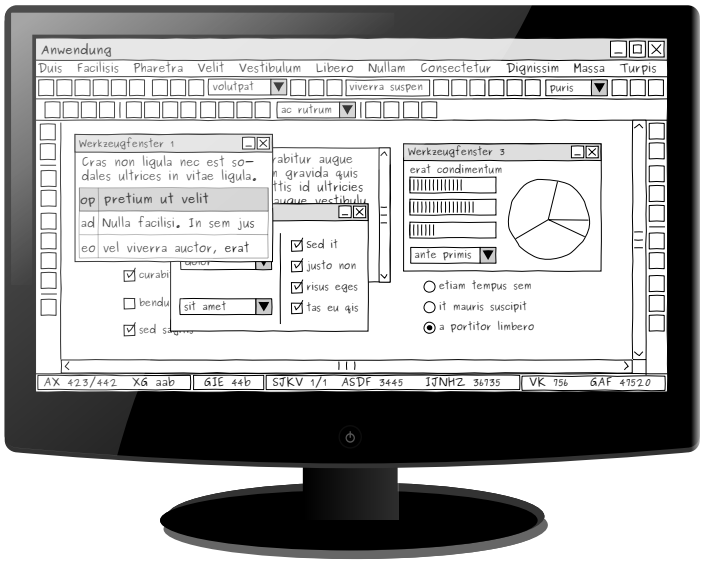
Komplexität und Benutzbarkeit 2012



Julian Fietkau

Universität Hamburg

10. Mai 2012

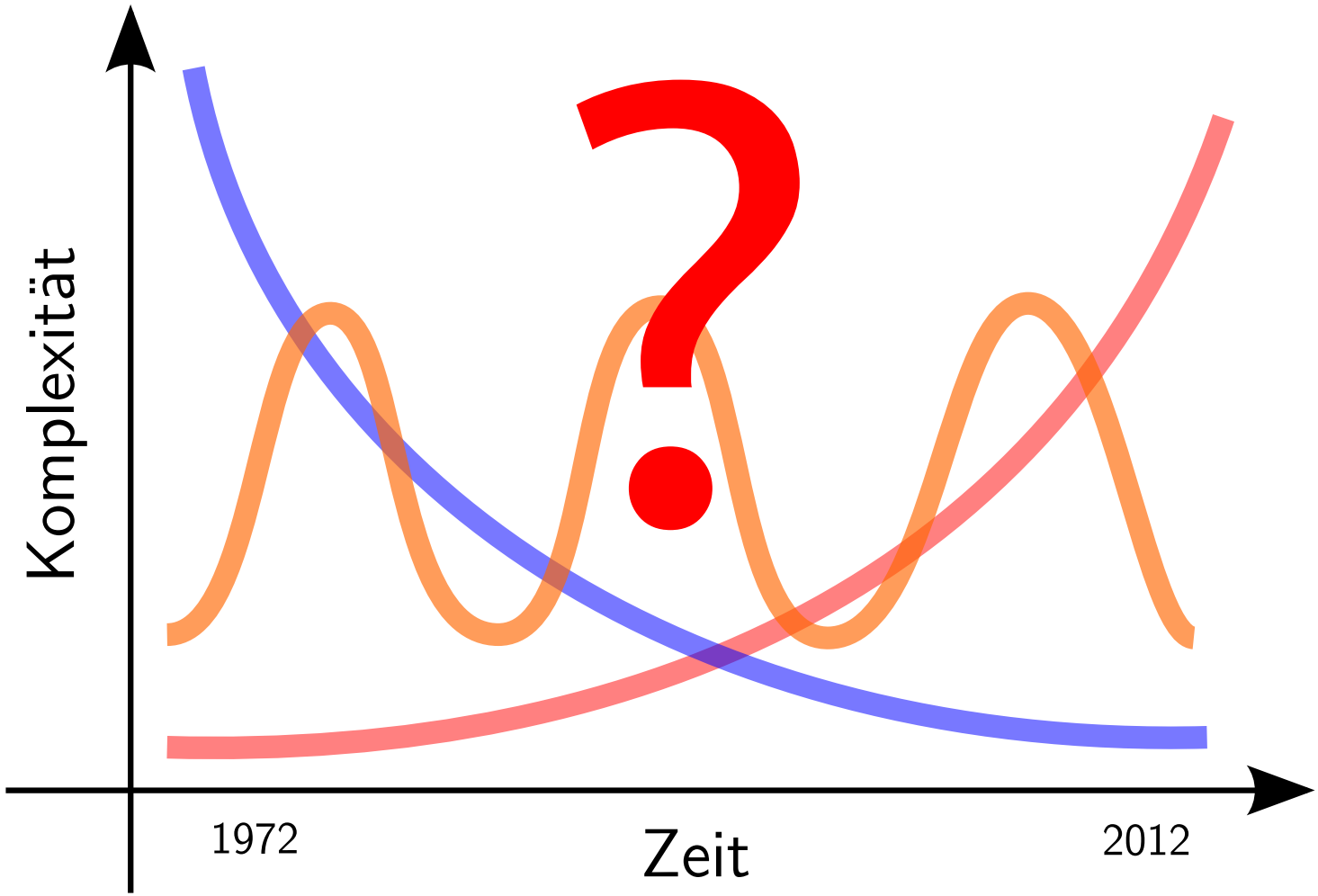
Übersicht

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**

Komplexität über die Zeit



Arten von Software

Systemsoftware: Software, die für Betrieb und Wartung des Systems benutzt wird (Systemkonfiguration, Treiber etc.)

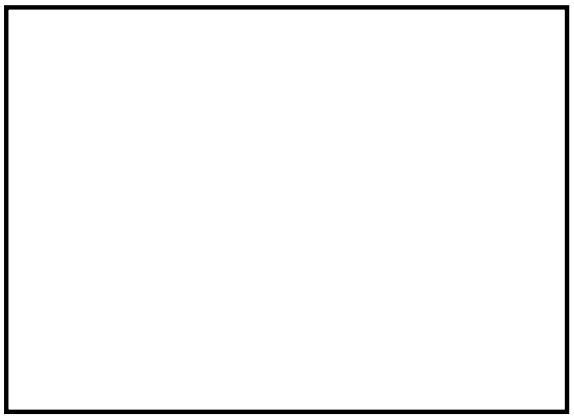
Anwendungssoftware: Software, die mit einem externen Ziel verwendet wird, z.B. für einen Arbeitsprozess.

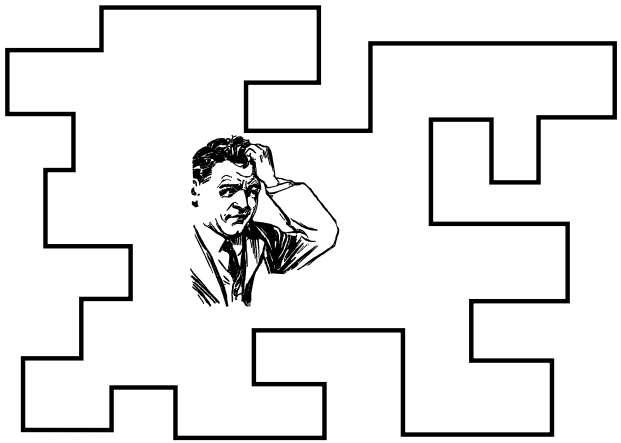
(Computer-)Spiele: Software, die keine produktiven Ergebnisse liefert, aber trotzdem aus intrinsischer Motivation verwendet wird. (Präzisere Betrachtung folgt später!)

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.





Was bedeutet Einfachheit?

Hier verwendet als Gegenteil von Komplexität.

Ergo: Ein „einfaches“ System sagt unmittelbar nichts über intuitive Bedienung o.Ä. aus. Lediglich: Das System hat wenige Ecken und Kanten.

Ein Beispiel für eine meist bekannte Schnittstelle

Komplex? Kompliziert?

Foto von Niels Heidenreich (schoschie) **via flickr/cc-by-sa**

Ein Beispiel für eine meist unbekannte Schnittstelle

Komplex? Kompliziert?

Foto von Bill Abbott (wbaiv) via flickr/cc-by-sa

UNIX: Do One Thing (And Do It Well)

- Jede Software hat eine einzige, klar definierte Aufgabe.
- Die universelle Schnittstelle ist Text.
- Für komplexe Aufgaben werden mehrere Tools verkettet.

`true`: Programm, das sich selbst erfolgreich beendet

- **1970, AT&T UNIX:** leeres Script (0 Bytes)
- **2012, GNU coreutils:** in C geschriebene Software (2,2 KBytes)

UNIX: Do One Thing (And Do It Well)

Problem

Schlechte Discoverability, Umgang mit UNIX-Tools ähnlich komplex wie „richtige“ Programmierung

```
cat liste.txt | grep -i „Müller“ | sort | uniq
```

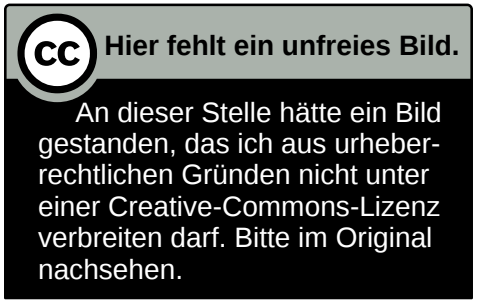
Frühe grafische Schnittstellen

Ära: Erste Rechner der Apple-Macintosh-Reihe, erste Versionen von Microsoft Windows

- Grafische Schnittstellen fördern erkundende Nutzung von Software.
- Funktionalitäten und Konfigurationsmöglichkeiten der Software sind noch nicht sehr umfangreich, deshalb vollständig mental erfassbar.

Archetyp „Enterprise-Anwendung“

Software wird so komplex, dass niemand sie mehr durchschaut.



Das langsame Wachstum von 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**.)

Rettung vor der Komplexität durch Web 2.0?

- Neue, **einfache** Anwendungen im Web 2.0 sollen uns das Leben und die Arbeit erleichtern.
- Es gibt eine (kurze) Welle von „Ein-Button-Anwendungen“.
- Google Docs betritt als schlanke und schnelle web-basierte Textverarbeitung das Feld.
  - Jedoch: Von Funktionsumfang und Komplexität her schließt es schnell zur Desktop-Konkurrenz auf.
- Die „Einfachheit“ erweist sich als Schein-Paradies: Letztlich möchte doch niemand auf liebgewonnene Features verzichten, die Web-Apps expandieren oder gehen unter.

Moderne Touch-Geräte und einfache Apps

- Kleine und simple Apps sind momentan wieder „en vogue“.
  - Evtl. auch bedingt durch technische Grenzen? (Bildschirmgröße etc.)
  - Oder doch ein Effekt der sinkenden Preise für Software?
- Große Frage: Wie lange bleibt das so?

Übergreifende Beobachtung

- Es gibt immer wieder den Wunsch nach einfachen und kleinen Anwendungen.
- Traurige Wahrheit: Bisher haben die sich nie lange gehalten.

Warum der stetige Feature-Zuwachs?

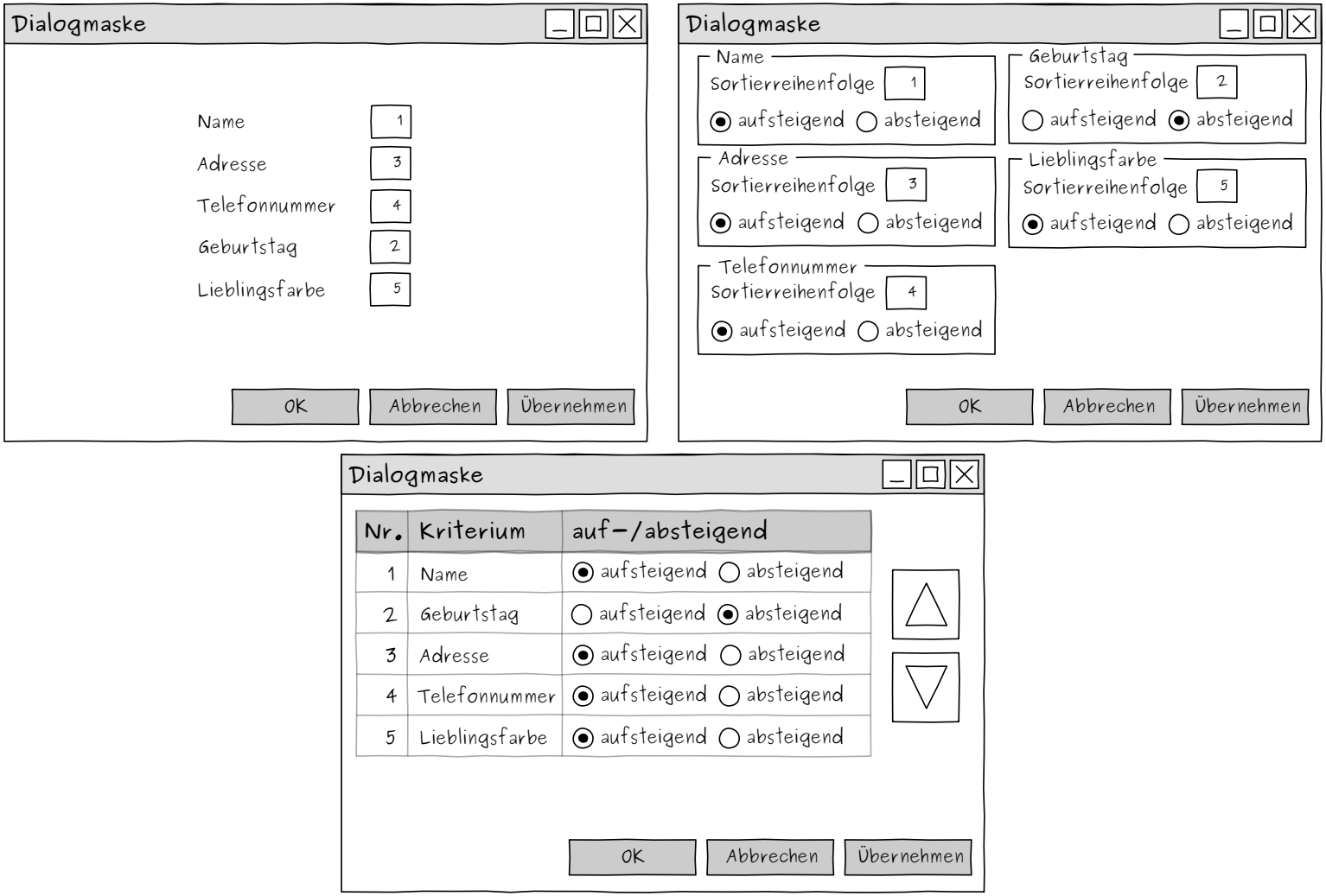
Software-Entwickler wollen einen stets wachsenden Kundenkreis.

„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**

Komplexität ist okay

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



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

Was und warum ist ein „Ribbon“?

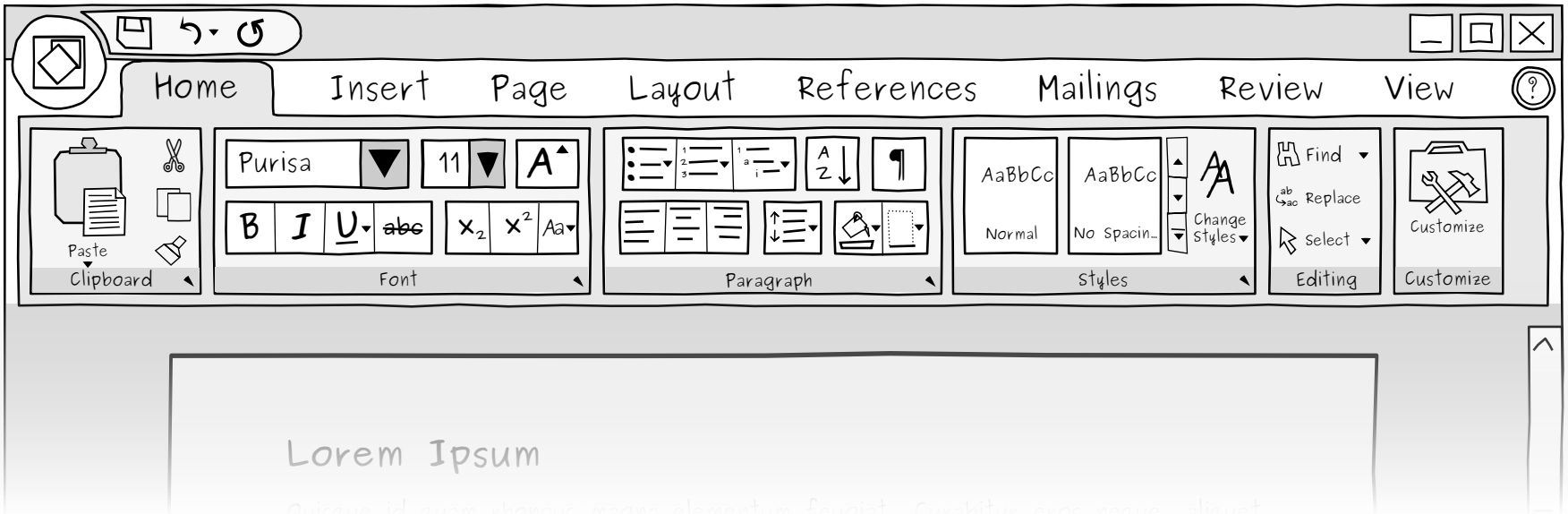
Situation: Die Interaktionsmuster „Menüleiste“ und „Symbolleisten“ skalieren nicht mit der wachsenden Funktionalität der Software.

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



Die Empirie, die das Redesign ermöglichte

Ausschnitt aus

Jensen Harris – The Story of the Ribbon

User testing considered harmful?

Bekannte Situation: Beim Nutzertest, Benutzer kommt nicht mit der Software klar. Was nun?

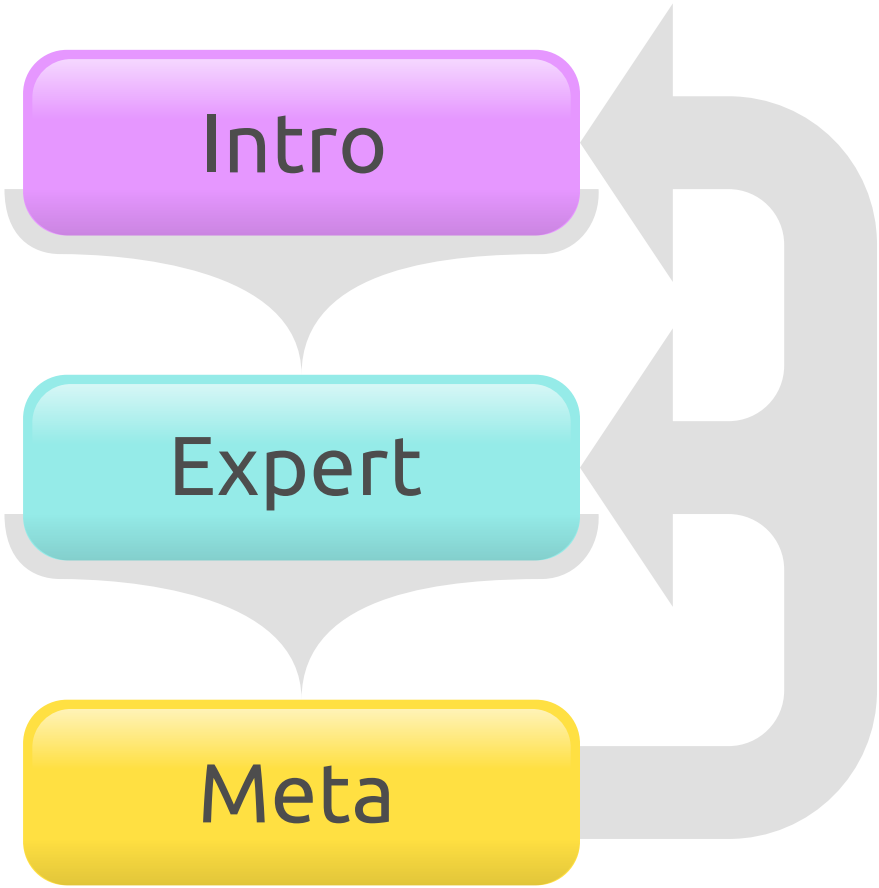
Gefahr von Nutzertests: Ausschließlicher Fokus auf Erstanwender, Vernachlässigung von Lernpotenzial

Benutzer und ihre Lernstadien

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

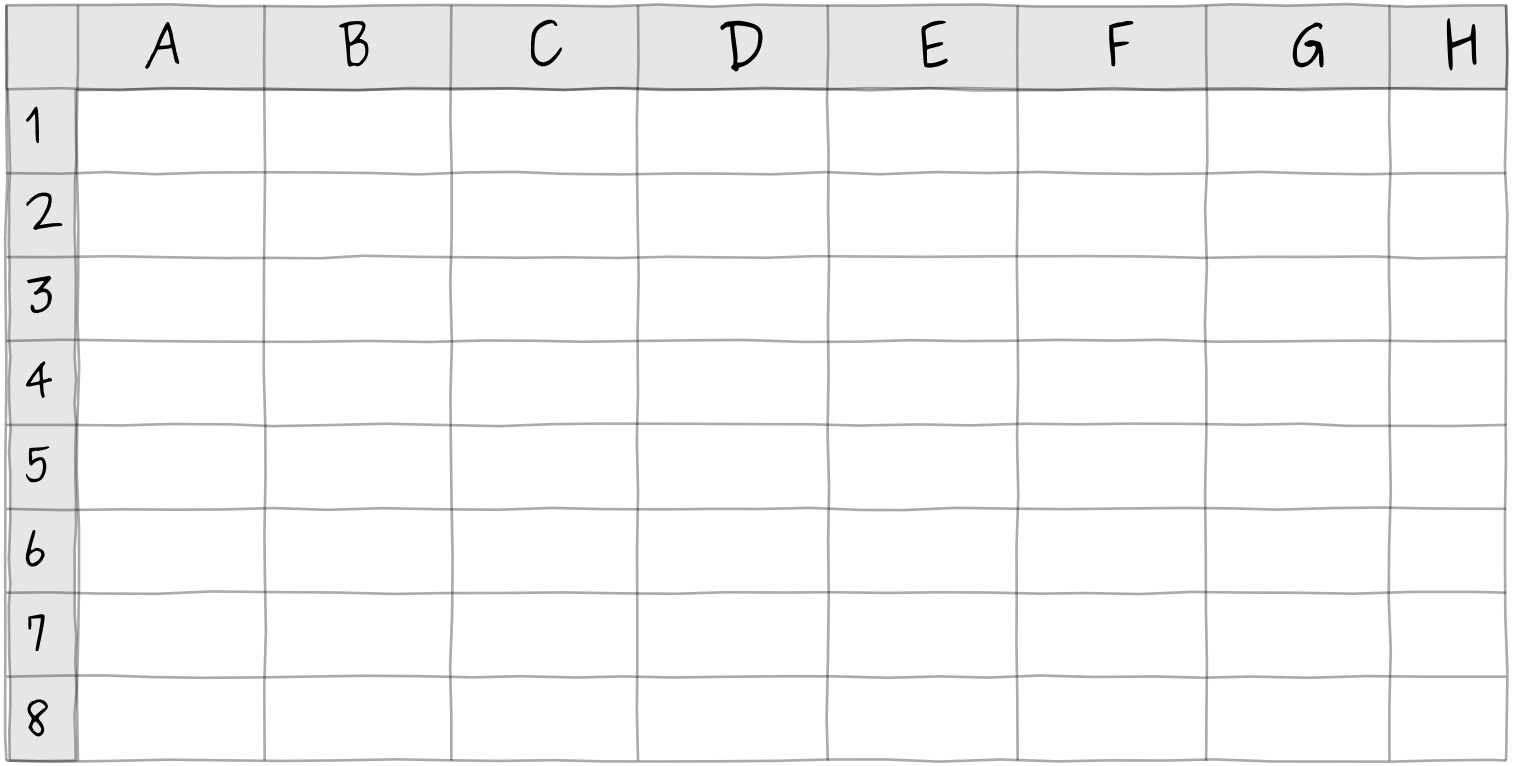
- Ermögliche konsistente mentale Modelle und einfache Metaphern für Anfänger.
- Gestatte es Fortgeschrittenen, in Details einzusteigen und ihre eigene Effizienz zu steigern.
- Gib Profis die Werkzeuge, die Funktionalität der Software zu erweitern und die Erweiterungen den weniger erfahrenen Benutzern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

Layers of Mastery



Ein Modell für die Strukturierung von Software-Funktionalität.

vgl. **Daniel Cook: One Billion Buttons Please**



Ein Beispiel für Feature Layering: Tabellenkalkulationen

Intro: intuitive Papiermetapher, befüllbare Tabelle

Expert: Formeln, Verweise

Meta: Makros

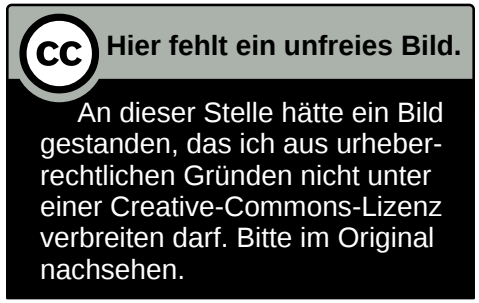


Abbildung: VisiCalc (via **English Wikipedia**)

Lernen... Anstrengung?

Don't make me think. (Steve Krug)

vs.

I want to learn. (Mastery-based Design)

Spiele sind irgendwie anders

Anwendungssoftware...

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

Spiele...

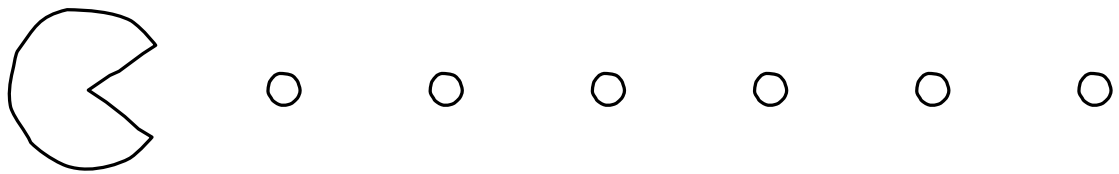
- ...sind Selbstzweck.
- ...bewirken wie von selbst (teils beeindruckende) Lernprozesse.
- ...machen Spaß. $\rightarrow$ **Motivation!**

Wie schaffen Spiele es, Spaß zu machen?

Game Design und Motivatiosforschung

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.



Was zeichnet Spiele aus?

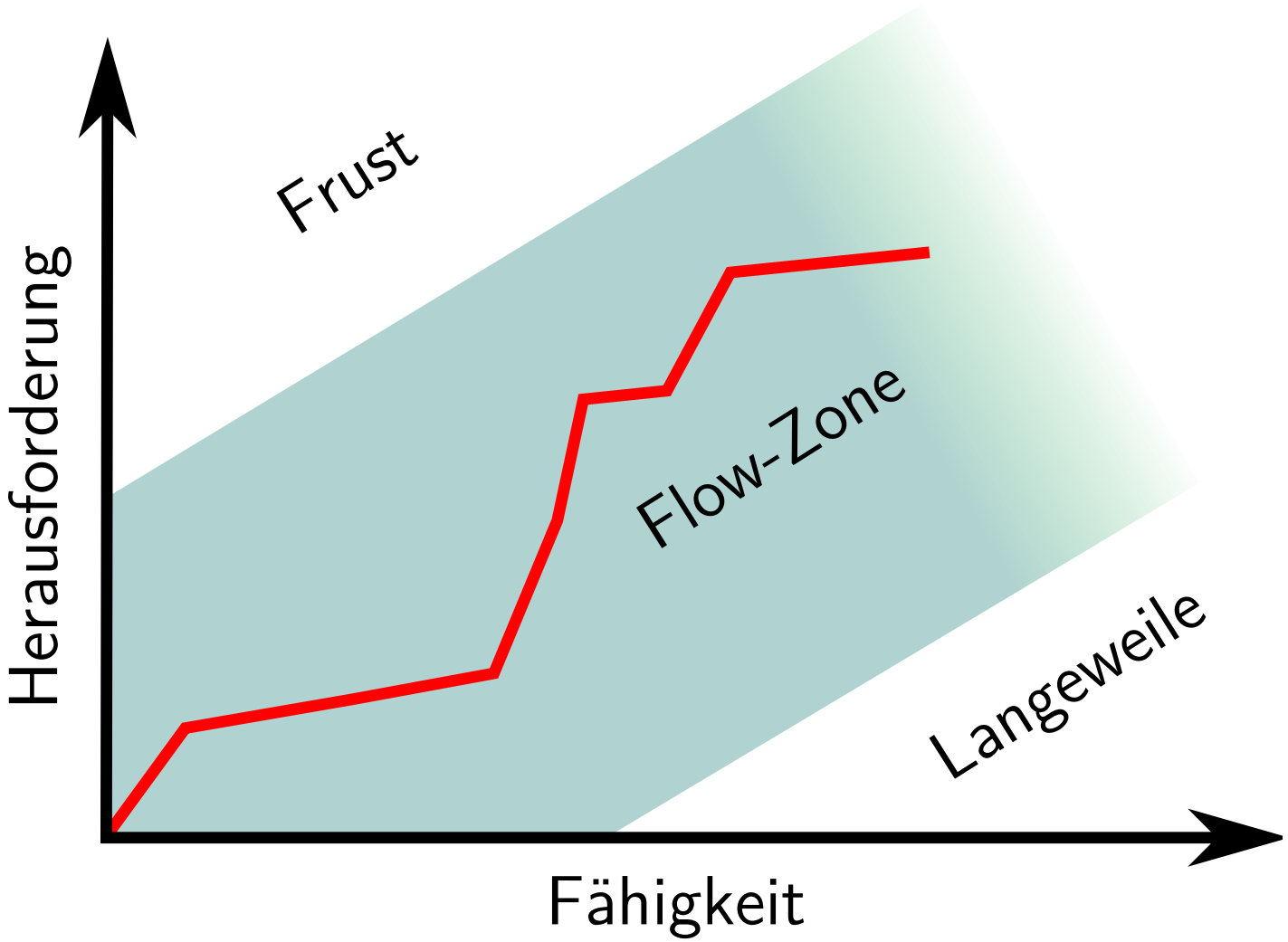
Ausschnitt aus

Jane McGonical – Gaming Can Make a Better World

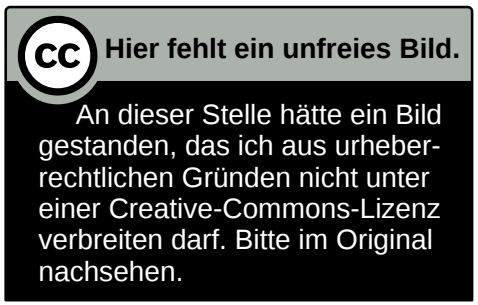
Spiele vs. Spielzeug

„play“ (paidia)	„game“ (ludus)
Spielzeug	Spiel
spontan	Regeln
verspielt	Spielmechanik
Zeitvertreib	Gewinnen
Unterhaltung	Bedeutung

Das „Flow“-Konzept



Herausforderungs-Steigerung in World of Warcraft



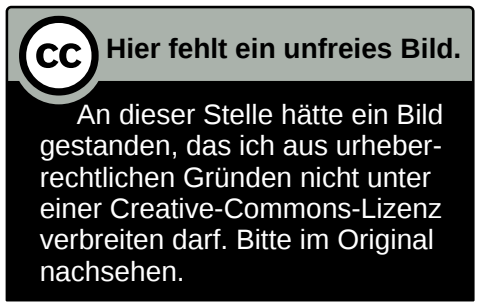


Abbildung: World of Warcraft – Einsteiger und Profi

Design-Prinzipien für Super Mario Bros

"Now, a fun game should always be easy to understand – you should be able to take one look at it and know what you have to do straight away. It should be so well constructed that you can tell at a glance what your goal is and, even if you don't succeed, you'll blame yourself rather than the game."

– **Shigeru Miyamoto**

Regeln Erlernen mit Super Mario Bros

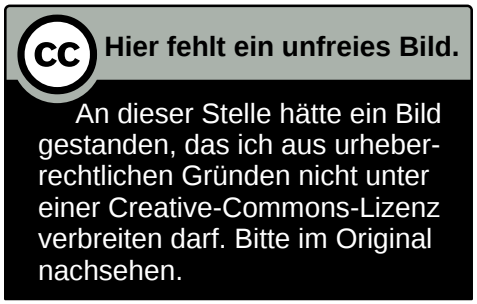


Abbildung: Anfang von Super Mario Bros (NES) (via **Auntie Pixelante**)

Natürliches Lernen durch Spiele

Video

Super Mario Bros – Die ersten zwei Minuten des Spiels

Modernes Beispiel

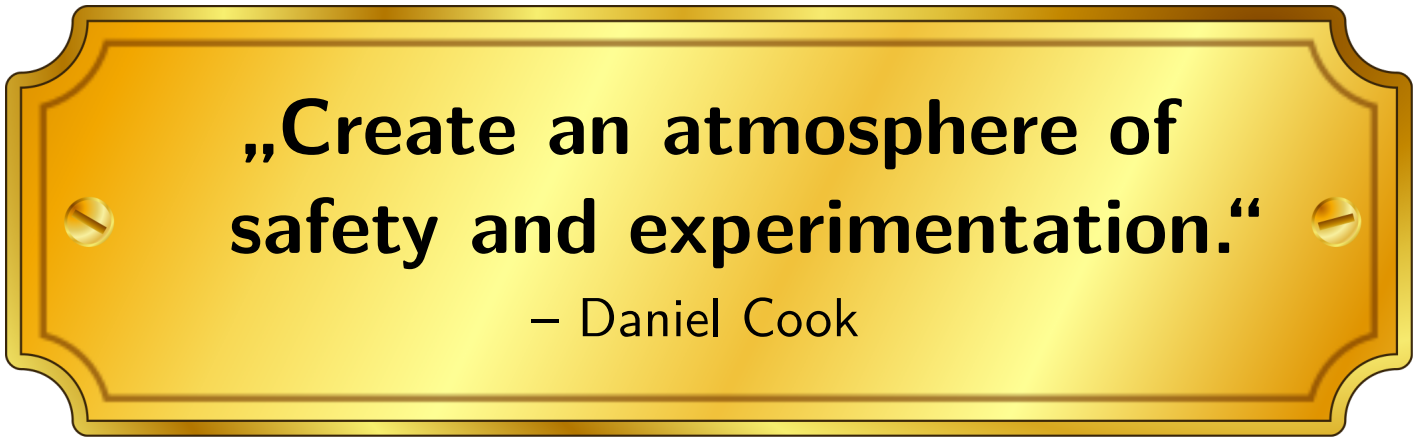
Video

Journey (PS3) – Trailer

Grundprinzip von Gamification

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.

vgl. **Daniel Cook: Building fun into your software designs**



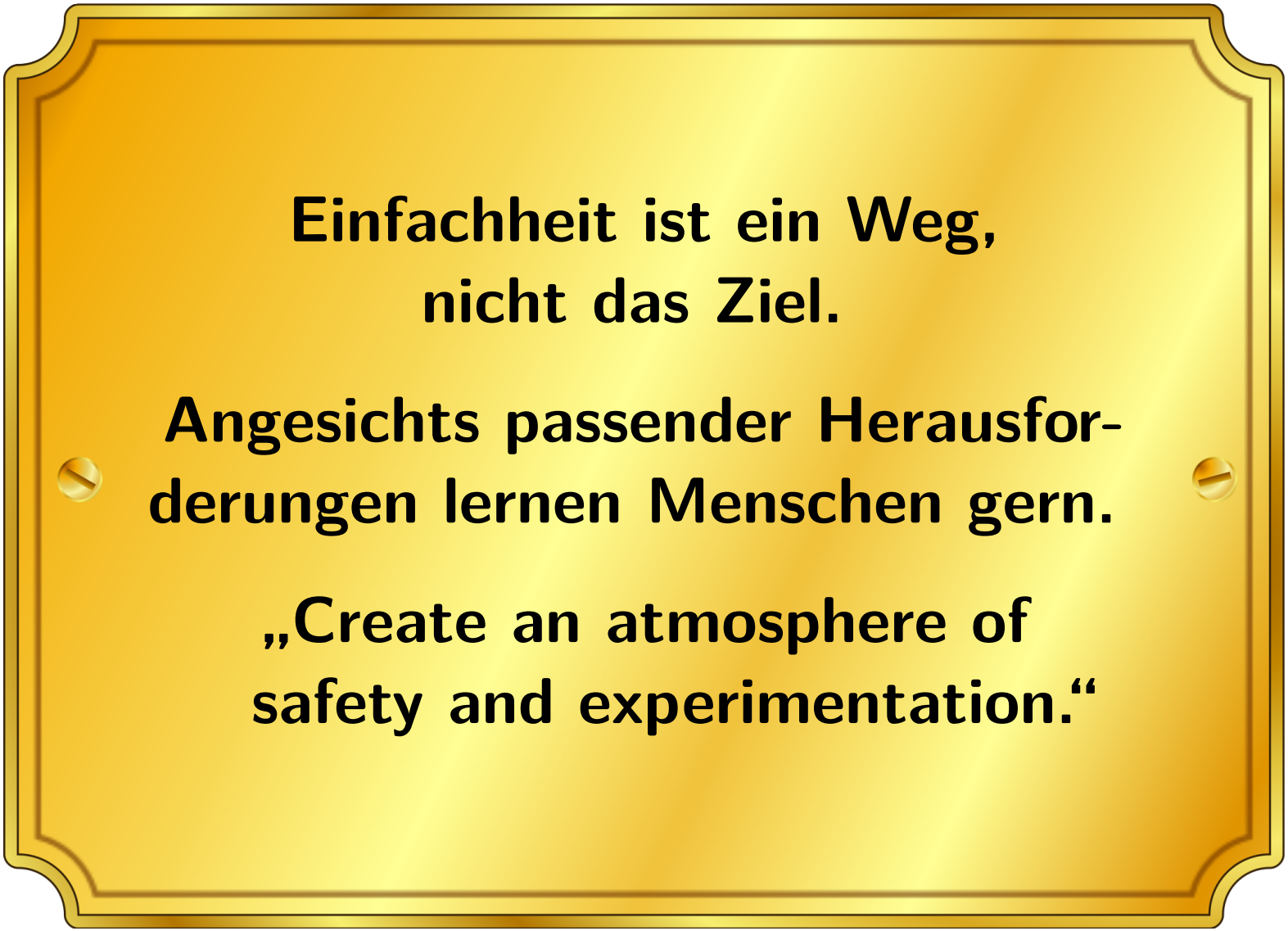
Nichts ist Perfekt

Spiele haben diverse Eigenschaften und Elemente, die sich nicht ohne Weiteres auf jede andere Software übertragen lassen.

- Narration
- Eigendynamik
- Arbeit $\neq$ Spiel
- ...

Fazit

- Die Komplexität von Software wird langfristig weiter steigen.
  - 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.


Einfachheit ist ein Weg,
nicht das Ziel.
Angesichts passender Herausfor-
derungen lernen Menschen gern.
„Create an atmosphere of
safety and experimentation.“

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 (1)

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

Weblinks (2)

Sebastian Deterding: Gamification: A roundtable on game studies and HCI perspective

http://www.slideshare.net/dings/gamification-a-...-perspectives

Sebastian Deterding: Gaming it?

http://www.slideshare.net/dings/gaming-it-was-user-...-2496856

Jenova Chen: Flow in Games (and Everything Else)

http://www.jenovachen.com/flowingames/thesis.html

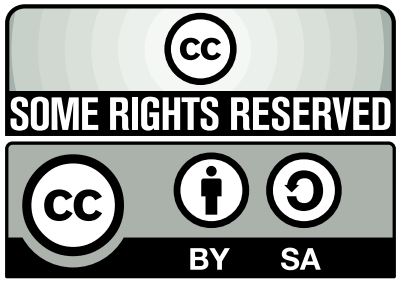
Radek Koncewicz: Super Mario Bros 3 Level Design Lessons

http://www.significant-bits.com/super-mario-bros-3-...-lessons

Egoraptor: Sequelitis – Mega Man Classic vs. Mega Man X

http://www.youtube.com/watch?v=8FpigqfcvlM

Freigabe und Download



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_2012