Vorlesung "Softwaretechnik"

Wiederverwendung, Teil 2

Lutz Prechelt

Freie Universität Berlin, Institut für Informatik

- Beispiele für Muster
  - Benutzbarkeitsmuster
  - Prozessmuster
  - Anti-Muster
- Mustersprachen

Lernziele

- Einen Satz von Benutzbarkeits-Anforderungsmustern grob kennen lernen
  - plus die Idee von zugehörigen Architekturmustern
- Die Idee von Prozessmustern (und einige Beispiele) verstehen
  - auch als Beispiel für eine Mustersprache
- Die Idee von Antimustern (und einige Beispiele) verstehen

Wo sind wir?: Taxonomie "Die Welt der Softwaretechnik"

Freie Universität Berlin

Welt der Problemstellungen:

- Produkt (Komplexitätsprob.)
  - Anforderungen (Problemraum)
  - Entwurf (Lösungsraum)
- Prozess (psycho-soziale P.)
  - Kognitive Beschränkungen
  - Mängel der Urteilskraft
  - Kommunikation, Koordination
  - Gruppendynamik
  - Verborgene Ziele
  - Fehler

Welt der Lösungsansätze:

- Technische Ansätze ("hart")
  - Abstraktion
  - **Wiederverwendung**
  - Automatisierung
- Methodische Ansätze ("weich")
  - Anforderungsermittlung
  - Entwurf
  - Qualitätssicherung
  - Projektmanagement

Wo sind wir?: Wiederverwendung

- Einsicht: Etwas bekanntes wiederzuverwenden kann Qualität und Produktivität stark erhöhen und Risiko senken

- Prinzipien:
  - **Normales Vorgehen**: Vermeide radikales Vorgehen
  - **Universalität**: Fast alles lässt sich im Prinzip wiederverwenden
    - z.B. Anforderungen, Anforderungsmuster, Architekturen, Teilentwürfe, Entwurfsmuster, Testfälle, Dokumentschablonen, Vorgehensbeschreibungen, Checklisten, Prozessmuster
  - **Abwägung:** Wäge sorgfältig den Gewinn an Produktivität und (hoffentlich) Qualität ab gegen den Verlust an Flexibilität und Kontrolle

Teil 2:
Arten von Mustern

Freie Universität Berlin

- Anforderungen
  - Analysemuster ←
  - Akzeptanzkriterien
- Entwurf
  - Referenzarchitekturen
  - Architekturstile/-muster, Entwurfsmuster
  - Produktfamilien
- Benutzungsschnittstellen
  - **Benutzbarkeitsmuster** ←
- Management, Vorgehen
  - Prozessmuster ←
  - Best practices
  - Standards

- Allgemein
  - Prinzipien ←
  - Notationen ←

- Zu den markierten (←) folgen nun Erläuterungen/Beispiele

Benutzbarkeitsmuster I: Anforderungen

- Jedes Benutzbarkeitsmuster ist eine Regel darüber, wie sich interaktive Software verhalten sollte, um gut benutzbar zu sein
  - Sind das also Anforderungs-, Entwurfs- oder Prozessmuster?

- Die allgemeineren dieser Regeln sind inzwischen recht bekannt, deshalb hier nur in Kurzform
  - d.h. Angabe der Lösung ohne
    - Beschreibung von Kontext und Problem
    - Aufgliederung in einzelne Teilanforderungen ("Verantwortlichkeiten")
    - Diskussion von Varianten oder möglichen Nachteilen
- Es folgt ein Katalog solcher Benutzbarkeitsmuster
  - Genannt "usability scenarios" (Bass und John, 2001)
  - Achtung: Diese Muster sind unterschiedlich nützlich, unterschiedlich häufig anwendbar, stellenweise etwas einseitig und die Liste ist ganz sicher nicht vollständig -- aber ein Anfang.

Benutzbarkeitsmuster I: Anforderungen (2)

Freie Universität Berlin

- Aggregating Data
  - Ermögliche, dieselbe Operation auf mehreren Datenelementen zugleich auszuführen
- Aggregating Commands
  - Ermögliche, mehrere verschiedene Operationen zusammen auszuführen (Makros)
- Canceling Commands
  - Ermögliche, längere Operationen unterwegs abzubrechen
- Using Applications Concurrently
  - Ermögliche die gleichzeitige (evtl. verschachtelte) Nutzung mehrerer Anwendungen ohne Konflikte
    - z.B. Spracherkennung in Textverarbeitung ohne Konkurrenz um Cursor

Benutzbarkeitsmuster I: Anforderungen (3)

Freie Universität Berlin

- Checking for Correctness
  - Prüfe Benutzereingaben und zeige Warnungen
  - Biete evtl. vollautomatische Korrektur an (z.B. "hte"→"the")
- Device Independence
  - Arbeite einheitlich und konfliktfrei mit allen in Frage kommenden Eingabe-, Ausgabe- und Speichergeräten zusammen
- Evaluating the System
  - Baue Mechanismen zur Unterstützung von Benutzbarkeitstests gleich mit ins System ein

?

- Failure Recovery
  - Minimiere den möglichen Datenverlust bei HW- oder SW-Ausfällen

Benutzbarkeitsmuster I: Anforderungen (4)

Freie Universität Berlin

- Forgotten-password Reset
  - Sehe einfache, aber hinreichend sichere Mechanismen vor, wie Benutzer nach Vergessen des Passworts wieder Zugang erlangen können
- Help
  - Biete Hilfe kontextabhängig und detailliert genug an, um ein Problem auch wirklich zu lösen
  - Hilfe sollte dafür sorgen, dass die Benutzer die Grundkonzepte des Systems verstehen
- Information Reuse
  - Erlaube, Daten leicht von und zu anderen Anwendungen zu übertragen (import, export, cut/paste, etc.)
- International Use
  - Unterstütze die Umstellung auf andere Sprachen (Zeichensätze, Fonts, Tastaturen etc.)
  - Unterstütze Umstellung auf andere Kulturen (Farben, Symbole)

Benutzbarkeitsmuster I: Anforderungen (5)

Freie Universität Berlin

- Leveraging Human Knowledge
  - Lehne Bedienung an bekannte Software an
  - insbesondere an ggf. frühere Versionen desselben Produkts
- Navigating Within a Single View
  - Erlaube schnelle und einfache Navigation auch in großen Datenmengen und ohne die Ansicht verlassen zu müssen
- Observing System State
  - Zeige alle benutzerrelevante Information über den Systemzustand in verständlicher und übersichtlicher Form an
- Working at the User's Pace
  - Passe die Geschwindigkeit automatischer Aktionen (z.B. Rollen, Timeouts) der Arbeitsweise der Benutzer an (+Einstellbarkeit)

Benutzbarkeitsmuster I: Anforderungen (6)

Freie Universität Berlin

- Predicting Task Duration
  - Zeige bei längeren Arbeitsphasen des Rechners deren voraussichtliche Dauer an
- Supporting Comprehensive Searching
  - Erlaube Suche flexibel über alle Daten mittels aller benutzerrelevanter Kriterien
- Supporting Undo
  - Erlaube, Aktionen rückgängig zu machen; auch mehrere hintereinander
- Working in an Unfamiliar Context
  - Biete einen Anfängermodus an, in dem viele Erklärungen angezeigt werden

Benutzbarkeitsmuster I: Anforderungen (7)

- Verifying Resources
  - Prüfe vor einer Operation, ob diese vermutlich erfolgreich ausgeführt werden kann, insbesondere alle Ressourcen verfügbar sind (typischer Fall: Plattenplatz)
- Visualizations
  - Biete mehrere strukturell verschiedene Ansichten der Daten an
- Operating Consistently Across Views
  - Stelle in verschiedenen Ansichten der gleichen Daten möglichst immer die selben Operationen zur Verfügung
- Making Views Accessible
  - Umgekehrt sollten während aller Operationen und Modi möglichst alle Ansichten leicht zugänglich bleiben

Puh. Viel Arbeit!

Taxonomie von Benutzbarkeits-Maßnahmenarten

Eine Gruppierung der Anforderungen (als Auswahlhilfe):

- Increases individual user effectiveness
  - *Expedites routine performance*
    - Accelerates error-free portion of routine performance
    - Reduces the impact of routine user errors (slips)
  - *Improves non-routine performance*
    - Facilitates learning
    - Supports problem-solving
  - *Reduces the impact of user errors caused by lack of knowledge*
    - Prevents mistakes
    - Accommodates mistakes
- Reduces the impact of system errors
  - *Prevents system errors*
  - *Tolerates system errors*
- Increases user confidence and comfort

Wozu gehört
Supporting Undo?
Predicting Task Duration?

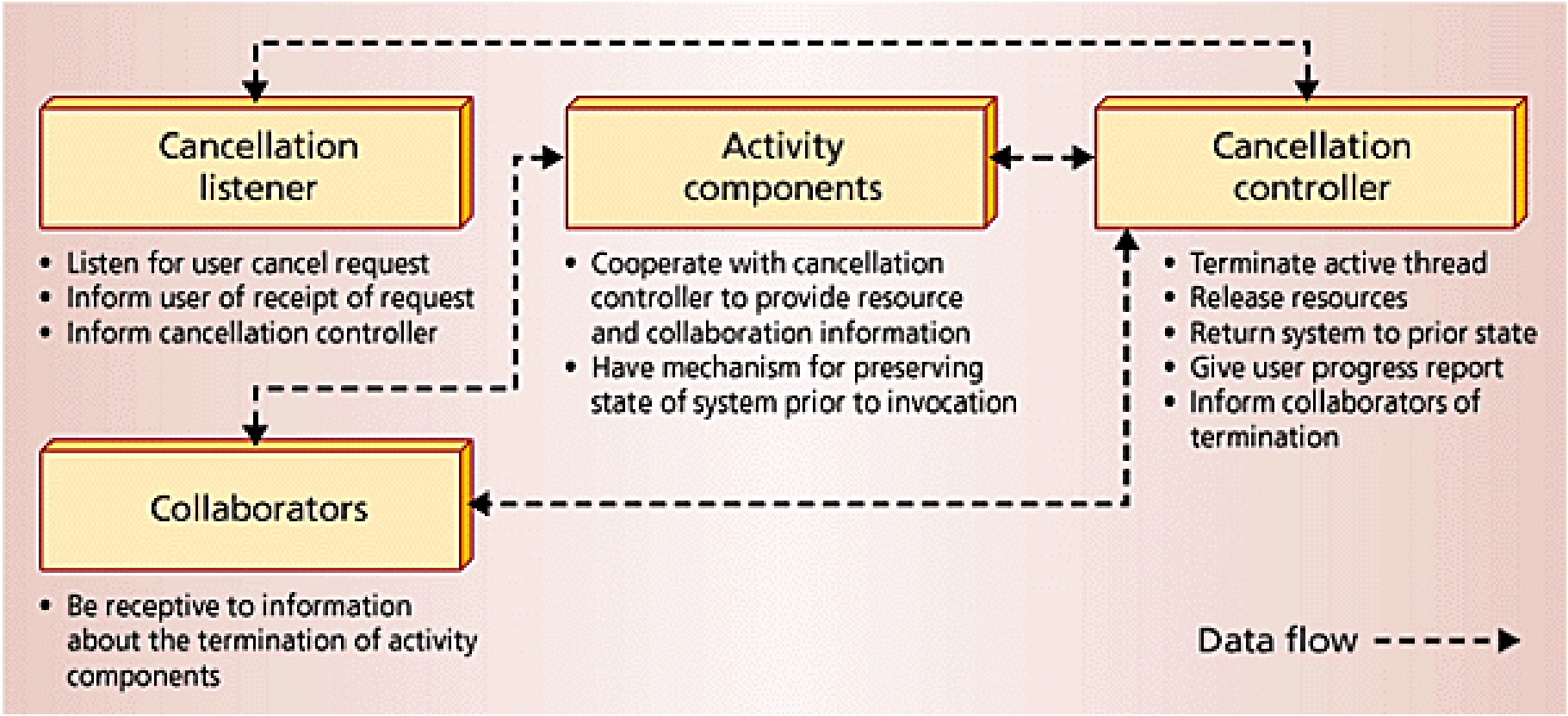
Benutzbarkeitsmuster II: Entwürfe

- Usability-supporting architectural patterns (USAP)
  - beschreiben den Zusammenhang zwischen den obigen Benutzbarkeitsanforderungen und den Mechanismen zu Ihrer Implementierung

Zunächst werden Basismechanismen identifiziert und beschrieben:

- Separation
  - Encapsulating function
  - Sep. data from commands
  - Sep. data from the view
  - Sep. authoring from execution
- Replication
  - Data
  - Commands
- Indirection
  - Data
  - Function
- Recording
- Preemptive scheduling
- Models
  - Task
  - User
  - System
- Jedes USAP erläutert für eine Anforderung, wie die Mechanismen einzusetzen sind.
  - Beispiel:

USAP: Abbruch von Operationen



- Muster für Einzelplatz-Desktop-Anwendungen
  - Im Original natürlich mit genauerer verbaler Beschreibung

Arten von Mustern

- Anforderungen
  - Analysemuster ←
  - Akzeptanzkriterien
- Entwurf
  - Referenzarchitekturen
  - Architekturstile/-muster, Entwurfsmuster
  - Produktfamilien
- Benutzungsschnittstellen
  - Benutzbarkeitsmuster ←
- Management, Vorgehen
  - **Prozessmuster** ←
  - Best practices
  - Standards

- Allgemein
  - Prinzipien ←
  - Notationen ←

- Zu den markierten (←) folgen nun Erläuterungen/Beispiele

Prozessmuster

- Prozessmuster lassen sich auf unterschiedlichen Ebenen formulieren:
  - Projektmanagement
    - Wer macht was wann und wie wird geplant?
  - "Technische" Aufgaben
    - Vorgehensweisen für z.B. Durchsichten, Entwurfsdiskussionen, Systemfreigaben
  - Rahmenbedingungen
    - Aufbau der Organisation, Verantwortlichkeiten, Form der Zusammenarbeit

- Wir besprechen im Folgenden einige Projektmanagement-Muster
  1. Muster über Planung und Termineinhaltung
  2. Muster über Umgang mit Störungen

Eine Projektgeschichte

- Jemand erzählt:
  - "Wir übertrugen ein erfolgreiches Produkt auf eine völlig neue Technologie. Der Erfolg unserer Prototypen gab uns das Vertrauen, ein Lieferdatum festzulegen.
  - Leider ging unser Lieferzeitpuffer verloren. Daraufhin rief der Projektleiter alle zusammen, gab eine lange Terminverzögerung bekannt und erhielt von allen wichtigen Projektmitgliedern gemeinsame Zustimmung zum neuen Plan.
  - Wir hielten diesen neuen Plan ein und lieferten sehr gute Qualität: Teile des Produktes wurden noch viele Jahre später in anderen Projekten wiederverwendet."

- Die unterstrichenen Phrasen deuten auf die Anwendung von Prozessmustern
  - Wir besprechen diese (und einige mehr) auf den nächsten Folien
  - Beachte: Muster beschreiben stets etwas ***Bewährtes***, nicht etwas *Neues*

Muster: Baue Prototypen (*build prototypes*)

- Wenn
  - Anforderungen genauer verstanden oder mit Kunden validiert werden müssen
  - Bedienbarkeit überprüft werden soll
  - oder die Vor- und Nachteile wichtiger Entwurfsentscheidungen unklar sind
- dann
  - baue einen Prototyp: Ein möglichst kleines System, das nur den Zweck hat, diese Fragen zu klären und daraus zu lernen
  - Wirf den Prototyp anschließend weg (den Code, nicht unbedingt den Entwurf)

- Vorteile:
  - Senkt das Risiko der anschließenden Entwicklung

Muster: Liefertermin festlegen (*size the schedule*)

Freie Universität Berlin

- Wenn
  - die Anforderungen stabil und verstanden sind und
  - der Projektumfang geschätzt werden kann
- dann
  - lege einen verbindlichen Liefertermin gegenüber dem Kunden fest
  - lege einen etwas früheren verbindlichen Abschlusstermin in Abstimmung mit dem Projektteam fest
    - (die Differenz erzeugt einen Liefertermin-Puffer)
  - belohne das Team, wenn es diesen Termin einhält

- Vorteile:
  - Erzeugt klare Orientierung und gute Motivation
  - Es ist meistens sinnvoller, nötigenfalls die Funktionalität zu reduzieren als den Liefertermin hinauszuschieben

Muster: Beobachte den Liefertermin-Puffer (*completion headroom*)

- Wenn
  - ein fester Liefertermin eingehalten werden muss
  - und ein realistischer Zeitplan dafür vorliegt
- dann
  - aktualisiere ständig (z.B. wöchentlich) die Restzeit-Schätzungen
  - beobachte die Größe des Zeitpuffers zwischen Schätzung und verbindlichem Termin
  - und ergreife Gegenmaßnahmen, wenn der Puffer verschwindet und dauerhaft verschwunden bleibt

- Vorteile:
  - Kontinuierlich hohe Glaubwürdigkeit der Terminplanung
  - Rechtzeitiges Warnsignal, sobald ernste Terminschwierigkeiten auftauchen

Muster: Keine kleinen Terminverzögerungen (*take no small slips*)

Freie Universität Berlin

- Wenn
  - der Liefertermin-Puffer negativ wird
  - und sich ein Trend von weiteren Verzögerungen einstellt
- dann
  - mache eine neue Liefertermin-Schätzung
  - und sehe genug zusätzliche Zeit vor, dass später keine weiteren Verzögerungen des Termins nötig werden

- Vorteile:
  - Vermeidet Verluste, die durch zu großzügige anfängliche Terminsetzungen entstehen können
  - Vermeidet Ausbrennen von Entwicklern durch dauerhaft zu hohen Termindruck; sorgt für Glaubwürdigkeit der Terminplanung
  - Vermeidet Verärgerung von Kunden durch mehrfache Verzögerungen

Muster: Gemeinsame Zustimmung zum neuen Plan (*recommittment meeting*)

Freie Universität Berlin

- Wenn
  - der bisherige Zeitplan eindeutig nicht eingehalten werden kann
  - und Hilfsmaßnahmen wie Wochenendarbeit oder Zusatzpersonal nicht als Lösung ausreichen
- dann
  - führe ein Treffen mit Management und den Projekt-Schlüsselpersonen durch
  - in dem verschiedene Lösungsmöglichkeiten diskutiert werden (Funktionalität verringern, Zeit verlängern),
  - ein neuer Inhalts- und Zeitplan erarbeitet wird
  - und sich abschließend alle auf diesen neuen Plan verpflichten

- Vorteile:
  - Erzeugt angemessene und realistische neue Pläne
  - Erzeugt die Bereitschaft aller, einen neuen Plan wirklich ernst zu nehmen

Muster für Ablenkungen, Störungen, Unterbrechungen

Freie Universität Berlin

Ablenkungen sind Wünsche, die von außen neu an das Projekt herantreten und den Arbeitsfluss stören/unterbrechen

- **Opfere eine Person** (*sacrifice one person*):
  - Für eine kleinere Ablenkung, stelle eine Person ab, die sich allein um dieses Problem (und evtl. weitere) kümmert.
    Alle anderen arbeiten ungestört weiter
- **Ein Team pro Aufgabe** (*team per task*):
  - Für jede große Ablenkung ("Krise"), stelle mehrere Personen ab, die sich gemeinsam um das Problem kümmern.
    Alle übrigen arbeiten ungestört weiter

Vorteil?

- **Irgendjemand macht Fortschritt** (*someone always makes progress*):
  - Wenn ständige Ablenkungen den Fortschritt bremsen, sorge unter allen Umständen dafür, dass zumindest irgendjemand noch auf das ursprüngliche Ziel zuarbeitet

Muster für Unterbrechungen (2)

- **Kindertagesstätte** (*day care*):
  - Wenn die erfahrenen Entwickler zu stark durch die Einweisung von Neulingen aufgehalten werden, stelle einen davon ab, sich um alle Neulinge zu kümmern und lasse die übrigen in Ruhe
  - Die Neulinge müssen nur wenig produktive Arbeit abliefern, die aber in hoher Qualität

- **Söldner-Autor** (*mercenary analyst*)
  - Wenn die erfahrenen Entwickler zu stark durch das Schreiben von (Entwurfs-)Dokumentation aufgehalten werden, heuere einen erfahrenen Technischen Autor mit Domänenwissen an, um alle Dokumentation zu verfassen
- **Löse Blockaden** (*interrupts unjam blocking*)
  - Jemand, der zwingend für eine Aufgabe benötigt wird, deren Verzögerung das ganze Projekt zum Stillstand brächte, wird sofort unterbrochen und dort hin gerufen
  - und kann dann keinesfalls unterbrochen werden

Muster: Programmieren in Episoden (*programming episodes*)

- Wenn
  - Anforderungen und Grobentwurf verstanden sind,
  - eine große Menge Code zu schreiben ist und
  - die eigene Konzentrationsfähigkeit begrenzt ist
- dann
  - finde und verwende einen Rhythmus des Entwickelns in sinnvoll abgeschlossenen Episoden:
  - Wähle für die Episode aus, was darin fertiggestellt werden wird,
  - stelle ausreichend Zeit und Konzentration dafür bereit,
  - treffe alle nötigen Entwurfsentscheidungen und setze sie um.
  - Breche eine Episode nicht vor Fertigstellung ab.

- Vorteile:
  - Erzeugt hohe Qualität
  - Erlaubt effizient einen verlässlichen, planbaren Fortschritt

Anmerkung zu den Prozessmustern

Freie Universität Berlin

- Diese Beschreibungen sind nur Skizzen
  - Im Original sind viel mehr Details, Überlegungen, Auswirkungen, Verbindungen zu anderen Mustern etc. beschrieben
    - Die Muster sind nicht so naiv, wie sie hier zum Teil vielleicht geklungen haben
- Wie alle Muster, so stammen auch diese direkt aus der Praxis
  - Sie sind echte, erfolgreiche Lösungen aus realen Software-Organisationen
- Sie wirken vielleicht banal, sind es aber nicht
  - Sie sind relevante und erprobte Antworten auf häufige und schwierige Entscheidungsprobleme

Arten von Mustern

- Anforderungen
  - Analysemuster
  - Akzeptanzkriterien
- Entwurf
  - Referenzarchitekturen
  - Architekturstile/-muster, Entwurfsmuster
  - Produktfamilien
- Benutzungsschnittstellen
  - Benutzbarkeitsmuster
- Management, Vorgehen
  - Prozessmuster
  - Best practices
  - Standards

- Allgemein
  - Prinzipien
  - Notationen

Jetzt folgen noch:

- Mustersprachen (pattern languages)
- Anti-Muster

Mustersprachen (Pattern Languages)

Freie Universität Berlin

Anmerkung zu Mustern (fast aller Art):

- Wo Muster anwendbar sind, treten oft immer wieder mehrere ähnliche Probleme zusammen auf
  - aber nicht unbedingt immer alle zusammen
- Dementsprechend kann man auch häufig mehrere Muster gemeinsam einsetzen
- Um das zu erleichtern, werden Muster oft nicht einzeln präsentiert, sondern zu einer Gruppe zusammengeschlossen
  - genannt **Mustersprache**
  - Verwandtschaften, Querverweise oder gruppenweise Beschreibungen der Muster beschreiben, wie die Muster zusammenwirken (können)
    - siehe die Prozessmuster eben
- Versuchen Sie ggf. stets die Mustersprache als ein Ganzes zu verstehen
  - nicht nur die einzelnen Muster

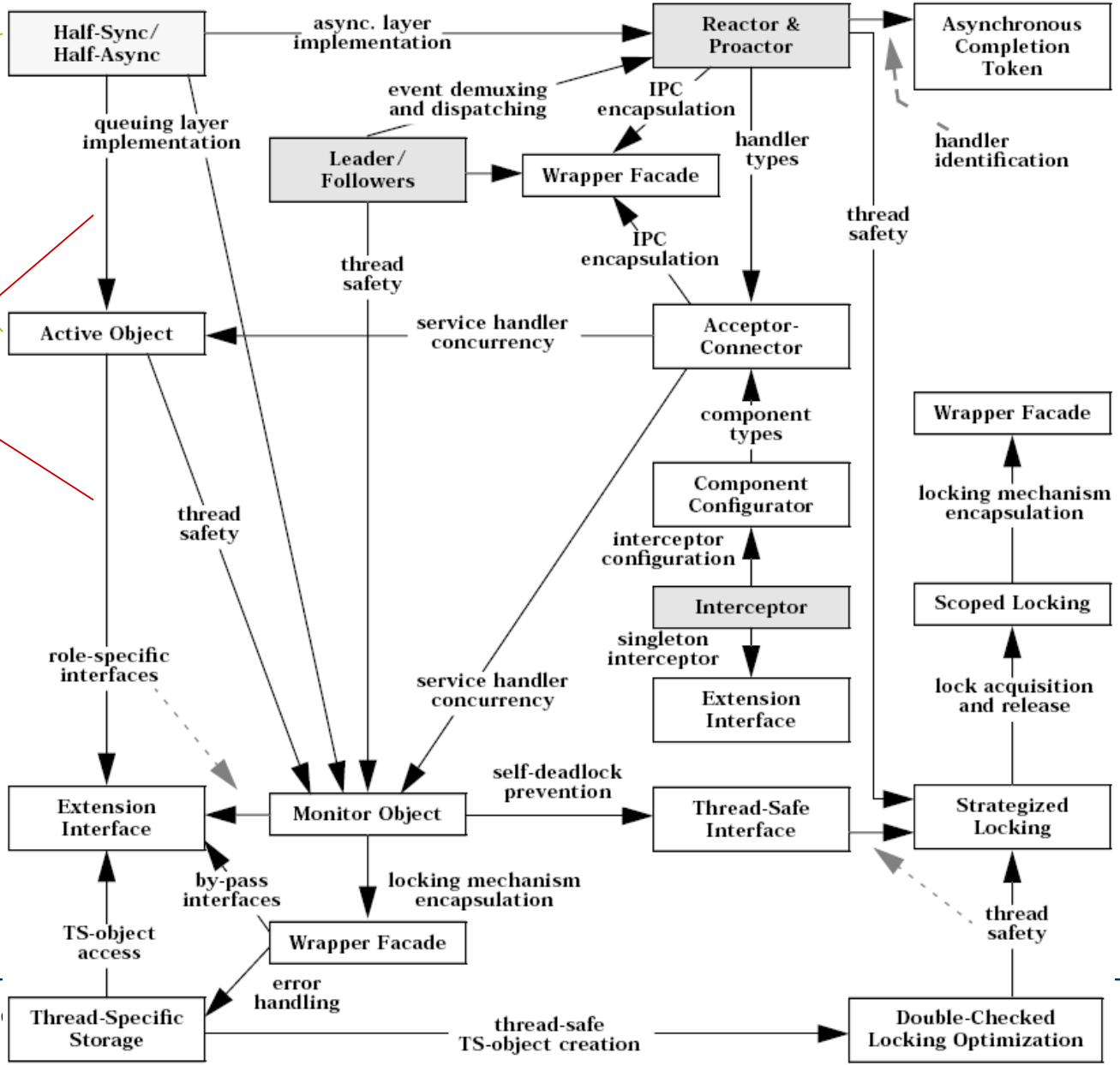
Beispiel: Eine Mustersprache für verteilte Anwendungen

Freie Universität Berlin

Muster

unterstützt-von

aus:
D. Schmidt et al.:
"Pattern-orientierte Softwarearchitektur: Muster für nebenläufige und vernetzte Objekte"



Nachbemerkung: Anti-Muster

- Antimuster sind Lösungen die besonders ungünstige Eigenschaften haben,
  - aber ebenso wie Muster in der Praxis wiederholt auftreten

- Sie zu studieren, kann ebenfalls lohnend sein
  - vor allem als Argumentationshilfe gegen drohende inkompetente Lösungen oder für deren Reparatur

Es folgen einige Beispiele (in zufälliger Reihenfolge):

Beispiele für Anti-Muster

- Prozess: Analyse-Paralyse (*analysis paralysis*)
  Prozess: Kernkompetenz Bildentwurf (*viewgraph engineering*)
  Prozess: Tod durch Planen (*death by planning*)
  - Perfektion anstreben in der Anforderungsermittlung, in einer reinen Entwurfsphase oder in der Projektplanung und dadurch enorm viel Zeit verschenken

- Überall: Das Rad wiedererfinden (*reinvent the wheel*)
  - Eine ausgereifte, bekannte Lösung nicht benutzen, sondern etwas Schlechteres selbst entwickeln
    - Vor allem bei Architektur, Entwurf, Implementierung; aber auch Anforderungen und Prozess
    - → unnötiges radikales Vorgehen

- Überall: Goldener Hammer (*golden hammer*)
  - Ein bekanntes und bewährtes Konzept, das übermäßig intensiv eingesetzt wird

Beispiele für Anti-Muster (2)

- Wiederverwendung: Sackgasse (*dead end*)
  - Man hat eine erhebliche Veränderung an einer Komponente eines Drittherstellers vorgenommen und ist nun von deren Fortentwicklung abgekoppelt
- Wiederverwendung: Minenfeld (*bleeding edge*)
  - Man benutzt zu viel unausgereifte, supermoderne Technologie und die Software ist deshalb chronisch instabil
    - ein Fall von unnötigem radikalem Vorgehen
- Prozess: Dumm halten
  - Die Entwickler werden ausdrücklich gegen jeden Kontakt mit Benutzern abgeschirmt und entwickeln deshalb nie ein brauchbares Verständnis für die Anforderungen und Prioritäten

Beispiele für Anti-Muster (3)

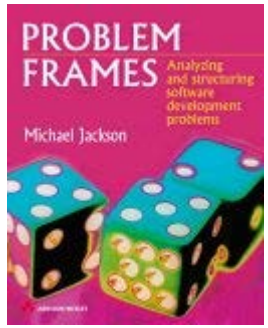
- Prozess: Pseudo-Experten-Herrschaft (*blowhard jamboree*)
  - Entscheidungen (vor allem Technologieauswahl) werden zu sehr auf Basis irgendwelcher Presseberichte getroffen anstatt auf Basis lokal vorhandenen Sachverstands
  - Entwickler müssen unentwegt dem Management Auskunft geben
- Projektmanagement: Todesmarsch (*death march*)
  - Zeitplan und Budget sind von vornherein so knapp, dass ein Erfolg praktisch unmöglich ist
  - Auslöser ist oft das Muster *Blendwerk (vaporware)*: Etwas noch nicht existentes als existent verkaufen
- Kommunikation: Schriftglaube
  - Verwendung von z.B. Email als Kommunikationsmedium, wo mündliche Verständnisklärung (inhaltlich) oder sensibles Kommunizieren (zwischenmenschlich) nötig wäre

Beispiele für Anti-Muster (4)

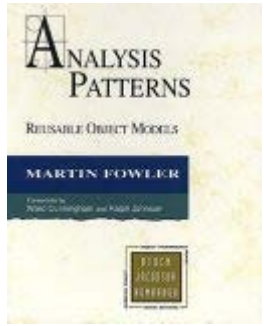
- Architektur: Entwurf per Komitee (*design by committee*)
  - "Zu viele Köche verderben den Brei" und führen zu einer Architektur, die in keiner Hinsicht wirklich gut ist
- Architektur: Conways Gesetz (*Conway's law*)
  - Die Architektur eines großen Systems bildet die bestehende Organisationsstruktur der am Bau beteiligten Gruppen ab
    - http://www.melconway.com/research/committees.html
  - (Kann, klug eingesetzt, auch ein positives Muster sein!)
- Entwurf: Krake (*the blob*)
  - Es gibt ein Modul, das an fast allem irgendwie beteiligt ist, und viel zu viele Kopplungen und Aufgaben hat
  - Überkomplexe Schnittstellen sind auch bekannt als *Eierlegende Wollmilchsau* (unübersetzbar) oder etwas schwächer als *Schweizer Messer (swiss army knife)*

Quellen: Anforderungsmuster

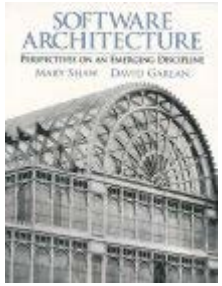
- Michael Jackson: *"Problem Frames: Analyzing and Structuring Software Development Problems"*, Addison Wesley 2001
  - Identifiziert einige grundlegende Problemklassen



- Martin Fowler: *"Analysis Patterns: Reusable Object Models"*, Addison Wesley Longman 1997
  - wie vorhin beispielhaft gesehen
  - Achtung: Das Buch benutzt eine Vor-UML-Notation, bei der Rollennamen immer genau auf der anderen Seite der Assoziation stehen als bei UML üblich
  - Aktualisierungen und Ergänzungen siehe http://www.martinfowler.com/articles.html#ap
  - Die beschriebenen Beispiele entstammen http://www.martinfowler.com/apsupp/accountability.pdf



Quellen: Architektur



- Mary Shaw, David Garlan: "Software Architecture: Perspectives on an Emerging Discipline", Prentice Hall 1996
  - Erstes Buch zum Thema Architekturstile und –muster
  - Nur wenige Muster, aber konzeptuell sehr gute Einführung

Shaw

- Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: *"Pattern-orientierte Softwarearchitektur: Ein Pattern-System"*, Addison-Wesley 1998
  - "Pattern-oriented software architecture: a system of patterns"
  - Architektur-, Entwurfs- und Kodiermuster

Quellen: Entwurfsmuster

- Martin Fowler: *"Patterns für Enterprise Application-Architekturen",* mitp 2003
  - *"Patterns for Enterprise Application Architectures"*, Addison Wesley 2000
  - schöne breite Sammlung für Informationssysteme

- Erich Gamma, Richard Helms, Ralph Johnson, John Vlissides: *"Entwurfsmuster"*, 1995
  - der Klassiker: das "Gang-of-Four"-Buch (GoF)
  - Allgemeine Muster und gute Einführung in OO-Entwurf

- Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann: *"Pattern-orientierte Softwarearchitektur: Muster für nebenläufige und vernetzte Objekte"*, dpunkt 2002
  - siehe oben die Mustersprache

Quellen: Benutzbarkeitsmuster

- http://www.cmis.brighton.ac.uk/research/patterns/home.html
  - Grundlegende Regeln für die Interaktionsgestaltung

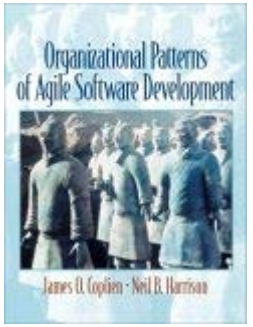
- Len Bass, Bonnie E. John, Jesse Kates:
  "Achieving Usability Through Software Architecture",
  CMU/SEI-2001-TR-005
  - Quelle für vorhin beschriebene Muster
  - http://www.sei.cmu.edu/publications/documents/01.reports/01tr005.html

Bass

John

?

Quellen: Prozessmuster

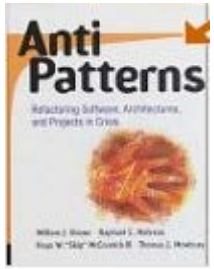


- James Coplien, Neil Harrison: "*Organizational Patterns of Agile Software Development*", Pearson Prentice Hall 2005
  - wie oben beispielhaft gesehen
  - Sehr gutes Buch für Projektmgmt. und Prozessmgmt.
    - (und gar nicht nur speziell für agile Prozesse)
- Inhalt:
  - Project Management Pattern Language 26 Muster
  - Piecemeal Growth Pattern Language 32 Muster
  - Organizational Style Pattern Language 23 Muster
  - People and Code Pattern Language 23 Muster
  - Organizational Principles 18 Erwägungen
  - Antropological Foundations
  - Case Studies 2 Fallstudien
  - Summary Patlets Kurzbeschreibungen

Coplien

Quellen: Anti-Muster

- William Brown, Raphael Malveau, Hays McCormick: "Anti-Patterns: Refactoring Software, Architectures and Projects in Crisis", John Wiley 1998
  - Die meisten der oben zitierten Antimuster sind in diesem Buch genauer beschrieben



- http://en.wikipedia.org/wiki/Antipattern
  - enthält auch eine Liste von Mustern
- http://c2.com/cgi/wiki?AntiPattern
  - http://c2.com/cgi/wiki?AntiPatternsCatalog

Danke!