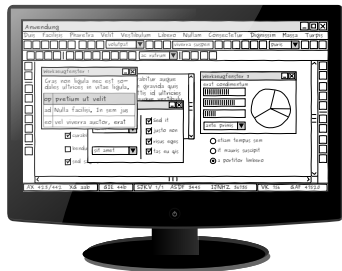


Komplexität und Benutzbarkeit von Anwendungssoftware

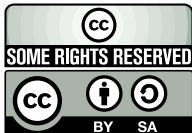


Julian Fietkau

Universität Hamburg

15. Januar 2011

Organisatorisches vorweg



Diese Folien sind unter [CC-BY-SA 3.0](#) freigegeben.

Die GUI-Mock-Ups wurden mit [Pencil](#) gestaltet.

Alle sonstigen Illustrationen, soweit nicht anderweitig gekennzeichnet, stammen aus dem [OpenClipArt-Projekt](#) bzw. basieren auf Inhalten von dort.

Folien-Download und Feedback-Möglichkeit:

http://www.julian-fietkau.de/komplexitaet_und_benutzbarkeit

Übersicht

Einleitung

- Fragestellung

- Begriffe

Komplexität und Kompliziertheit unterscheiden

- Untere Ebene – Einzelne Dialogmasken

- Obere Ebene – Gesamte Anwendung

Ideen und Impulse

- Das Ribbon-UI

- Adaptive User Interfaces

- Feature Layering

- Motivationstechniken aus dem Bereich Game Design

Fazit

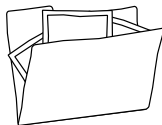
Ein Zitat zum Einstieg

„I have always wished for my computer to be as easy to use as my telephone; my wish has come true because I can no longer figure out how to use my telephone.“

– **Bjarne Stroustrup**

Die „Datenexplosion“

```
C:\> cd STEURRKL  
  
C:\STEURRKL> dir  
<DIR>      .  
<DIR>      ..  
          3,738 ST1985A.TXT  
          3,893 ST1985B.TXT  
          4,102 ST1986.TXT  
3 File(s), 2 Dir(s)  
  
C:\STEURRKL>
```



Mallorca 2010
1079 Objekte
1,58 GB

früher

heute

Blick auf Softwarefunktionalität

Hypothese

Analog zur Datenexplosion existiert eine **Funktionalitäts-Explosion**, in deren Zuge die Anforderungen an vielfältige Funktionalität in Anwendungssoftware stetig steigen.

„I can tell you that nothing we have ever done (...) has increased our revenue more than releasing a new version with more features. Nothing. (...) When we tried Google ads, when we implemented various affiliate schemes, or when an article about FogBugz appears in the press, we could barely see the effect on the bottom line. When a new version comes out with new features, we see a sudden, undeniable, substantial, and permanent increase in revenue.“

– Joel Spolsky

Quantifizierung von Softwarefunktionalität

Beispiel: Microsoft Word

Versionsnr.	Symbolleisten
1.0	2
2.0	2
6.0	8
95	9
97	18
2000	23
2002	30
2003	31
2007ff.	N.A.



(Die Zahlen stammen von [Jensen Harris](#).)

Aber: Die Bedienung soll leichter werden

... oder wenigstens nicht schwieriger.

- Software wird immer schneller konsumiert und soll sofort verwendet werden können. („Don't Make Me Think“ – [Steve Krug](#))
- Das Publikum für Software ist zunehmend heterogen.
- Steigender Konkurrenzdruck macht Benutzbarkeit zum Kaufkriterium.

Was bedeutet Komplexität?

Komplexität nach Don Norman

Komplexität ist ein Maß für die strukturelle Differenziertheit eines Systems. Sie ist objektive Realität und weder gut noch schlecht. Dagegen ist die **Kompliziertheit** ein Maß dafür, wie sehr die Komplexität des Systems einen konkreten Benutzer verwirrt und in seiner Handlungsfähigkeit hemmt, sie ist subjektiv und existiert auf psychologischer Ebene.

Komplexität erscheint dann kompliziert, wenn sie zufällig oder willkürlich ist. Komplexität kann jedoch auch sinnvoll und notwendig sein, um komplexe Aufgaben bewältigen zu können. Das Ziel sollte sein, die komplizierte und verwirrende Komplexität zu erkennen und zu beheben, dabei aber die sinnvolle Komplexität zu erhalten und benutzergerecht zugänglich zu machen.

Ein Beispiel für eine komplexe Schnittstelle



Komplex? Kompliziert?

Foto von Bill Abbott (wbaiv) [via flickr/cc-by-sa](https://www.flickr.com/photos/wbaiv/)

Komplexität ist okay

Komplexität an sich ist nichts schlechtes. Komplexe Aufgaben erfordern ein gewisses Maß an Komplexität in der Anwendungssoftware.

Was die **ISO 9241-110** nicht verlangt: ~~Einfachheit~~

Stattdessen: **Selbstbeschreibungsfähigkeit, Lernförderlichkeit, Erwartungskonformität** (u.a.)

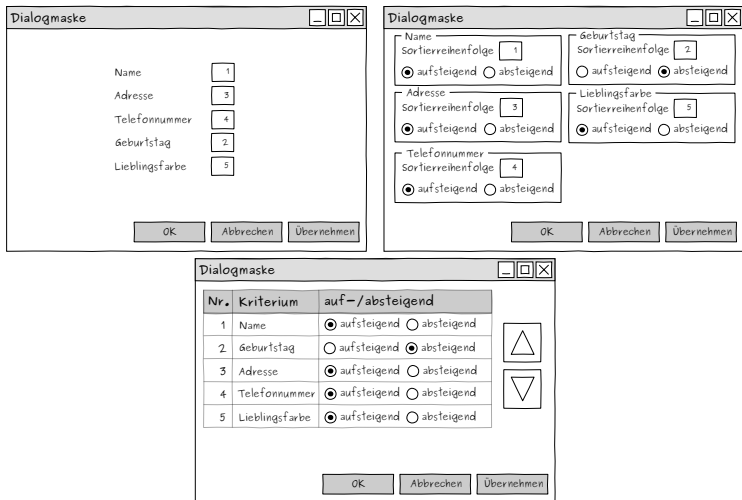
Ein Zitat zwischendurch

„Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves.“

– Alan Kay

Untere Ebene: Einzelne Dialogmasken


- Die benötigte Komplexität ergibt sich aus den Aufgaben.
- Darüber hinausgehende Kompliziertheit kann mit traditionellen Richtlinien für Schnittstellengestaltung eingedämmt werden.
 - Gestaltgesetze, Interaktionsmuster, Nutzen von Vorerfahrung usw.
- In diesem Bereich hat eine beträchtliche Menge Forschung stattgefunden, Hilfen und Literatur existieren zuhauf.




Eine Dialogmaske: (1) zu stark vereinfacht, (2) komplizierte Struktur, (3) klare Struktur mit allen nötigen Features

Obere Ebene: Gesamte Anwendung

- Es gibt erheblich weniger gesicherte Erkenntnisse in diesem Maßstab.
- Grundfrage: **Ist die Struktur der Anwendung sinnvoll hinsichtlich der Aufgaben und Vorkenntnisse der Benutzer?**
 - Sie kann ggf. zu flach und einfach sein (nicht aufgabenangemessen), oder aber zu kompliziert, willkürlich und verwirrend.
- Im Bereich der **Informationsarchitektur** wird an der geeigneten Strukturierung und Präsentation großer Informationsmengen gearbeitet, bisher hauptsächlich für Webseiten.
 - Dort werden Dinge wie Suchstrategien, Aufbereitung großer Inhaltsmengen, Navigationsmöglichkeiten auf Basis semantischer Zusammenhänge der Inhalte etc. untersucht.

 **Fehler**
Sie können auf Ihr Favoriten-Album erst zugreifen, wenn Sie die Bewertung für Einzelbilder aktiviert haben.
Sie finden die Option unter "Extras", "Einstellungen..." unterhalb des Reiters "Bilder-Datenbank".

 **Bewertung für Einzelbilder aktivieren?**

Damit das Favoriten-Album verwendet werden kann, muss erst die Bewertung für Einzelbilder aktiviert werden.

Ein Beispiel für Navigation:

- (1) schlecht, belastet das Kurzzeitgedächtnis, Benutzer soll unnötige Arbeit leisten.
- (2) besser, direkte Hilfestellung, keine weitere Suche nötig.

Ein Zitat zwischendurch

„The old days when we could just go into the back room and develop technology for the Department of Defense are gone. Now we're developing technology for my mother, and that requires a whole new set of skills.“

– Peter J. Denning

Was und warum ist ein „Ribbon“?

Situation: Die Interaktionsmuster „Menüleiste“ und „Symbolleisten“ skalieren nicht mit der wachsenden Funktionalität der Software.

Lösungsvorschlag: Das „Ribbon“-UI als neue Möglichkeit, Funktionen aufgabenorientiert und strukturiert zugänglich zu machen.

Eigenschaften des Ribbon:

- unterschiedlich große Icons je nach Verwendungshäufigkeit
- Tooltips mit vollständigen Beschreibungen statt nur Titeln
- Erweiterte Funktionen in Pop-Ups hinter speziellem Button
- ...

Ein Ribbon-Beispiel

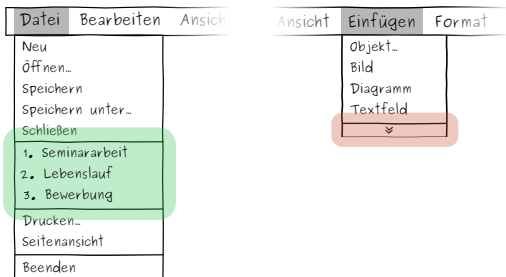


Interfaces, die sich dem Benutzer anpassen

Idee: Unterschiedliche Benutzer verwenden unterschiedliche Funktionen der Software. Kann die Software für den jeweiligen Benutzer „lernen“, welche Funktionen am meisten verwendet werden, und die Struktur der Oberfläche entsprechend anpassen?

Vorteil: Arbeit in Form von Navigation kann eingespart und Wege können verkürzt werden.

Nachteil: Der (Kennen-)Lernprozess kann gestört werden, wenn die Oberfläche sich ohne erkennbaren Grund ändert.



Ein Beispiel für adaptive Aspekte von Menüoberflächen:

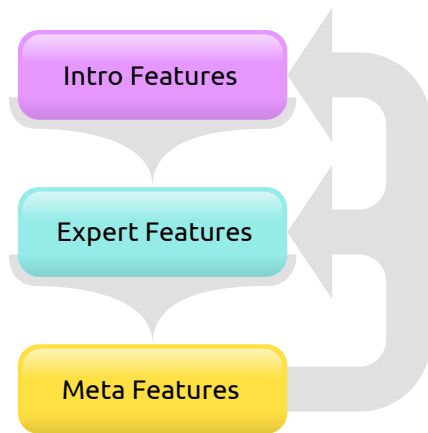
- (1) gut, die Auflistung zuletzt verwendeter Dokumente ist eine etablierte adaptive Funktion.
- (2) schlecht, die adaptiven Menüs hindern die Gewöhnung und das Erlernen der Software.

Benutzer und ihre Lernstadien

Wie kann Software so gestaltet werden, dass Anfänger, Fortgeschrittene und Profis damit arbeiten und beim Lernprozess unterstützt werden können?

- Ermöglichte konsistente mentale Modelle und einfache Metaphern für Anfänger.
- Gestatte es Fortgeschrittenen, in Details einzusteigen und ihre eigene Effizienz zu steigern.
- Gib Profis die Werkzeuge, die Funktionalität der Software zu erweitern und die Erweiterungen den weniger erfahrenen Benutzern zugänglich zu machen.

Benutzer und ihre Lernstadien



Ein Modell für die Strukturierung von Software-Funktionalität.

vgl. [Daniel Cook: One Billion Buttons Please](#)

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								

Ein Beispiel für Feature Layering: Tabellenkalkulationen

Intro: intuitive Papiermetapher, befüllbare Tabelle

Expert: Formeln, Verweise

Meta: Makros

Spiele sind irgendwie anders

Anwendungssoftware. . .

- . . . ist Mittel zum Zweck.
- . . . muss „mühsam“ erlernt werden.
- . . . wird meist als „notwendiges Übel“ empfunden.

Spiele. . .

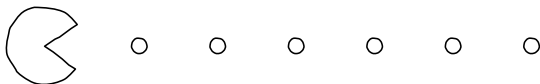
- . . . sind Selbstzweck.
- . . . bewirken wie von selbst (teils beeindruckende) Lernprozesse.
- . . . machen Spaß. → **Motivation!**

Wie schaffen Spiele es, Spaß zu machen?

Game Design und Motivationsforschung

Entwickler von Computer- und Konsolenspielen haben über Jahrzehnte Erfahrung darin gesammelt, Software mit Spaß zu vereinen. Benutzer sind bereit, viel Zeit und Lernaufwand in Software zu investieren, wenn sie dabei Spaß haben.

Erreichbar ist das durch die richtige Balance aus Herausforderung, Feedback zur eigenen Leistung und positiver Bestätigung.



Motivierende Software

- 1 Trenne große Lernprozesse in mehrere kleine auf.
- 2 Baue fortgeschrittene Konzepte auf bereits Erlerntem auf.
- 3 Sorge für eine sanfte Lernkurve.
- 4 Miss den Fortschritt des Benutzers.
- 5 Bewerte und belohne die Leistung.
- 6 Sorge dafür, dass die erlernten Fähigkeiten (mindestens lokal) nützlich sind.

vgl. [Daniel Cook: Building fun into your software designs](#)

Ein Beispiel für zielorientierte Software als Spiel



Wii Fit von Nintendo

Foto von WiredRyo [via flickr/cc-by-sa](#)

Fazit

- Die Komplexität von Software steigt.
 - Das ist kein Grund zur Panik.
- Als Softwareentwickler müssen wir die Komplexität unserer Software kritisch betrachten, bewerten und beherrschen.
 - Einige Ideen zur Bewältigung von Komplexität wurden vorgestellt.
 - Es gibt kein Wundermittel.
- Es gibt noch viel Raum für Forschung und Experimente.

Ein Zitat zum Abschluss

„Complexity is here to stay. The frame of mind is essential: learn to accept complexity, but also learn to conquer it. (. . .) The technologies we use must match the complexity of the world: technological complexity is unavoidable.“

– [Don Norman](#)

Literatur

DONALD A. NORMAN: *Living with Complexity*

The MIT Press (7. Dezember 2010)

ISBN-13: 978-0262014861 ([amazon](#))

Weblinks

Don Norman: Simplicity Is Highly Overrated

http://www.jnd.org/dn.mss/simplicity_is_highly.html

Joel Spolsky: Simplicity

<http://www.joelonsoftware.com/items/2006/12/09.html>

Jensen Harris: The Story of the Ribbon

<http://blogs.msdn.com/b/jensenh/archive/2008/03/12/...-ribbon.aspx>

Daniel Cook: One Billion Buttons Please

<http://www.lostgarden.com/2007/02/one-billion-...-we.html>

Daniel Cook: Building fun into your software designs

<http://www.lostgarden.com/2006/12/building-...-designs.html>