

Vorlesung Software-Reengineering

Dr. Rainer Koschke
Universität Stuttgart

15. Oktober 2003

Organisatorisches

- Erreichbar: Zimmer 1.218, Tel. 7816-206, koschke@informatik.uni-stuttgart.de
- Sprechstunde nach Vereinbarung
- Literatur: Semesterapparat; Folien zur Vorlesung und verwendete Artikel
<http://www.iste.uni-stuttgart.de/ps/Lehre/reengineering> (Folien zur Vorlesung)
- Reengineering-Bibliographie:
<http://www.iste.uni-stuttgart.de/ps/reengineering>

Organisatorisches

- Vorlesung: mittwochs, 11:30 - 13:00 Uhr, Raum V38.04
- Übungen: dienstags, zweiwöchig, 11:30 - 13:00 Uhr, Raum V0.463,
 - erster Termin am 28. Oktober 2003
 - Übungen sind freiwillig
 - in den Übungen werden Aufgaben besprochen, aber auch Werkzeuge vorgestellt

Lernziele

- Grundlegende Begriffe
- Übersicht über die Gebiete des Reengineerings
- Abgrenzung zum traditionellen Software Engineering

Lehman und Beladys Hypothesen

Software-Evolution

- Gesetz des fortgesetzten Wandels
- Gesetz der ansteigenden Komplexität
- . . .

⇒ ständige Anpassung erforderlich

⇒ Komplexität muss kontrolliert und begrenzt werden

Wunsch

- Gewählte Lösung antizipiert mögliche Änderungen.
- Änderungen werden auf der adäquaten Ebene vorgenommen.
- Dokumentation wird mitgeführt.

Wirklichkeit

- Die Zukunft lässt sich nur begrenzt vorhersagen.
- Ursprüngliche Systemstruktur wird ignoriert.
- Dokumentation ist unvollständig oder obsolet.
- Mitarbeiter verlassen das Projekt (und mit ihnen verschwindet das ganze Wissen).

Legacy System

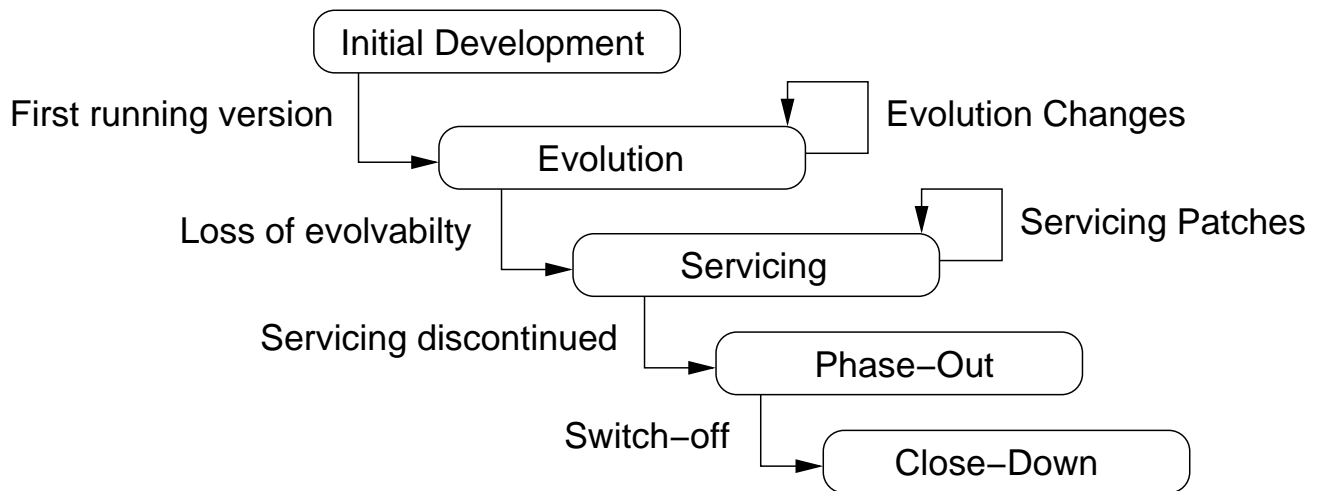
Legacy:

“A sum of money, or a specified article, given to another by will; anything handed down by an ancestor to a predecessor.”

– Oxford English Dictionary

Legacy System: Software-System, das erbt wurde und einen Wert darstellt.

Staged Software Life Cycle Model



Begriffe

- Software-Wartung
- Software-Evolution
- Reengineering
- Software-Reengineering
- Business-Process-Reengineering
- Renovation
- Reclamation
- Reverse Engineering
- Restrukturierung
- Wrapping

Software-Wartung

ANSI/IEEE Standard 729-1983:

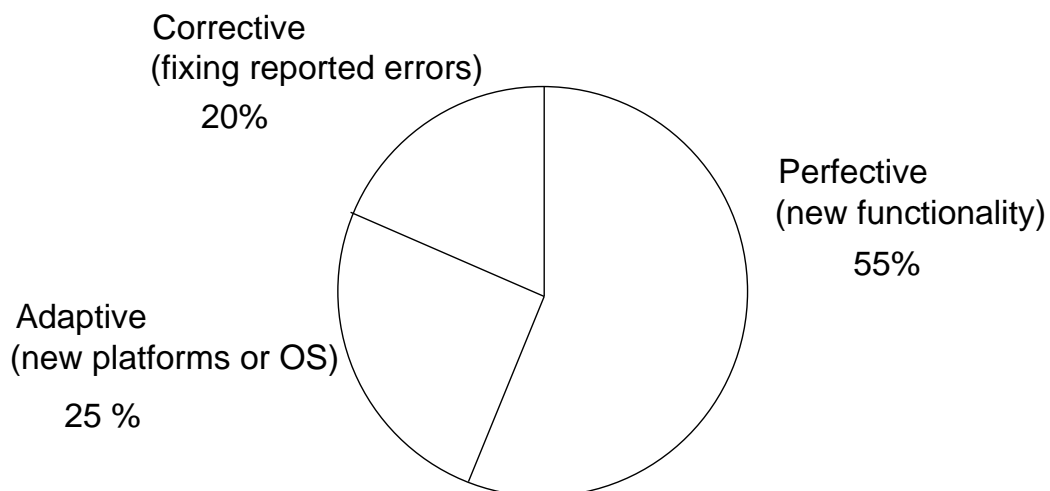
“Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment.”

Häufiger Sprachgebrauch: Änderungen am System nach dessen Auslieferung.

Schließt Anpassungen an neue Anforderungen ein.

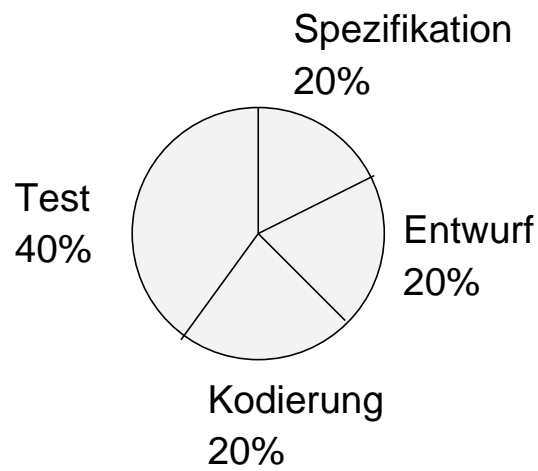
Besserer Begriff hierfür: Software-Evolution.

Aufwand für Software-Wartung



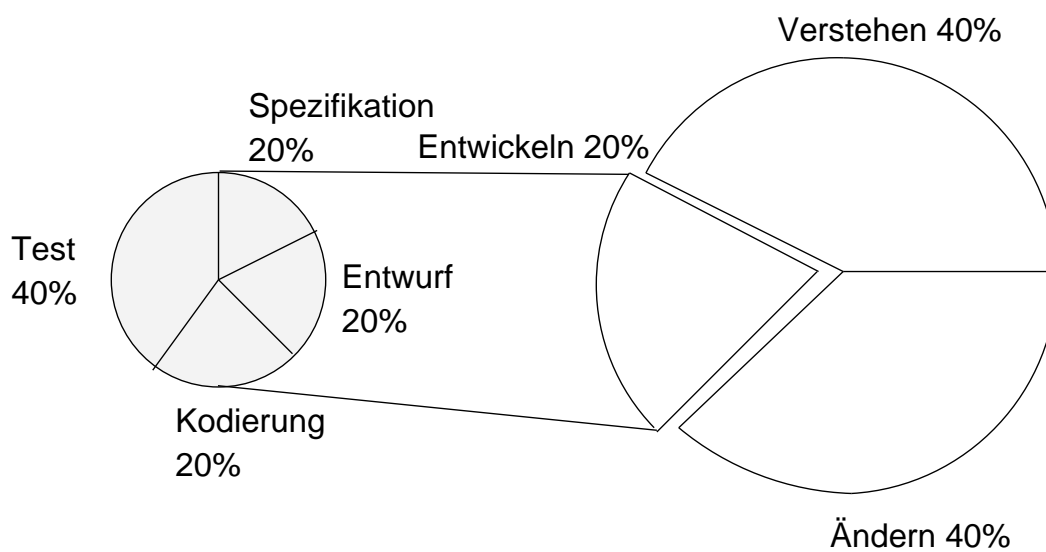
– Lientz und Swanson, 1980

Aufwand im Software-Lifecycle



– Boehm, 1981

Aufwand im Software-Lifecycle



– Nosek und Palvia, 1990

Aufwand für Software-Evolution

US Air Force System (Boehm, 1975):

- \$ 30 / Statement bei Erstentwicklung
- \$ 4000 / Statement in der Wartung

Beseitigung des Jahr-2000-Problems (geschätzt von Cassell, 1997)
500.000.000.000 - 1.000.000.000.000 DM

Reverse Engineering

License Restrictions:

“Customer may not reverse engineer, disassemble, decompile, or translate the Software, or otherwise attempt to derive the source code of the Software.”

Reverse Engineering

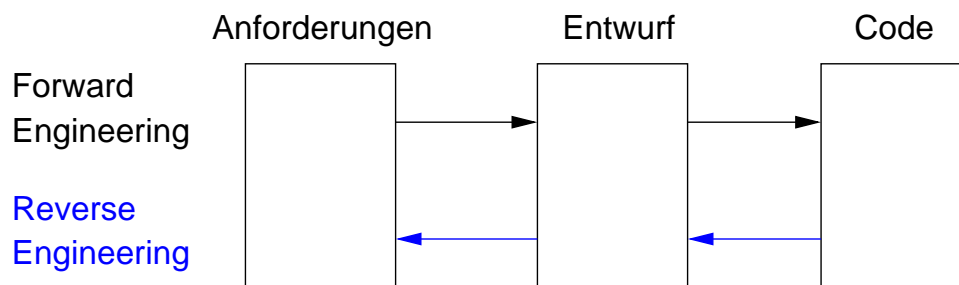
“To me the flow of time is irrelevant. You decide what you want. I then merely make sure that it has already happened.”

The Hitch Hiker's Guide to the Galaxy

Reverse Engineering (Chikofsky und Cross, 1990)

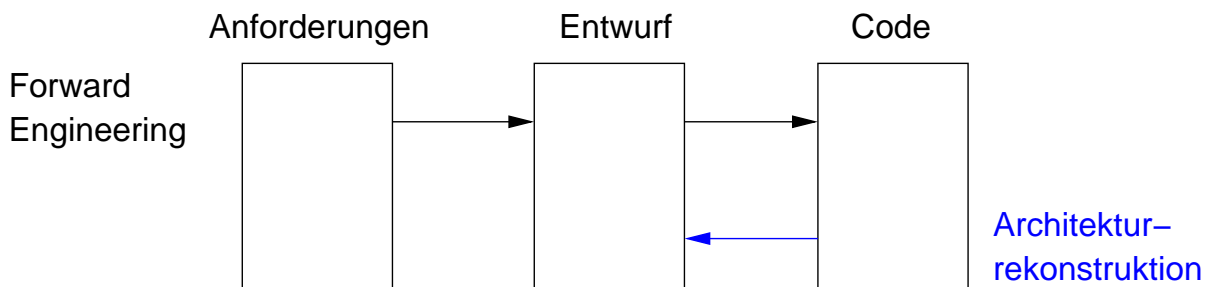
Identifikation der Systemkomponenten und deren Beziehungen.

Ziel ist die Beschreibungen des Systems in einer anderen Form oder auf höherem Abstraktionsniveau.



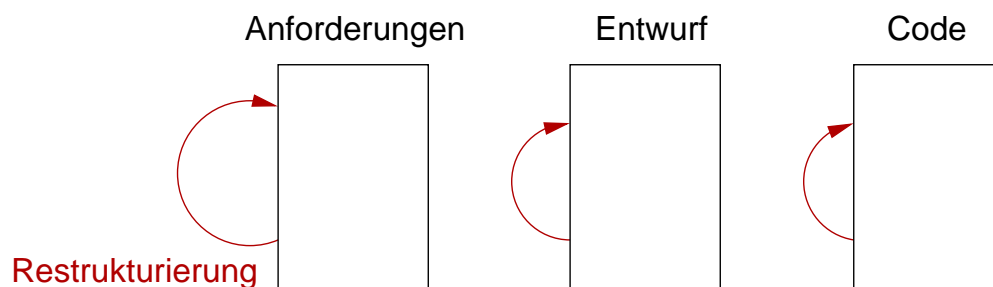
Architekturrekonstruktion

Reverse Engineering mit dem Ziel, eine Beschreibung der Architektur des Systems zu erstellen.



Restrukturierung (Chikofsky und Cross, 1990)

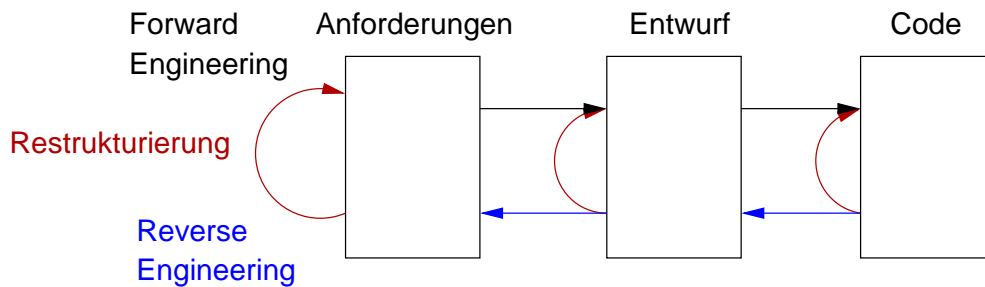
Transformation einer Repräsentation in eine andere auf derselben Abstraktionsebene, ohne Änderung der Funktionalität des Systems.



Reengineering (Chikofsky und Cross, 1990)

Untersuchung (Reverse Engineering) und Änderung des Systems, um es in neuer Form zu implementieren.

Synonyme: Renovation, Reclamation.



Reengineering-Varianten

Reines Reengineering:

- das System soll lediglich restrukturiert werden
- keine Funktionalität kommt hinzu / wird geändert

Erweiterndes Reengineering:

- System wird zunächst analysiert und/oder restrukturiert, um dann Funktionalität zu ändern oder hinzuzufügen

Wrapping

Das System erhält eine neue Schnittstelle, bleibt aber ansonsten unangetastet.

Interimslösung, wenn System bald ausgewechselt werden soll.

Notwendig, wenn das System Subsystem per Subsystem geändert werden muss.

Oft eingesetzt, um zeichenorientierte Anwendungen mit einer graphischen Benutzerschnittstelle zu versehen.

Organisatorische Gründe:

- “altes” Wartungspersonal behält Kontrolle über Wartung “ihres” Systems
- “junges” Wartungspersonal hat “moderne” Sicht

Business Process Reengineering

“Business process reengineering is the search for, and the implementation of, radical change in business process to achieve breakthrough results.”

– T.A. Stewart, Fortune Magazine'93.

Business Process Reengineering

Etwas sachlicher:

- Wiedergewinnung der tatsächlichen Abläufe der Geschäftsprozesse (Workflow) (z.B. Bestellwesens, Auftragsabwicklung, etc.)
- Überarbeitung und Neudefinition der Abläufe

Ziele des Reverse Engineerings

- Kontrolle der Komplexität
- Gewinnung alternativer Sichten
- Wiedergewinnung verlorener Information
- Erkennung von Seiteneffekten
- Schaffung höherer Abstraktionen
- Unterstützung von Wiederverwendung

Häufige Reengineering-Aufgaben

- Plattformanpassung
- Änderung der Programmiersprache
 - neuer Standard, neue Sprache, neues Paradigma
- Benutzerschnittstelle zeichenorientiert hin zu graphisch orientiert
- Mainframe hin zu Client-Server-Architektur
- Datenbankumstellung

Präventive Maßnahmen, wie z.B. Verbesserung des Information Hidings, Remodularisierung, sind eher selten.

Mass Changes

Änderungen, die weite Teile des Codes bzw. sehr viele Systeme betreffen.

- Einführung des Euros
- Legacy to Internet Interoperability (Electronic Commerce)
- Änderung von Repräsentationsformen:
 - Y2K Problem
 - Erweiterung des Bar-Codes
 - Unix-Datum

Reengineering in der Praxis

Gegenwärtig zur Verfügung stehende Werkzeuge:

- grep
- symbolische Debugger
- Cross-Reference-Tools (fertige Parser, die Basisinformationen extrahieren; z.T. mit Visualisierung durch Graphen)
- programmierbare Analyseumgebungen basierend auf abstrakten Syntaxbäumen (z.B. Refine von Reasoning Systems, DMS von Semantic Designs, RainCode)

Übersicht über diese Vorlesung

- Programmanalysen und -repräsentationen
- Program-Slicing
- Erkennung duplizierten Codes
- Refactoring und Remodularisierung
- Begriffsanalyse
- Analyse und Restrukturierung von Vererbungshierarchien
- Merkmalsuche
- Mustersuche
- Reengineering-Projekte

Software Engineering

“Software Engineering is reengineering on the empty system.”

“Is it?”

Unterschiede Forward Eng. / Reengineering

Forward Engineering auf grüner Wiese

- Problem noch unklar
- Aussagen über Aufwand, Dauer, Zuverlässigkeit etc. sind schwierig
- System existiert nicht
 - Entwurf hat viele Freiheiten
 - im sauberen Entwurf gibt es keine versteckten Abhängigkeiten

Reengineering

- Problem weitgehend klar
- Idealerweise: Daten aus der Vergangenheit existieren, die Grundlage für Schätzungen darstellen
- System existiert
 - Genaue Struktur/Qualität bekannt?
 - Lösung ist durch existierendes System beschränkt
 - Änderungen können globale Auswirkungen haben (viele versteckte Abhängigkeiten)

Software Engineering & Reengineering

Reengineering beginnt oft bereits während der Erstentwicklung:

- neue Anforderungen treffen ein
- Missverständnisse und Unklarheiten werden sichtbar
- der Entwurf hat sich als unzureichend erwiesen
- Integration anderer Komponenten macht Umstrukturierungen notwendig

Weiterführende Literatur

Chikofsky, E.J., Cross II, J. H., „Reverse Engineering and Design Recovery: A Taxonomy“, IEEE Software, January (1990).

definiert Terminologie; ist die begriffliche Grundlage

Baumöl et al. 'Einordnung und Terminologie des Reengineering', Informatik-Spektrum 19:191-195, Springer Verlag, 1996.

führt z.T. deutsche Begriffe ein