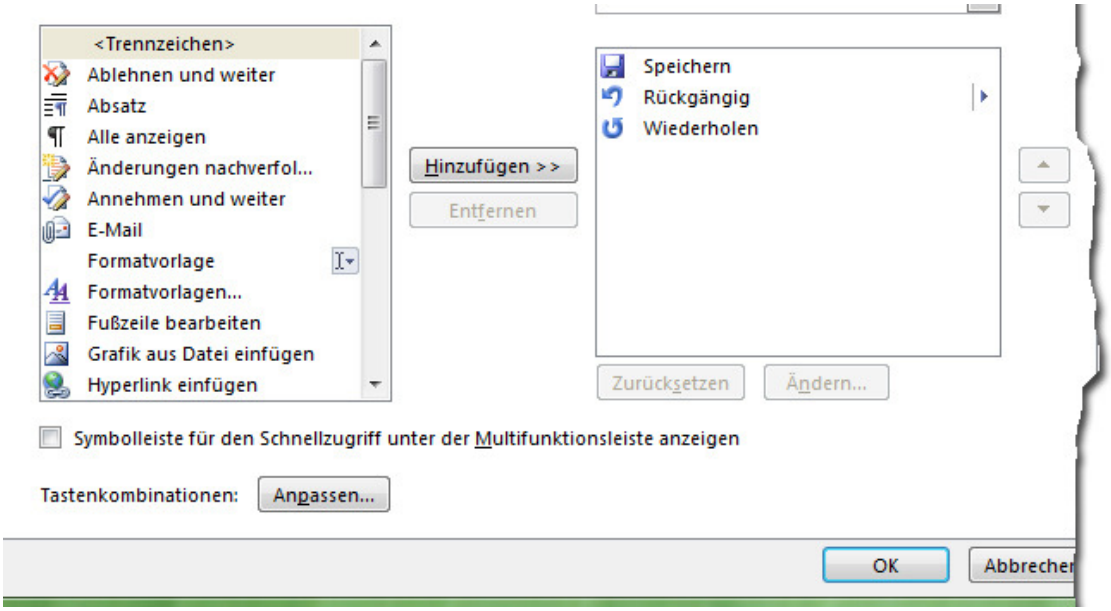
…the user needs to create a list or set based on items that already exist in another list.

Problem

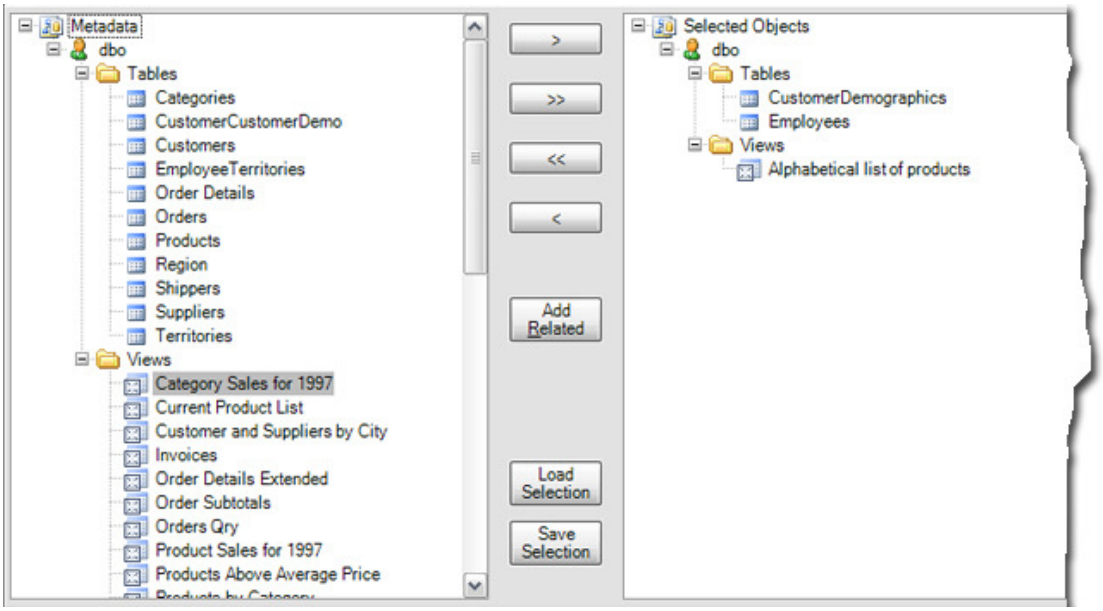
A simple solution might be a single list with checkboxes. This would solve the "select a subset" problem. But if you have a very large source list, a list of checkboxes does not scale. The user can not easily see what has been checked, and thus not get a clear picture of what she has selected.

Examples

Office 2007 enables customization of the Quick Access Toolbar by using two lists with buttons that let users move items from one to the other. This option provides an easier way to see the selected items but it consumes more space in the screen. The first list has the available options and the second list the selected ones. Besides the two buttons, the applications also allows to move items between the two lists by providing a drag and drop functionality.



The following screenshot from DeKlarit shows how two trees can be used when the data is hierarchical. It takes more screen real estate than a single tree with checkboxes, but it lets people easily see the selected items.



Solution

Show two lists on the same page. One list contains the available items. The other list will contain the selected items (and should be empty at the beginning). Let the user move items between the two lists.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

12 New Item Row

Context

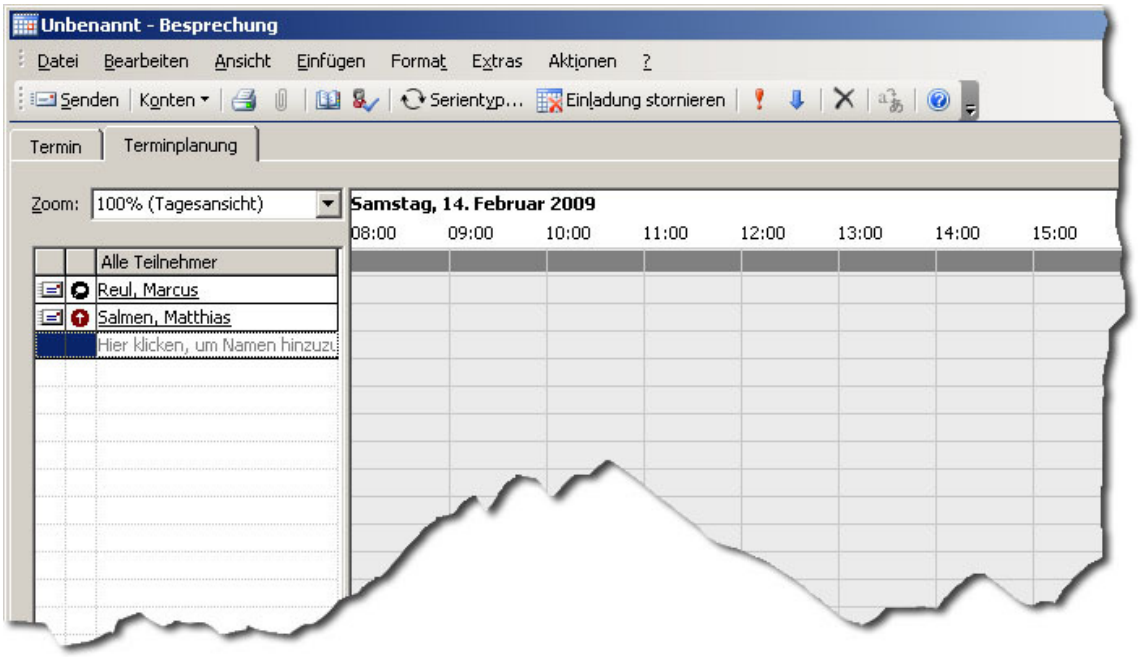
…your user interface design contains a table or list to which - at some point - the user needs to add new items.

Problem

Separating the place where new items will be created from the place where they actually "live" is not natural. Additionally, adding a new dialogue for creating items makes the user interface (and the interaction with it) more complex.

Examples

Microsoft Outlook uses a list to manage the participants of a meeting. To add an additional participant to the meeting, the user can click in the first empty row of the table and start typing the name of the desired participant. Microsoft Outlook matches the ongoing input with the internal address book and indicates whether the input is valid or not. If the input matches an address book entry, outlook adds the participant to the meeting.



Solution

Use the last row in a table or a list to create a new item in place. Each column in the table (if it´s a multicolumn table) should be editable, thus letting the user set up the values of that item. The cells could have text fields in them, dropdown lists, or whatever else is necessary to set the values quickly and precise.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

13 JUMP TO ITEM

Context

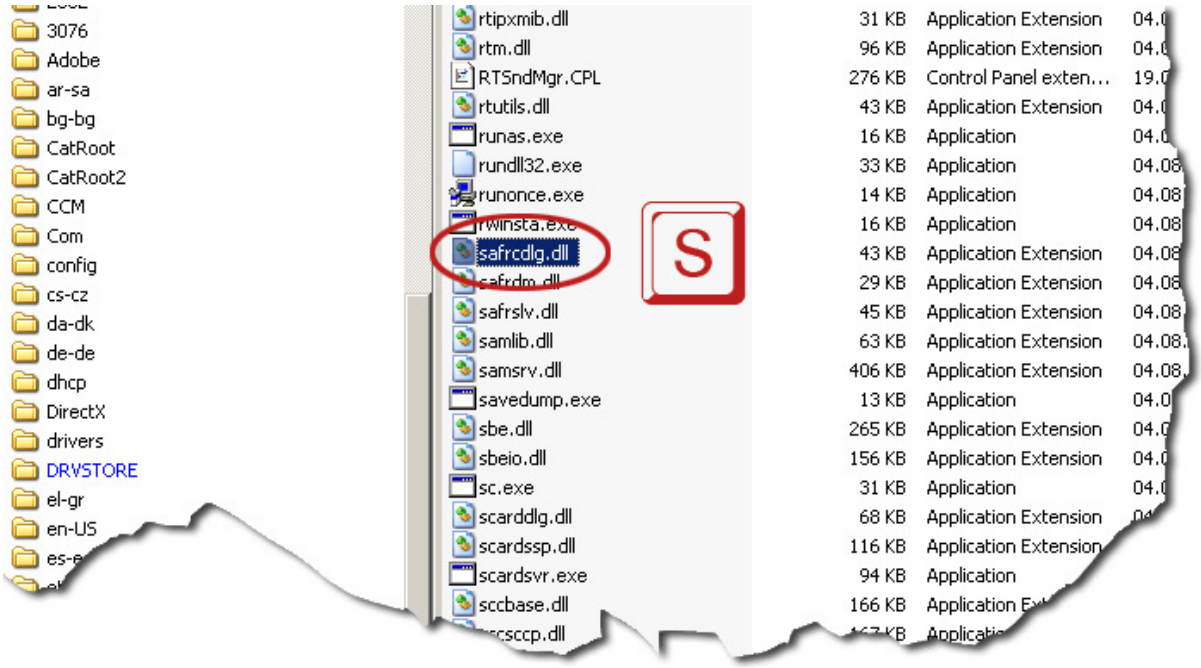
…your application uses a scrolling list, table, dropdown, combo box, or tree to present a long set of items. These items are usually sorted, either alphabetically or numerically. At some point, the user wants to find one particular item quickly and accurate.

Problem

Humans are not good at scanning down long lists of words or numbers. If the user knows exactly what he is looking for and he doesn't find it, scrolling through long lists can be boring and very frustrating.

Examples

In Windows Explorer, users can type the first letter of the name from a file, they are looking for. The file list jumps directly to the first file that begins with the entered letter and selects it. By pressing the same letter a second time, the list jumps to the next file whose name begins with the entered letter.



Many mobile phones feature a similar mechanism for browsing through their contact lists. When the user types the first letter of a contact's name, the contact list jumps to the first item that matches what the user typed. The list automatically scrolls to the item and selects it. As the user types more characters, the list jumps to the first item that matches the entire input.

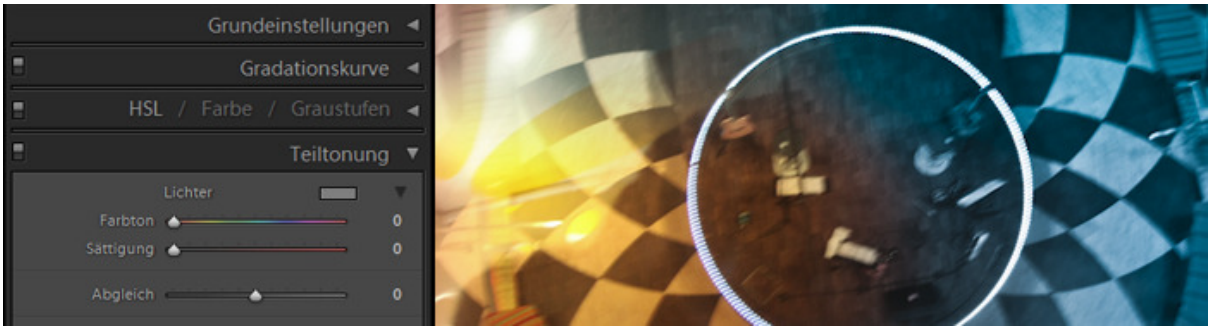
Solution

When the user types the name of an item, jump straight to that item and select it.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

14 Closable Panels



Context

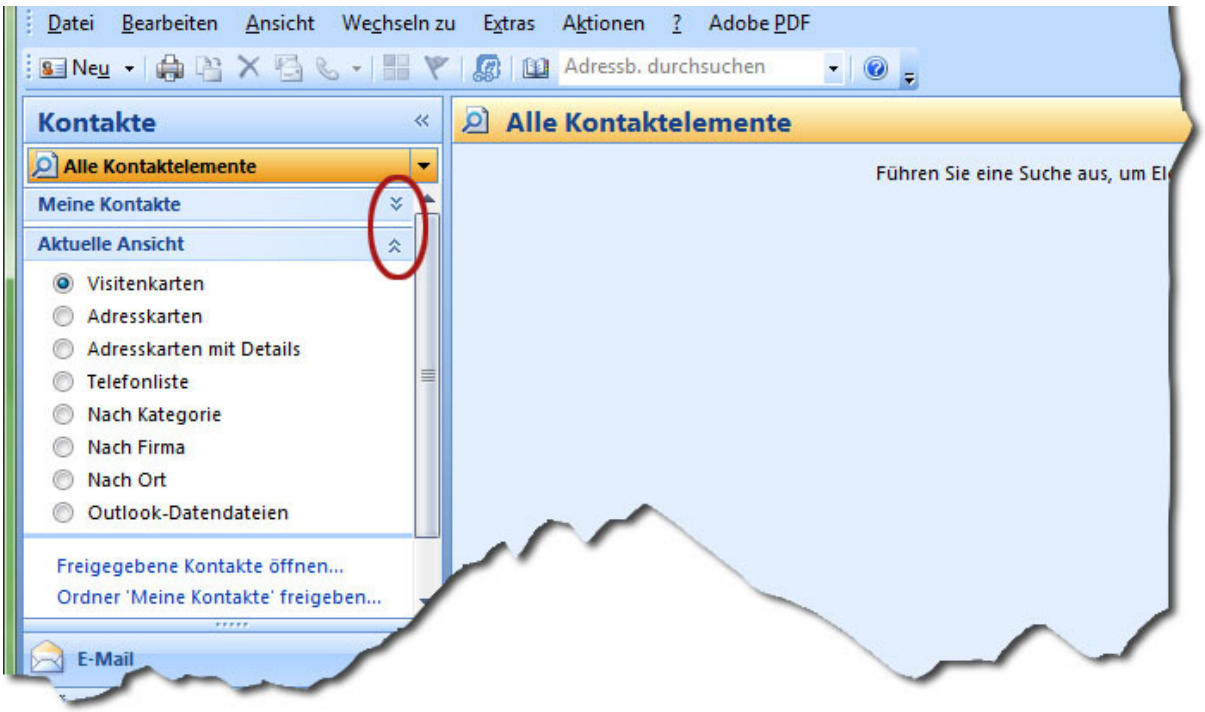
...you have a lot of things that you need to be readily available, but you don't want to (or even can't) show them all the time.

Problem

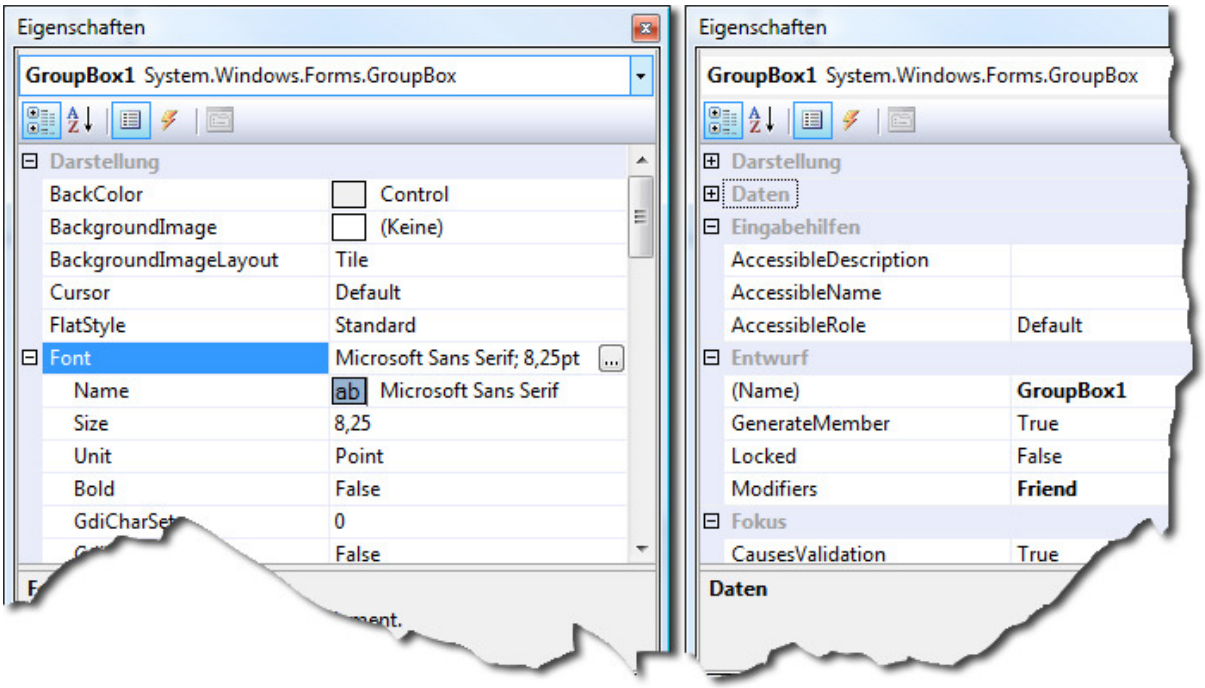
In some cases, there are so many actions possible, that (because of screen space limitations) you can't put all the controls on the screen at the same time. You can also imagine cases, where you have the space to display all these controls but you don't want to distract from other (more important) functions. However, you need to find a method to make all these functions available quick and comfortable.

Examples

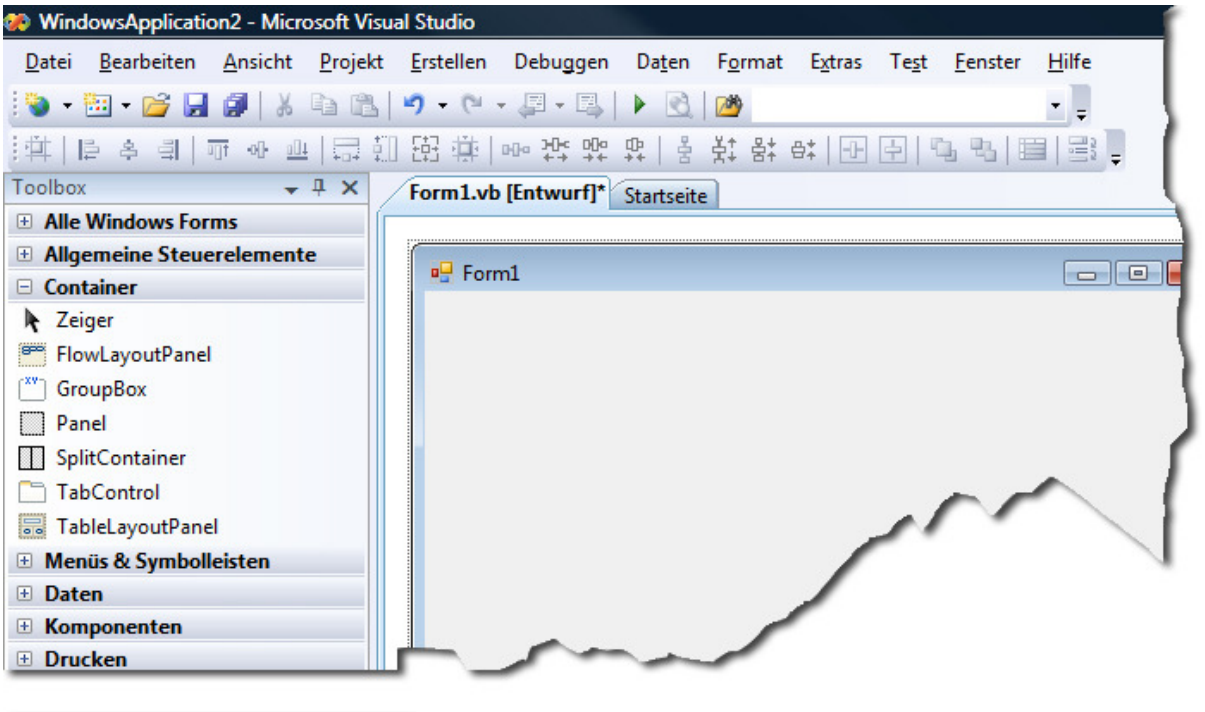
Microsoft Outlook 2007 uses closeable panels inside each area that provides tools to control the main view. The chevrons at the top of this collection of panels also enable you to close the panel group, which is very handy for smaller screens.



Microsoft Visual Studio's property sheets use Closable Panels to categorize its numerous properties by topic. A user who isn't interested in the Appearance or Font properties, for instance, can close them up while she concentrates on the other properties. Instead of a triangle, Visual Studio uses pluses-in-squares, just like the Windows treeview control. Even though this isn't a treeview, it takes advantage of what users already know: you click on the plus to open something, and click on the minus to close it.



Visual Studio uses a similar approach to organize the user interface elements in its user interface builder:



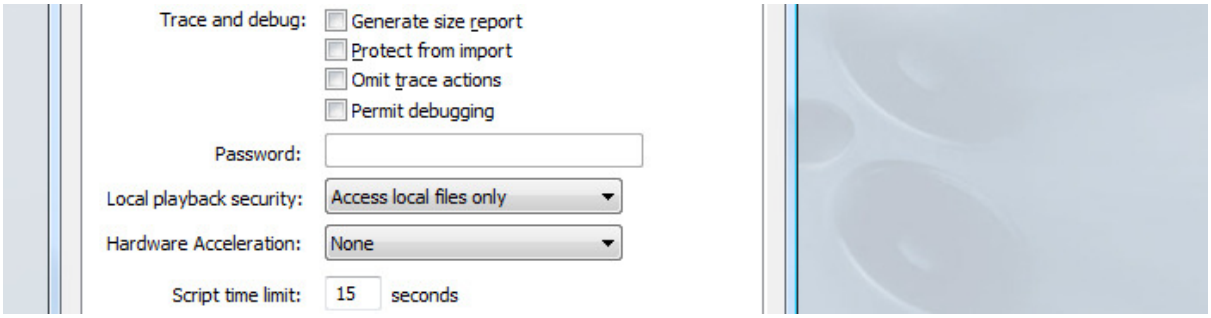
Solution

Put selections of content onto separate panels. Let the user open and close each of them separately from the others.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

15 Right Left Alignment



Context

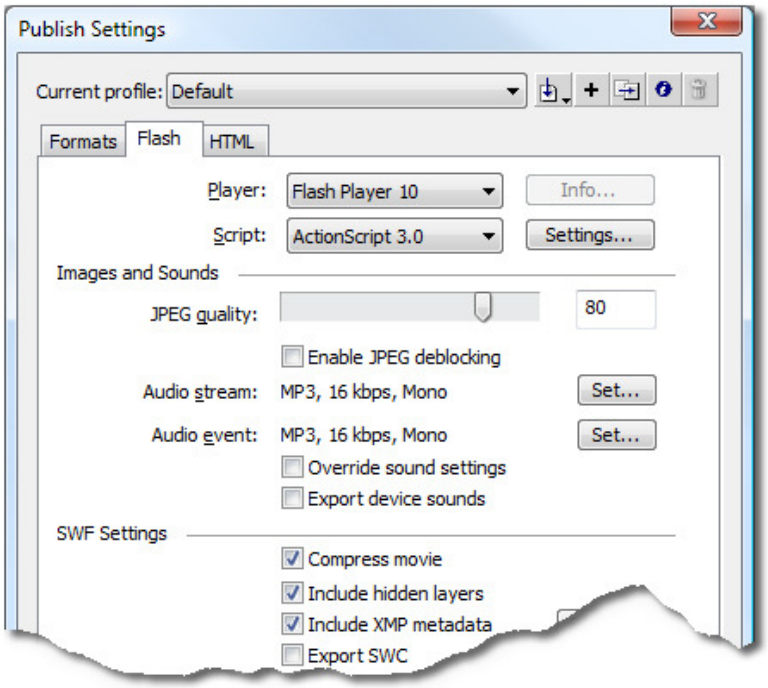
…you are laying out a form, or any other set of items that have text labels in front of them. This could also apply to the internal structure of tables, or any other two-column structure in which rows should be read left-to-right.

Problem

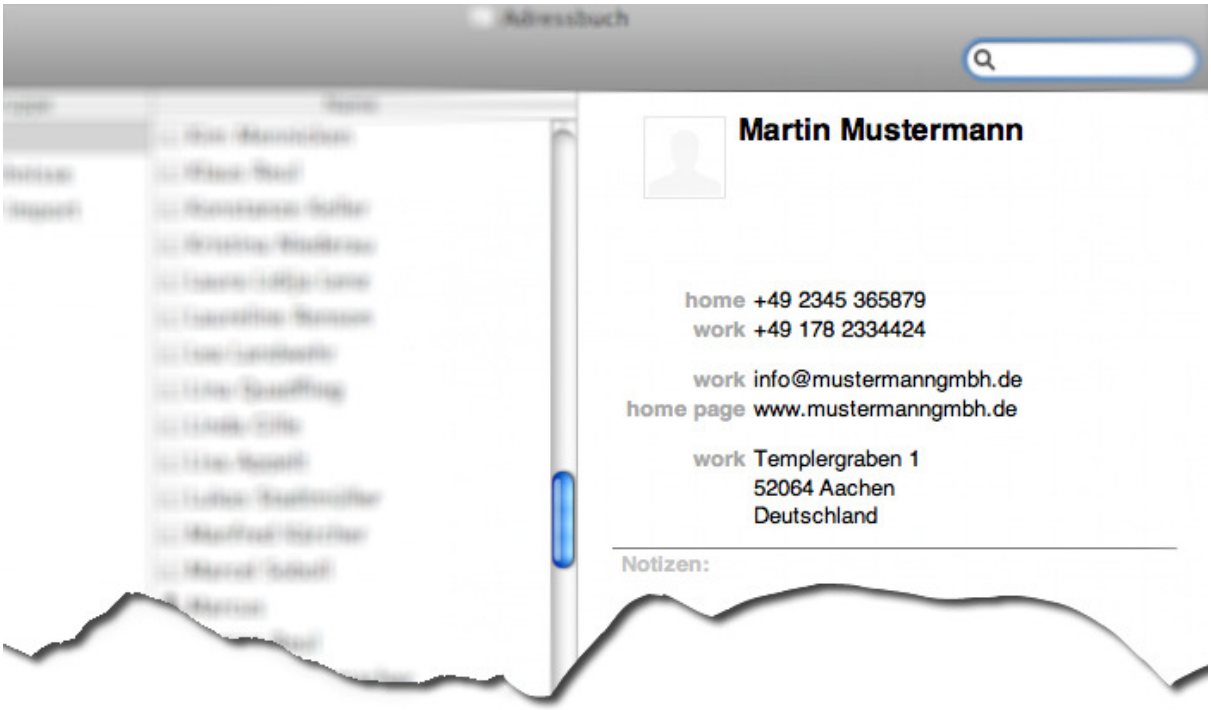
When you put text right next to the thing it labels, you form a strong perceptual grouping of that pair – much more so than if they were separated by a large amount of space. However, if you align variable-length labels along their left sides, the short labels won't be close to their controls.

Examples

The first example comes from Adobe Flash. In this form, the labels are aligned along their right sides. This layout emphasises the strong relation between labels and corresponding controls (text-boxes, radio-buttons, dropdown-menus,…).



This pattern also works with layouts that have no input controls at all. This Mac OS X address-book entry has very little whitespace between the two columns, but the difference in color helps to separate them visually. Notice that the label “home page” is much longer than the others; this would have made a lefthand label alignment less pleasing to the eye, and harder to read.



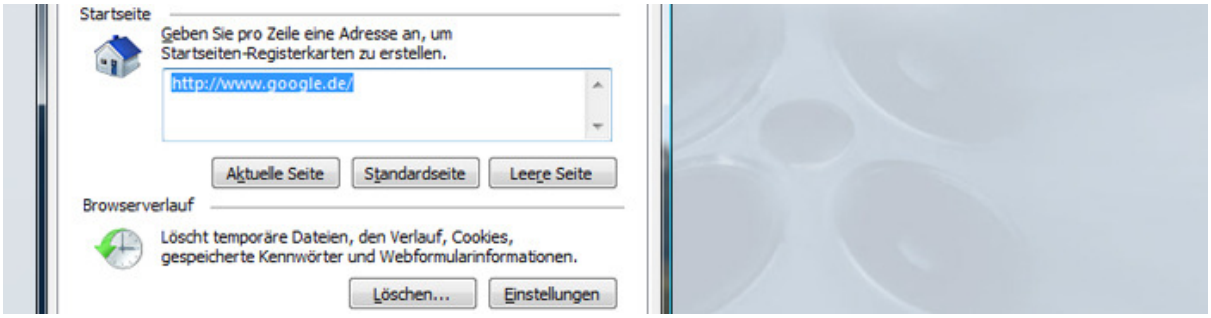
Solution

When designing a two-column form or table, right-align the labels on the left, and left-align the items on the right.

* * *

[1] Tidwell, J. Designing Interfaces. O’Reilly, Sebastopol, CA, USA, 2006.

16 Button Groups



Context

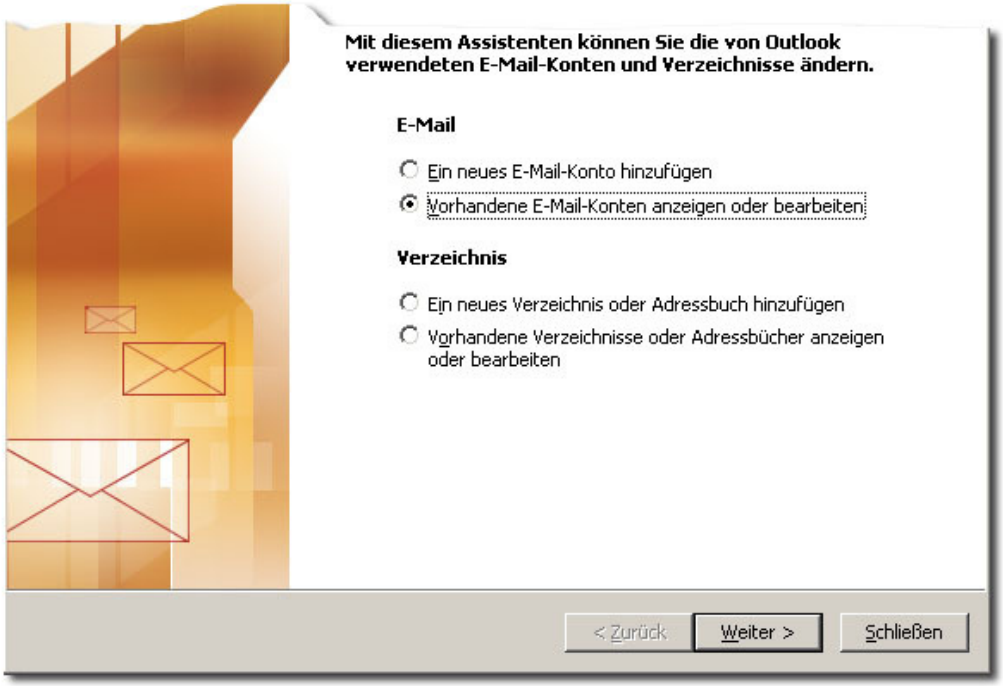
…you want to present a small number of related actions inside a window or a dialogue box. Each action has his dedicated button. You are now looking for a way to arrange these buttons.

Problem

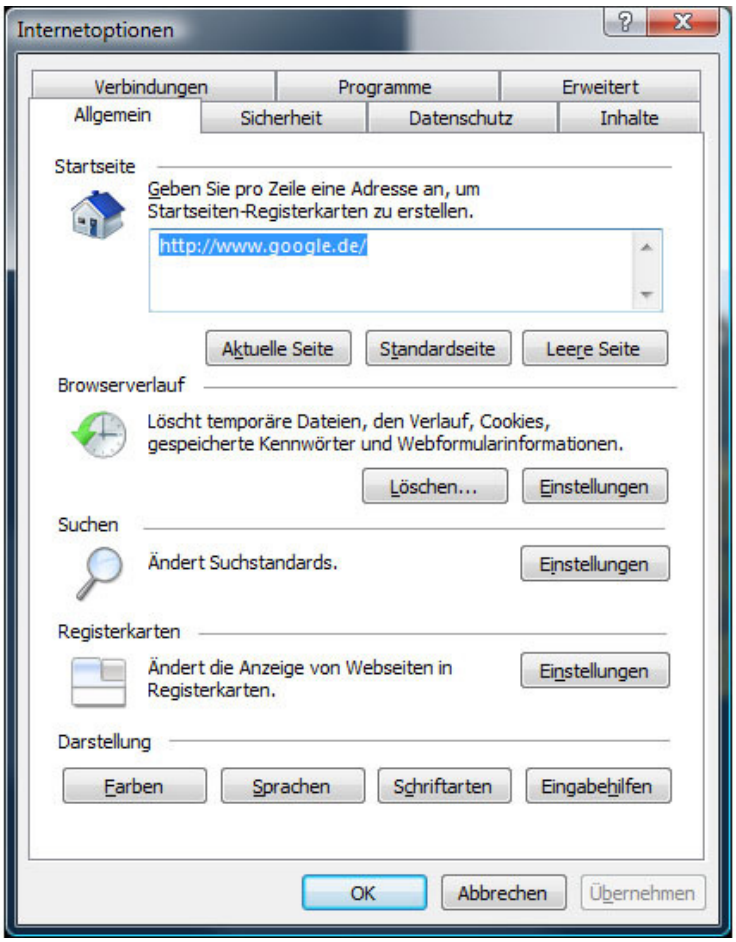
Unordered buttons that are scattered around the screen makes the user interface confusing. If the user interface must provide many buttons and they aren't arranged in a meaningful way, it can be very hard for the user to find the button, he is looking for.

Examples

This example shows the Microsoft Outlook Wizard for creating and managing e-mail accounts. The "back"- and the "next"-button are arranged together in one group. The proximity of these two buttons indicates that there is a close relation between the underlying functions (from user's perspective). The little gap between this button group and the "close"-button indicates that the underlying functions are not related.



The next example for this pattern is a dialog box from Microsoft's Internet Explorer that has several button groups. All buttons from one group address related actions (concerning the home page, browsing history, search, tabs, and display – there is also an additional button group that controls the main options of this window (OK, cancel, and apply).



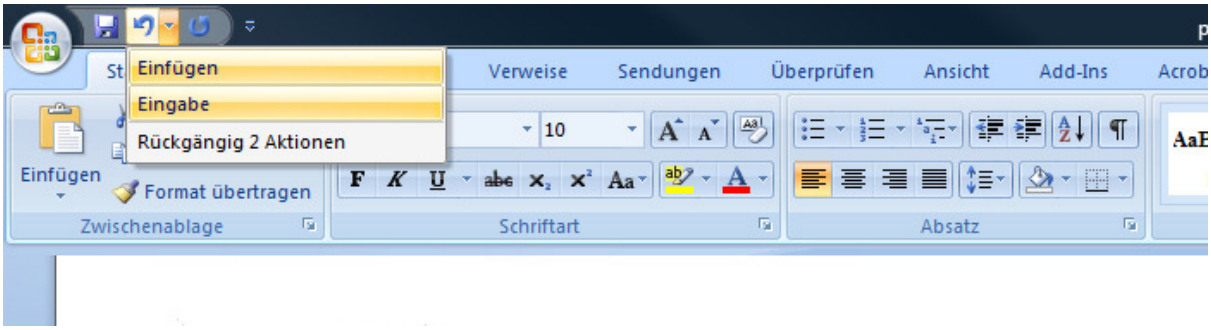
Solution

Group related actions as a small cluster of buttons, aligned either horizontally or vertically. Do not put more than three or four buttons in one group. Create several groups if there are more actions. All buttons in the group should have the same height and width (unless the label length varies widely).

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

17 UNDO



Context

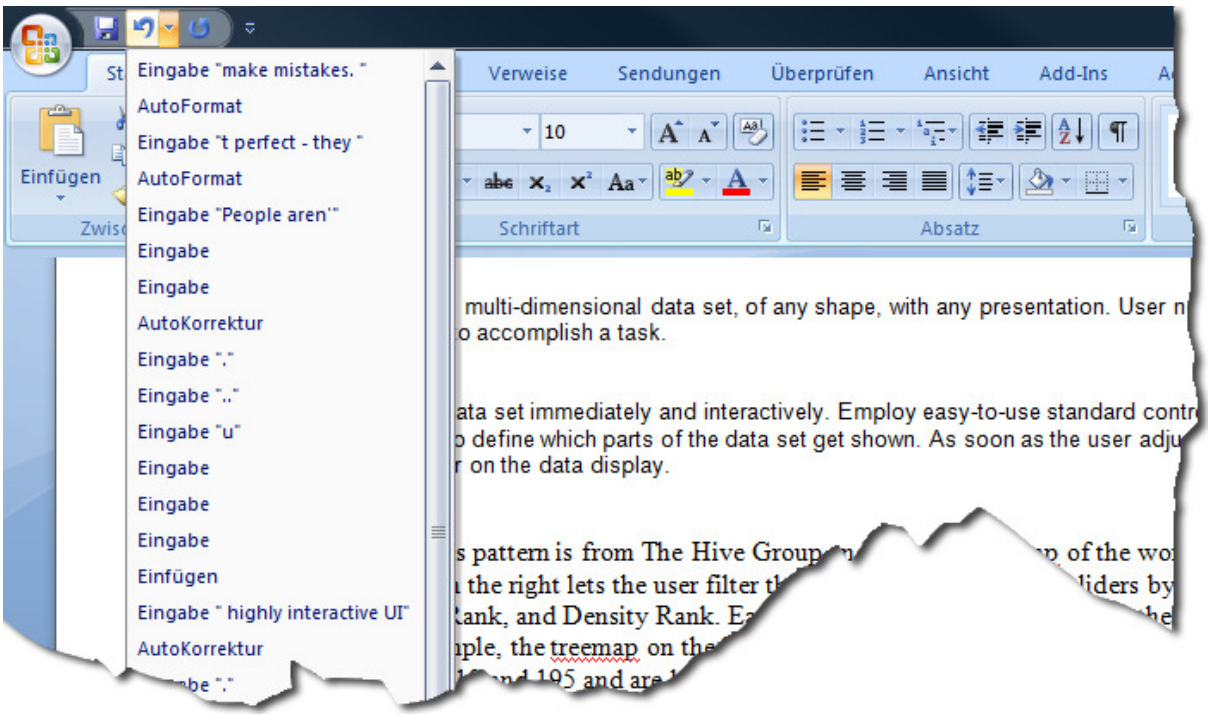
…Undo is one of those nearly ubiquitous patterns—you should probably default to supporting it unless you have constraints that prevent you from doing providing it.

Problem

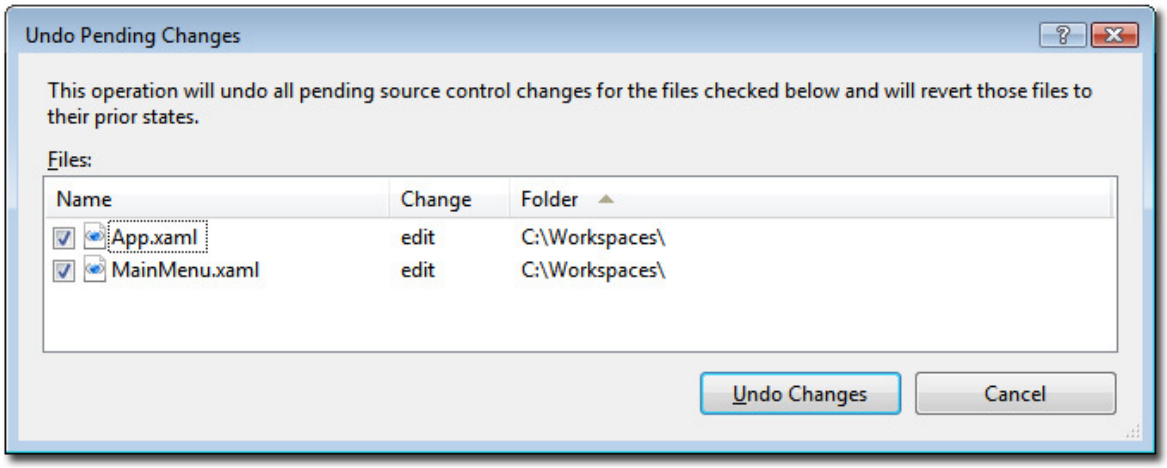
People aren't perfect – they make mistakes.

Examples

The primary example is from Microsoft Office 2007. The undo uses a Drop Down Button—clicking the icon will undo the latest action, but people can use the drop down part of it to choose to rollback to an even earlier action in the history. Note how they try, when possible, to make the items intelligible (e.g., Typing "thing typed"), and it is usually word or phrase-based—not letter based. It tracks a deep history and uses a scrollable area to let people explore the history—for instance.



Here you see Visual Studio 2008's Undo Pending Changes dialog. This is an example of file-based versioning undo - people can revert to the last state checked in on the server.



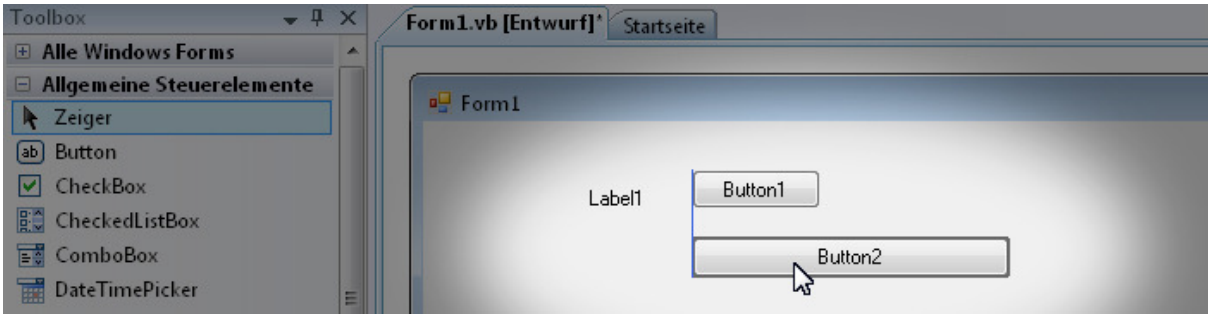
Solution

Provide a way to easily reverse a series of actions performed by the user.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

18 MAGNETISM



Context

…you have scenarios that involve positioning of objects on a screen. This often happens in graphic editors, of course, but it's also common in window managers and desktop framework, in which a user needs to move windows and palettes, for example.

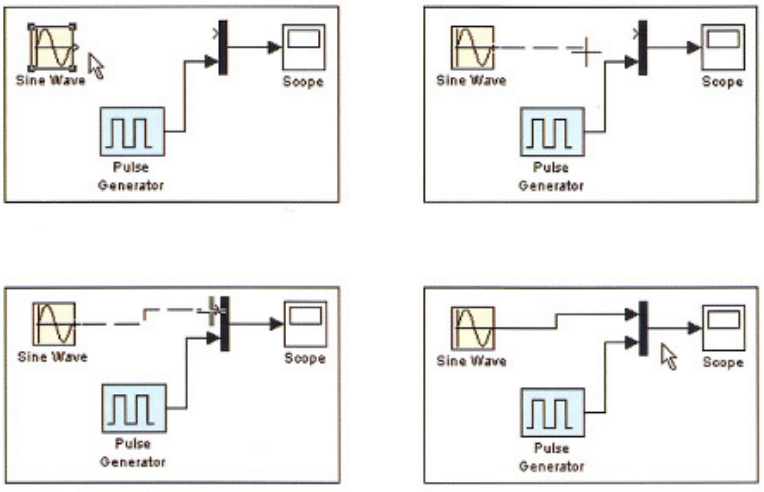
Problem

Your target audience would want to precisely position items but this is in many cases challenging or even frustrating to do without help.

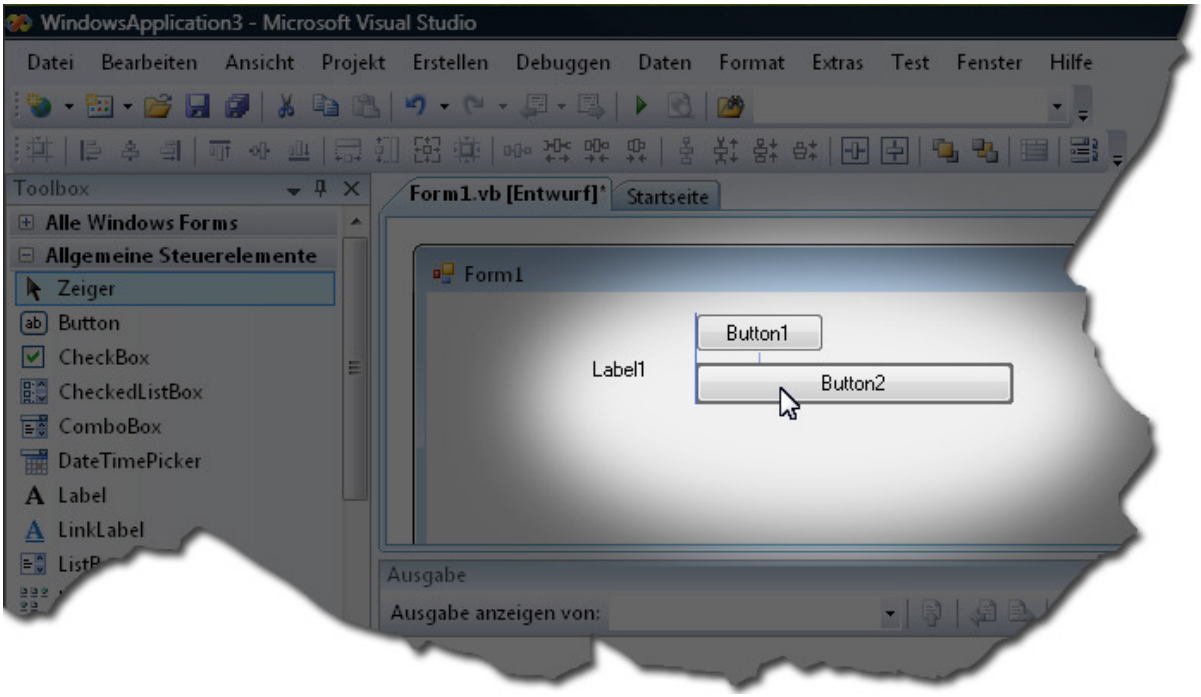
Examples

Diagram builder and visual programming environments sometimes use a mechanism called "Magnetism" to help the user link together a diagram. Here, you can use the Simulink application to put together a miniature signal-generator simulation. The user needs to connect the output port of the Sine Wave source to an input port of a multiplexer, shown as dark vertical bar.

Without Magnetism, this task would require the user to position the mouse on a very tiny target, click down on the mouse button, and then drag and release the connection to another very tiny target. However, the source output port actually is bigger than the few pixels shown. And as the user drags the connection – illustrated by a dotted line – towards the multiplexer's input port, the connection "snaps" into the input port as the mouse approaches within ten pixels or so. In the last picture, the user has released the mouse button, finishing the connection.



Visual Studio 2005 and up uses magnetism to help users align controls in a form. As in the example below, when you drop a control in a form near the left border of the control above, it tries to align the left borders. If it's close to the control above, it will try to keep a margin between both controls. Without magnetism, it requires a lot of work to have to align all the controls.



Solution

Make the objects "magnetic" to the things a user positions them against. When the user drags an object very near one of these things, it should stick.

* * *

[1] Tidwell, J. Designing Interfaces. O'Reilly, Sebastopol, CA, USA, 2006.

Anhang B

Workshop Fragebogen

Fragebogen zum HCI Design Pattern Workshop

Vielen Dank für Ihre Teilnahme am heutigen Workshop! Bitte füllen Sie zum Abschluss noch folgenden Fragebogen aus. Lesen Sie sich die Fragen aufmerksam durch und beantworten Sie sie gewissenhaft und ehrlich.

Vorgehensweise zur Beantwortung der Fragen:

Kreuzen Sie die zutreffende Antwort an. Pro Frage kann nur ein Kreuz gesetzt werden. Wenn Sie aus Versehen ein Kreuz falsch gesetzt haben, malen Sie bitte die runde Fläche komplett aus und kreuzen Sie die gewünschte Antwort an.

Mit dem Ausfüllen des Fragebogens erkläre ich mich einverstanden, dass die (anonymisierten) Ergebnisse für wissenschaftliche Zwecke weiterverwendet werden.

Wie groß ist der Anteil Ihrer Arbeitszeit, in dem Sie an der Implementierung (Programmierung) von User Interfaces arbeiten?

○ Weniger als 20% ○ Ca. 21%-40% ○ Ca. 41%-60% ○ Ca. 61%-80% ○ Mehr als 80%

Wie groß ist der Anteil Ihrer Arbeitszeit, in dem Sie an Spezifikationen von User Interfaces arbeiten?

○ Weniger als 20% ○ Ca. 21%-40% ○ Ca. 41%-60% ○ Ca. 61%-80% ○ Mehr als 80%

Ich habe vor dem heutigen Tag schon einmal einen Prototyp für ein User Interface erstellt

○ Nein ○ Ja

Ich kannte das Konzept der HCI Design Patterns bereits vor dem Workshop

○ Nein ○ Ja

Ich habe HCI Design Patterns bereits vor dem Workshop angewendet

○ Nein ○ Ja

Der Inhalt der ausgehändigten Design Patterns war verständlich

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Schätzen Sie die Übersichtlichkeit der ausgehändigten Patterns ab

Unübersichtlich	Eher unübersichtlich	Unentschieden	Eher übersichtlich	Übersichtlich
◯	◯	◯	◯	◯

Das Inhaltsverzeichnis hat mir geholfen, mich in der Pattern Sammlung zurecht zu finden

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die inhaltliche Beschreibung der Patterns ist …

nicht ausführlich genug	eher nicht ausführlich genug	genau richtig	eher zu ausführlich	zu ausführlich
◯	◯	◯	◯	◯

Die ausgehändigten Patterns passen inhaltlich zum Thema „Prüfstandsautomatisierungssysteme“

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Ich kann mir vorstellen die Ideen aus den ausgehändigten Patterns in Softwareentwicklungsprojekten der FEV anzuwenden

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die Patterns haben mich auf Probleme aufmerksam gemacht, die mir vorher nicht bewusst waren

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die Patterns beinhalten interessante Lösungsvorschläge, die ich noch nicht kannte

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die Patterns haben mich auf Lösungen aufmerksam gemacht, die ich zwar schon kannte, aber an die ich ohne das Lesen der Patterns nicht gedacht hätte

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die Beispiele haben zum Verständnis der Patterns beigetragen

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Ich habe beim Erstellen des zweiten Prototyps mindestens eins der ausgehändigten Patterns berücksichtigt

Nein	Ja
◯	◯

Wenn Sie die vorige Frage mit „Ja“, beantwortet haben: Welche Patterns haben Sie in Ihrem Design berücksichtigt (Angeben der Nummer reicht)? Erläutern Sie kurz wie die ausgewählten Patterns beim Design geholfen haben.

Ich habe mich beim Erstellen des zweiten Prototyps bewusst gegen eins oder mehrere der Patterns entschieden (Ich habe mich bewusst für eine andere Lösung entschieden, obwohl das Pattern vom Kontext her gepasst hätte)

◯ Nein ◯ Ja

Wenn Sie die vorige Frage mit „Ja“, beantwortet haben: Gegen welche Pattern haben Sie sich entschieden und warum?

Welche der ausgehändigten Patterns wären für die Aufgabe von heute morgen interessant gewesen (Angeben der Nummer des Pattern reicht)? Warum?

Die Patterns haben mir Anreize gegeben, wie man die Benutzbarkeit einer Software, an deren Entwicklung ich beteiligt bin/war, verbessern könnte

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Ich kann mir vorstellen, in Zukunft eines oder mehrere der Patterns in einem Entwicklungsprojekt einfließen zu lassen

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die Patterns würden bei der Anforderungsanalyse für das User Interface eines zukünftigen Entwicklungsprojektes helfen

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die Patterns würden bei der Spezifikation des User Interfaces eines zukünftigen Entwicklungsprojektes helfen.

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Die Patterns würden bei der Implementierung des User Interfaces eines zukünftigen Entwicklungsprojekts helfen.

Dem stimme ich gar nicht zu	Dem stimme ich eher nicht zu	Unentschieden	Dem stimme ich eher zu	Dem stimme ich voll zu
◯	◯	◯	◯	◯

Platz für allgemeine Anmerkungen zum Workshop und den Design Patterns (zum Beispiel: Schwierigkeiten bei den Design Aufgaben, Probleme die beim Anwenden der Patterns aufgetreten sind, Verbesserungsvorschläge zu den Patterns, …)

Vielen Dank für Ihre Hilfe ☺

Anhang C

Workshop Aufgaben

1 Grundziel

Ziel der Aufgabe ist es ein Design eines User Interfaces für die im Folgenden spezifizierte Softwareanwendung zu entwickeln. Unter dem Design eines User Interfaces versteht man die Entscheidungen bezüglich der Wahl der User Interface Elemente (Fenster, Textbox, Checkbox, Slider,…), ihre Ausrichtung und die daraus entstehenden Möglichkeiten der Benutzung (Interaktionsablauf).

2 Prototyp

Bei einem Prototyp kommt es darauf an, den richtigen Detaillierungsgrad zu wählen. Einerseits muss der Prototyp klar machen, wie das User Interface aufgebaut ist und wie es sich bei der Interaktion (Benutzer-Ein- und Ausgaben) verhält. Andererseits sollte man sich beim Erstellen des Prototyps nicht zu sehr an technischen- und Implementierungsdetails aufhalten. Es gibt eine unzählige Anzahl von Prototyp-Arten. In diesem Workshop ist es den Teilnehmern freigestellt, welche Art Prototyp sie erstellen. Jedoch sollte der Prototyp so konzipiert sein, dass ein Außenstehender anhand dieser Vorlage imstande wäre, das User Interface entsprechend zu implementieren. Dabei kommt es auf das Layout der User Interface Elemente (Widgets) und deren Verhalten an.

3 Arten von Prototypen

Eine Möglichkeit ist die Benutzung des in Visual Studio eingebauten Interface Builders. Der Interface Builder ermöglicht es das Grundlayout des User Interfaces durch einfaches Drag-and-Drop zu erstellen. Dazu werden die User Interface Elemente aus der Toolbox in das gewünschte Fenster gezogen. In der Projektansicht können neue Fenster und Dialoge angelegt werden. Das Grund-Interaktionsverhalten kann mit einigen Zeilen Programmcode simuliert werden (z.B. ButtonPress macht neues Fenster sichtbar). Jedoch kann es sein, dass Visual Studio die gewünschten User Interface Elemente nicht bereitstellt oder die Implementierung der Interaktion einen größeren Aufwand darstellen würde (z.B. Drag and Drop Verhalten, Elemente zur Darstellung von Graphen, …). Eine Möglichkeit dieses Problem zu lösen ist einen Screenshot vom in VisualStudio erstellten Grunddesign zu erstellen, diesen auszudrucken und die fehlenden Elemente per Stift einzuzeichnen. Stellen, die besondere Interaktionsmöglichkeiten erlauben, können ebenfalls mit Filzstiften entsprechend markiert werden. Verbindungen zwischen einzelnen Fenstern (Blättern) können zum Beispiel durch Nummerierung kenntlich gemacht werden.

Eine weitere Art von Prototypen sind Papier-Prototypen. Hier wird das gesamte User Interface auf Papier erstellt. Jedes Fenster und jede Dialogbox werden auf einem separaten Blatt Papier spezifiziert. Links und anderes Interaktionsverhalten müssen zusätzlich kenntlich gemacht werden. Post-Ist können kleine Dialoge oder andere dynamische Elemente darstellen (sich verändernde Menü-Einträge,…).

Aufgabe 1 - Vormittag

In dieser Aufgabe soll ein Prototyp des User Interfaces einer kombinierten User-Log- und Alarm Liste für den TCM erstellt werden.

Die Liste soll sowohl Alarm-Events des TCM darstellen als auch Log-Einträge des Benutzers (bezüglich der Benutzeraktivität). Desweiteren muss es möglich sein, neue User-Log Einträge hinzuzufügen. Die Einträge in der Liste sollen standardmäßig nach ihrer Erstellungszeit sortiert werden.

Die Liste soll folgende Funktionalitäten bereitstellen:

Ein auftretendes Alarm-Event muss direkt in der Liste dargestellt werden. Zu jedem Alarm-Event muss folgendes angezeigt werden:

- Zeitpunkt des Auftretens
- Modul, in dem das Alarm-Event aufgetreten ist.
- Beschreibung (Event-Description in TCM)

Dem Benutzer muss die Möglichkeit gegeben werden, ein Alarm-Event zu quittieren – Beim quittieren kann (muss nicht) ein Kommentar des Benutzers angegeben werden – der Kommentar soll ebenfalls in der Liste angezeigt werden. Nicht-quittierte Alarm-Events sollen aus den anderen Listeneinträgen hervorstechen.

Für jeden User-Log Eintrag muss dargestellt werden:

- Zeitpunkt des Eintrags
- Typ (Wartungsarbeit am Motor, Parameteränderung im TCM, Sonstiges)
- Beschreibung

Beim Hinzufügen eines neuen Log-Eintrags soll der Benutzer lediglich die Beschreibung und den Typ des Eintrags angeben – die Uhrzeit des Eintrags wird automatisch vom System ermittelt.

Desweiteren soll die Möglichkeit bestehen, innerhalb der Liste schnell und effektiv nach Einträgen zu suchen. Mögliche Suchszenarien können sein:

- Nur Alarm-Einträge anzeigen
- Nur User-Log Einträge anzeigen
- Einträge zu einer bestimmten Uhrzeit anzeigen (was ist um 11:30 passiert?)
- Nur Alarm-Einträge anzeigen, die von einem bestimmten Modul ausgelöst wurden
- … weitere Suchszenarien sind denkbar – Eigene Vorschläge?

Achtung: Neue, nicht quittierte, Alarm-Events müssen immer angezeigt werden.

Dazu soll die Möglichkeit bestehen, Listeneinträge besonders hervorzuheben – das Hervorheben von Listeneinträgen soll auch rückgängig gemacht werden.

Bei Fragen: meldet euch ☺

Aufgabe 2 - Nachmittag

In dieser Aufgabe soll ein Prototyp für das User Interface eines TCM-„Test-Schedulers“ erstellt werden.

Der Test-Scheduler soll dem Benutzer ermöglichen Testabläufe anzulegen, welche später automatisch abgefahren werden können. Unter einem Testlauf verstehen wir im Folgenden die Veränderung von Sollwerten über die Zeit. Der Test-Scheduler stellt dazu Methoden bereit, die seine elementaren Funktionen repräsentieren. Dem Benutzer soll die Möglichkeit geboten werden, diese Methoden so miteinander zu verknüpfen, dass ein Testlauf entsteht. Es ist sowohl die Hintereinander – sowie auch die Parallelausführung von Methoden möglich.

Ein Testlauf im zu erstellenden Prototyp besteht aus folgenden Methoden:

- Messung Start: Gibt den Zeitpunkt an, an dem die Messung starten soll. Bestimmt die Werte, die gemessen/geloggt werden sollen. Bestimmt den Namen und Ort des LogFiles.
- Messung Stop: gibt den Zeitpunkt an, an dem die Messung stoppen soll.
- Sprung: Der neue Sollwert wird direkt übernommen.
- Rampe: Während einer Zeit t steigt/sinkt der Sollwert linear vom AnfangsSollwert auf den EndSollwert

Jede Methode kann beliebig oft oder gar nicht in einem Testlauf auftauchen. Einzige Restriktion: nach jedem „Messung Start“ muss irgendwo im Programm ein dazugehörendes „Messung Stop“ auftreten.

Das User Interface muss Folgendes bereitstellen:

- Eine Bibliothek die zusätzlich zu den vier beschriebenen Methoden Platz für ca. 100 andere Methoden bietet (diese müssen im Prototyp nicht näher spezifiziert werden). Jede Methode gehört zu einer von 6 Methodenarten.
- Die Möglichkeit aus den Methoden einen Testablauf zusammenzustellen (parallele und sequentielle Ausführung der Methoden).
- Die Möglichkeit die vier vorgestellten Methoden innerhalb eines Testablaufs zu konfigurieren:
    - **Messung Start:** Hier muss der Benutzer angeben können, welche Motor-Messgrößen geloggt werden sollen und wo sie abgespeichert werden sollen. (Unter der Vorraussetzung: Eine Liste aller verfügbaren Motor-Messgrößen ist vorhanden). Desweiteren muss ein Timeout angegeben werden, nach dessen Ablauf die Messung stoppt (falls die Messung nicht vorher schon durch ein **Messung Stop** beendet wurde)
    - **Rampe:** Die Rampe soll die Möglichkeit bieten eine oder mehrere Variablen zu „rampen“. Dazu muss pro Variable ein Start- und Endsollwert angegeben werden. Jede Rampe hat eine Rampendauer t (für alle Variablen innerhalb dieser Rampe gleich) (Liste der Variablen ist vorhanden).
    - **Sprung:** Der Sprung soll die Möglichkeit bieten eine oder mehrere Variablen Sprunghaft zu ändern. Für jede Variable muss der neue Sollwert angegeben werden.
    - **Messung Stop:** ist nicht weiter konfigurierbar.

Bonus: - Einen Test laden/speichern & einen Testablauf starten. Wie wird dargestellt, was das System während eines Tests gerade macht? Wie können Fallunterscheidungen im Methodenablauf behandelt werden?

Bei Fragen: meldet euch ☺

Literaturverzeichnis

Christopher Alexander. *The Timeless Way of Building*. Oxford University Press, New York, 1979. ISBN 0195024028.

Apple. *Macintosh human interface guidelines*. Addison-Wesley Publishing Company, USA, 1992. ISBN 0201622165.

Kent Beck and Ward Cunningham. Using pattern languages for object-oriented programs. tektronix technical report no. cr-87-43. Technical report, Tektronix, 1987.

Hugh Beyer and Karen Holzblatt. Contextual design : Defining customer-centered systems. 1998.

Jan Borchers. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Ltd, 2001.

Jan Borchers. Interdisciplinary design patterns. August 1999. Position Paper, Workshop on Usability Pattern Language.

Dempsey Chang, Laurence Dooley, and Juhani E. Tuovinen. *Gestalt theory in visual screen design: a new look at an old subject*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2002.

Alan Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. *Human-Computer Interaction (3rd Edition)*. Prentice Hall, December 2003. ISBN 0130461091.

Nick Edgar, Kevin Haaland, Jin Li, and Kimberley Peter. Eclipse user interface guidelines, 2001. URL `http://www.eclipse.org/articles/Article-UI-Guidelines/Contents.html`.

International Organization for Standardization. *ISO 11064: Ergonomic design of control centres*. 2000.

International Organization for Standardization. *ISO 13407: Human-centred design processes for interactive systems*. 1999.

International Organization for Standardization. *ISO 14915: Software ergonomics for multimedia user interfaces*. 2002.

International Organization for Standardization. *ISO 9241: Ergonomics of Human System Interaction*. 1998.

International Organization for Standardization. *ISO 9241-110: Ergonomics of Human System Interaction – Dialogue principles*. 2006.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Elements of reusable object-oriented software. 1995.

Wilfred J. Hansen. User engineering principles for interactive systems. pages 523–532, 1971.

Scott Henninger, Charisse Lu, and Candace Faith. Using organizational learning techniques to develop context-specific usability guidelines. pages 129–136, 1997. doi: http://doi.acm.org/10.1145/263552.263594.

Microsoft. Windows user experience interaction guidelines, 2005.

Jakob Nielsen. Iterative user-interface design. *Computer*, 26(11):32–41, 1993a. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.241424.

Jakob Nielsen. *Usability Engineering*. AP Professional, Boston, MA, USA, 1993b. ISBN 0-12-518406-9.

Jakob Nielsen. Enhancing the explanatory power of usability heuristics. pages 152–158, 1994. doi: http://doi.acm.org/10.1145/191666.191729.

Nokia. *Nokia S60 UI Style Guide*. 2005.

Donald A. Norman. *The Psychology of Everyday Things*. Doubleday, 1988.

Donald A. Norman. *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. MIT Press, Cambridge, MA, USA, 1999.

Daphne Ogle. Contextual inquiry, 2009. URL `http://wiki.fluidproject.org/display/fluid/Contextual+Inquiry`.

Michael Plint and Anthony Martyr. *Engine Testing : Theory and Practice*. Butterworth-Heinemann, 1998.

Douglas Schuler and Aki Namioka, editors. *Participatory Design: Principles and Practices*. CRC, 1 edition, March 1993. ISBN 0805809511.

Jorg Schäuffele. *Automotive Software Engineering: Principles, Processes, Methods, and Tools*. SAE International, June 2005. ISBN 0768014905.

Ben Shneiderman. *Designing the User Interface : strategies for effectivehuman-computer interaction*. Addison-Wesley, Reading, Mass. [u.a.], 2. ed. edition, 1992. ISBN 0-201-57286-9.

Ben Shneiderman. Designing for fun: how can we design user interfaces to be more fun? *interactions*, 11(5):48–50, 2004. ISSN 1072-5520. doi: http://doi.acm.org/10.1145/1015530.1015552.

Jenifer Tidwell. *Designing interfaces : Patterns for Effective Interaction Design*. OReilly, 2005. ISBN 0596008031 9780596008031.

Edward R. Tufte. *Beautiful Evidence*. Graphics Pr, Cheshire, Conn., 2006. ISBN 0961392177.

Department of Geography University of Oregon. Design patterns for data graphics, 2007. URL `http://geography.uoregon.edu/datagraphics/patterns/index.htm`.

Index

Gesetzt am 1. Juni 2009