# Software Wartung und Evolution

**Harald Gall**
Institut für Informatik
Universität Zürich
http://seal.ifi.uzh.ch/


Universität Zürich

s.e.a.l.
software evolution & architecture lab

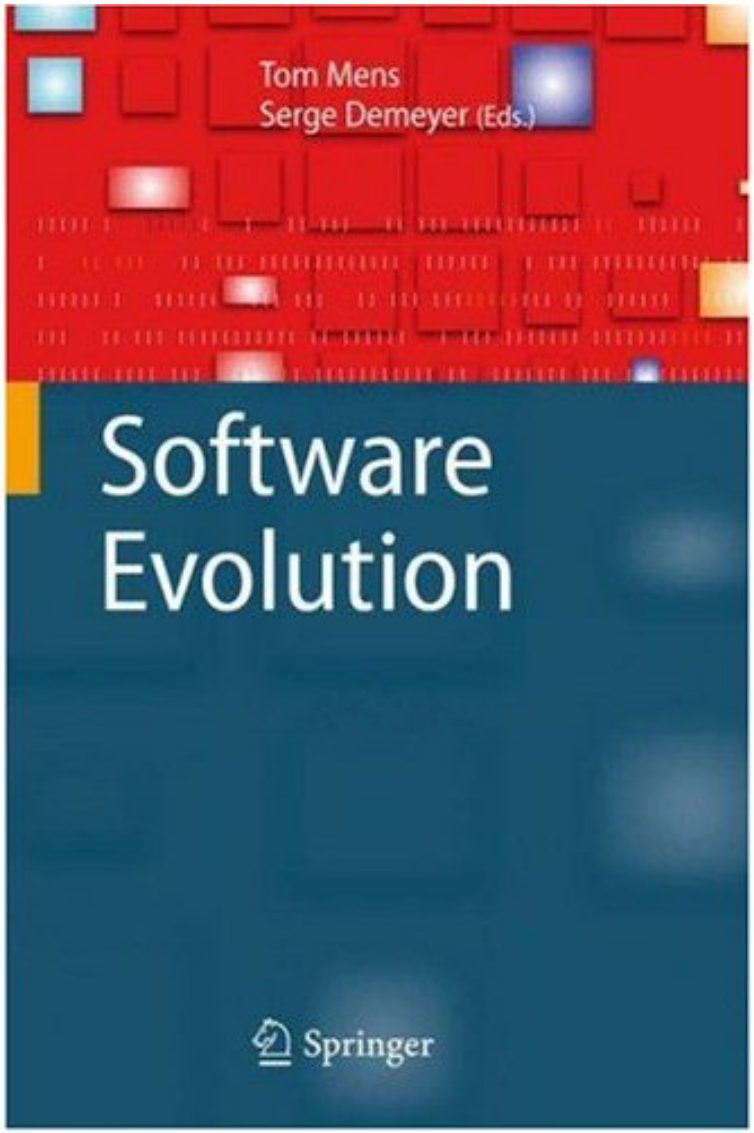
# Organisatorisches

- LV Info
  - Vorlesung mit Übungen
  - Freitag, 10:15 - 11:45 Uhr, BIN 2.A.10
- Vorlesungstermine
  - Ab 26. Februar, jeweils Freitag mit Ausnahmen!
- Voraussetzungen
  - BSc Informatik: Assessmentstufe, Module Software Engineering und Software-Praktikum
- Anrechenbarkeit
  - Wahlvorlesung  in Informatik, Bestandteil von themenübergreifenden Lehreinheiten

# Organisatorisches

- Leistungsnachweis
  - Erfolgreiche Bearbeitung von Übungsaufgaben während des Semesters und Teilnahme an der schriftlichen Endklausur
- Prüfung
  - schriftliche Klausurarbeit
  - Bei geringer  Teilnehmerzahl:
    mündliche Prüfungen (Termine nach Vereinbarung)
- LV-Web-Seite
  - http://seal.ifi.uzh.ch/evolution/

# Book



# Background

- Software Design & Architecture
- Reengineering
- Refactoring
- Legacy System
- Separation of Concerns
- Design Patterns, Architecture Patterns
- AST

# Inhalt der Vorlesung: Überblick

- Was versteht man unter Software Wartung / Software Evolution / Wartbarkeit?
- Was sind die speziellen Probleme?
- Welche adäquaten Technologien, Prozesse und Tools gibt es, um diesen zu begegnen?
- Was sind die *Best Practices* der Software Wartung?
- Wie managed man Software Wartung?

# Inhalt der Vorlesung: Themen

- Software Wartung: Motivation, Definition, Arten, Probleme
- Aspekte, Aktivitäten, Wartungskrise, Legacy Systeme, Reverse Engineering
- Restructuring, Re-Engineering, Re-Architecting, Architecture Reflexion, *Organisation der Wartung*: Life Cycle Modelle
- Defect Tracking, Software Configuration Management, Produktivstellung, Dokumentation, *Software Evolution:* e-type Programs, Laws, Formal Approach
- Change Patterns, Ensuring Maintainability: Refactoring, Design for Change (e.g. OO, AOP, SOC),
- Best Practices (e.g. Literate Programming), Wartung und Test, Program Comprehension, Impact Analysis, Change Propagation, *Software Wartung im Unternehmen*: Rollen, Gewährleistung, Software-Wartungsverträge

# Teil 1

- Inhalte
  - Motivation: Software Wartung und Evolution
  - Abgrenzung Software Wartung zu Entwicklung und Evolution
  - Definition Software Wartung
  - Probleme der Software Wartung

# Literatur zum Thema

- Suche nach dem Begriff „Software Engineering“ bei amazon.de, Kategorie “Englische Bücher”
  - Treffer: 1405 (2004: 1078)
  - Bester” Treffer: “Software Engineering” von I. Sommerville (2004)
- Suche nach „Software Maintenance“
  - Treffer: 92 (2004: 92)
  - “Bester” Treffer: “Troubleshooting your PC” (1994)
    - 2004: „Advances in Software Maintenance Management: Technologies and Solutions” – nicht unter den ersten 10 Treffern
- Suche nach „Software Evolution“
  - Treffer: 20 (2004: 15)
  - 1. Treffer: “The Distance Education Evolution: […]” (2003)
    - 2004: “Artificial Life Playhouse: Evolution at your Fingertips” – Treffer 3
  - 2. Treffer: “Software Evolution: The SW Maint. Challenge” (1988!)
    - 2004: “Successful Evolution of Software Systems” (2003) – Treffer 6

# Annäherung an den Begriff „Software Wartung“

- Software Wartung hat mit dem geläufigen Begriff der technischen Wartung wenig gemein
  - Software hat keine Verschleissteile
  - Software zeigt keine Abnützungserscheinungen („wear-out“)

# Annäherung an den Begriff „Software Wartung“

- Software gilt im Gegensatz zu physischen technischen Artefakten als „leicht“ änderbar

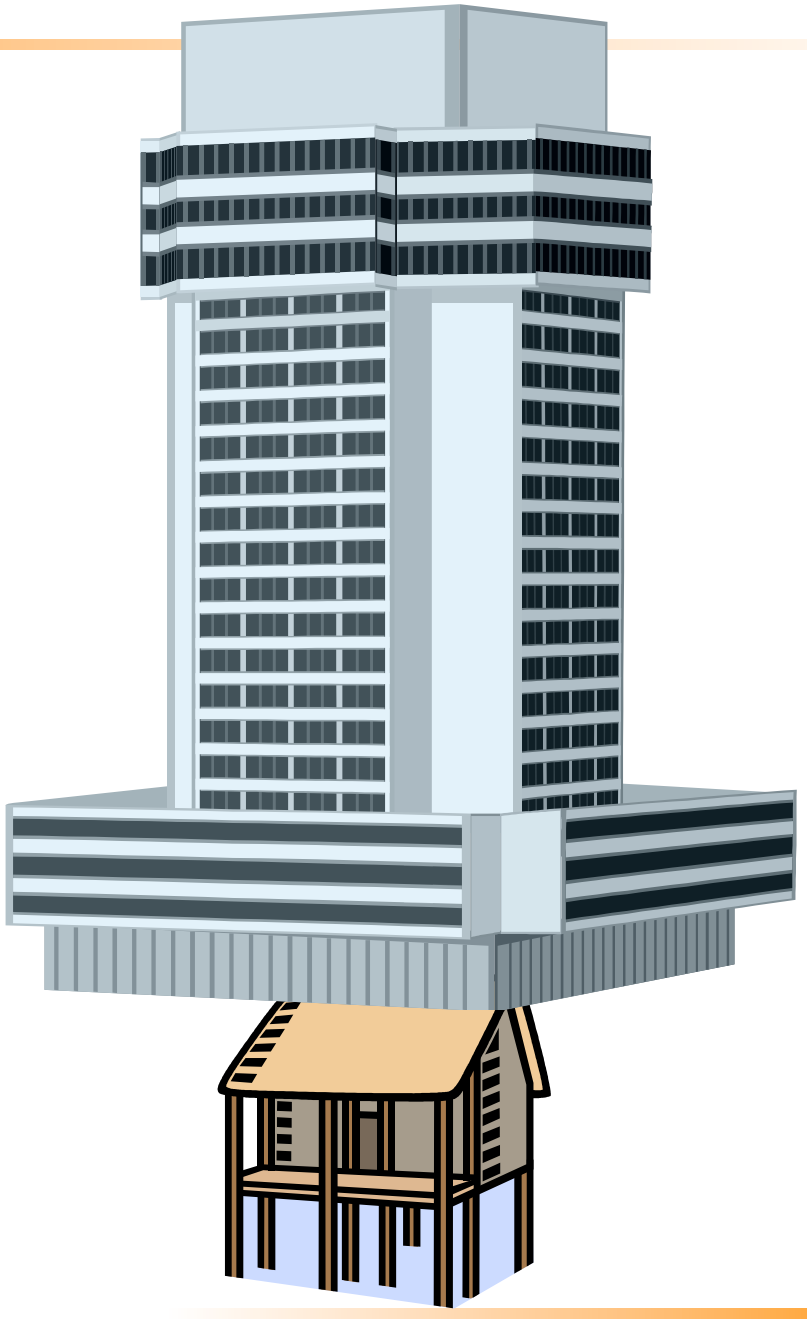
# Software ist leicht änderbar?

```
/**
 * Wrong name: should be getWettSportArtToSportArtIDAndWettArtID
 */
 public static WettSportArt getWettSportArtToWettArtIDAndLigaID (Integer
    iSAID, Integer iWAID) throws ExceptionPersist {

   Vector v;
   WettSportArt ws = new WettSportArt();
   ws.setSportArtID(iSAID);
   ws.setWettArtID(iWAID);
   v = ws.search();

   [...]
```

# Software ist leicht änderbar?

```
for(int j=0;j<vtz.size();j++) {

    tzkbez = ((TipZeile)vtz.elementAt(j)).getTipKurzBezeichnung();
    iZeichPos = tzkbez.indexOf(':');

    if (tzkbez.substring(0,iZeichPos).equals(tzkbez.substring(iZeichPos+1)))
      ivSortTip = 1;
    else if (tzkbez.substring(0,iZeichPos).compareTo(tzkbez.substring(iZeichPos+1))>0)
      ivSortTip = 0;
    else
      ivSortTip = 2;

    if (ivSortTip == 0) iZeichPos = 0;
    else iZeichPos++;

    int i;
    for(i=0;i<vSortTip[ivSortTip].size();i++) {
      if (((TipZeile)vSortTip[ivSortTip].elementAt(i)).getTipKurzBezeichnung().substring(iZeichPos).compareTo(tzkbez.substring(iZeichPos))
       >0)
        break;
    }

    if (i<vSortTip[ivSortTip].size())
      vSortTip[ivSortTip].insertElementAt(vtz.elementAt(j),i);
    else
      vSortTip[ivSortTip].add(vtz.elementAt(j));
}
```

# Software ist leicht änderbar?



# Annäherung an den Begriff „Software Wartung“

- „Programs, like people, get old“
- „Software aging will occur in all successful products“
  - David Lorge Parnas in „Software Aging“ [Parnas 1994]
- Es ist einsichtig, dass, was altert, irgendwie up-to-date gehalten werden muss.
- Die Frage ist: Wie altert Software und warum?

# Two Causes of Software Aging

- The first is caused by the failure of the product‘s owners to modify it to meet changing needs
  - „Lack of movement“
- The second is the result of the changes that are made
  - „Ignorant surgery“

[D. L. Parnas, 1994]

# Symptoms and Costs of Software Aging

- Inability to keep up
  - “As software ages, it grows bigger“
    - More code to change
    - More difficult to find routines that must be changed
- Reduced performance
  - More machine resources are needed
  - Poor design causes performance bottlenecks
- Decreasing reliability
  - „As the software is maintained, errors are introduced“

[D. L. Parnas, 1994]

# Preventive Medicine

- Design for success (aka „Design for change“)
  - Information hiding, abstraction, separation of concerns (SOC), data hiding, object orientation, …
- Documentation
  - Problem: Most documentation is ignored because not being accurate
- Second opinions – Reviews
  - Reviews often are neglected because of time pressure

[D. L. Parnas, 1994]

# „Software Geriatrics“

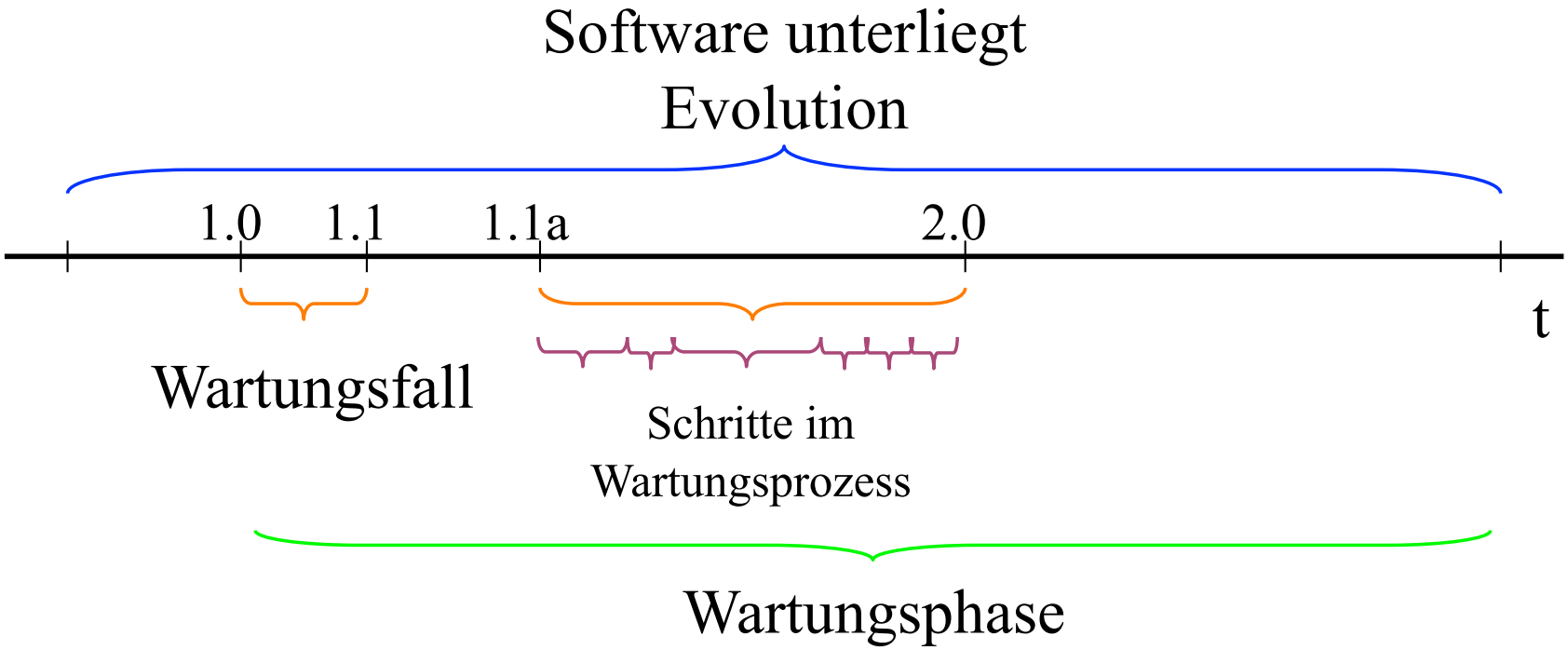
- Stopping deterioration
  - Requires techniques and resources!
- Retroactive documentation
  - Lack of formal basis and influence of short-term interests
- Retroactive incremental modularization
- Amputation
- Restructuring

[D. L. Parnas, 1994]

# Software Wartung vs. Software Evolution

- Software Wartung
  - bezeichnet als *Tätigkeit* eine Änderung an einem Softwaresystem nach dessen Auslieferung (aka Wartungsfall)
  - bezeichnet als *Prozess* die Schritte, die in einem Wartungsfall sequentiell durchzuführen sind
  - bezeichnet als *Phase* den Abschnitt des Lebenszyklus von der Auslieferung bis zur Stilllegung
- Software Evolution
  - bezeichnet den *Prozess* der Veränderung eines Softwaresystems von der Erstellung bis zur Stilllegung
  - umfasst: Entwicklung, Wartung, Migration, Stilllegung

# Software Wartung vs. Software Evolution



# Warum Evolution?

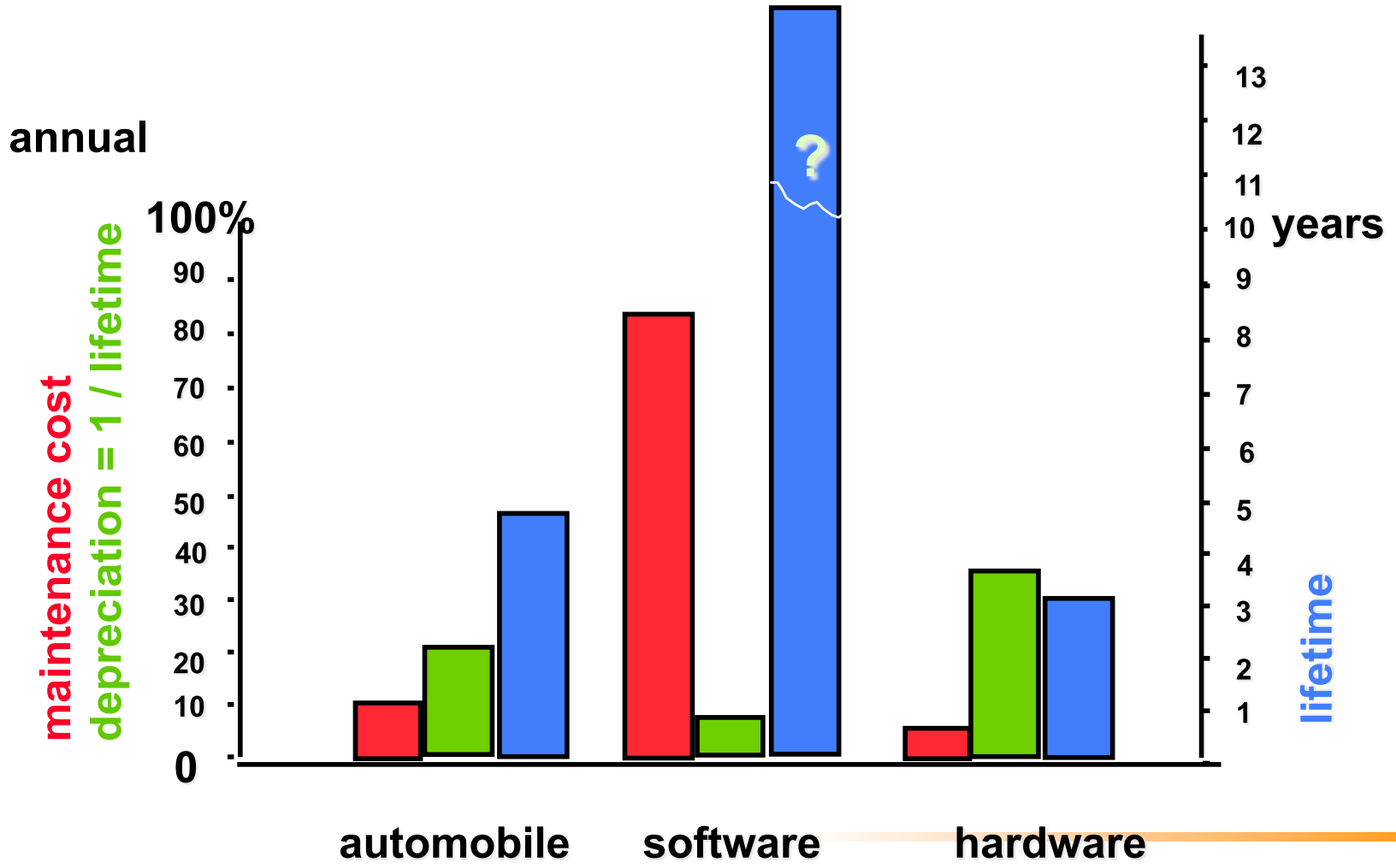
- Viele Software Systeme bilden Geschäftsprozesse der realen Welt nach
- Geschäftsprozesse unterliegen ständigen Änderungen
  - passiv durch Adaption an neue Gegebenheiten (neue rechtliche Gegebenheiten, Marktsituation, Euro, Basel II, …)
  - aktiv durch die Einführung neuer Produkte, neuer Prozesse (Business Process Reengineering (BPR))
- Daher muss die Software laufend an die sich ändernden Geschäftsprozesse angepasst werden

# Warum Beschäftigung mit Software Wartung/Evolution?

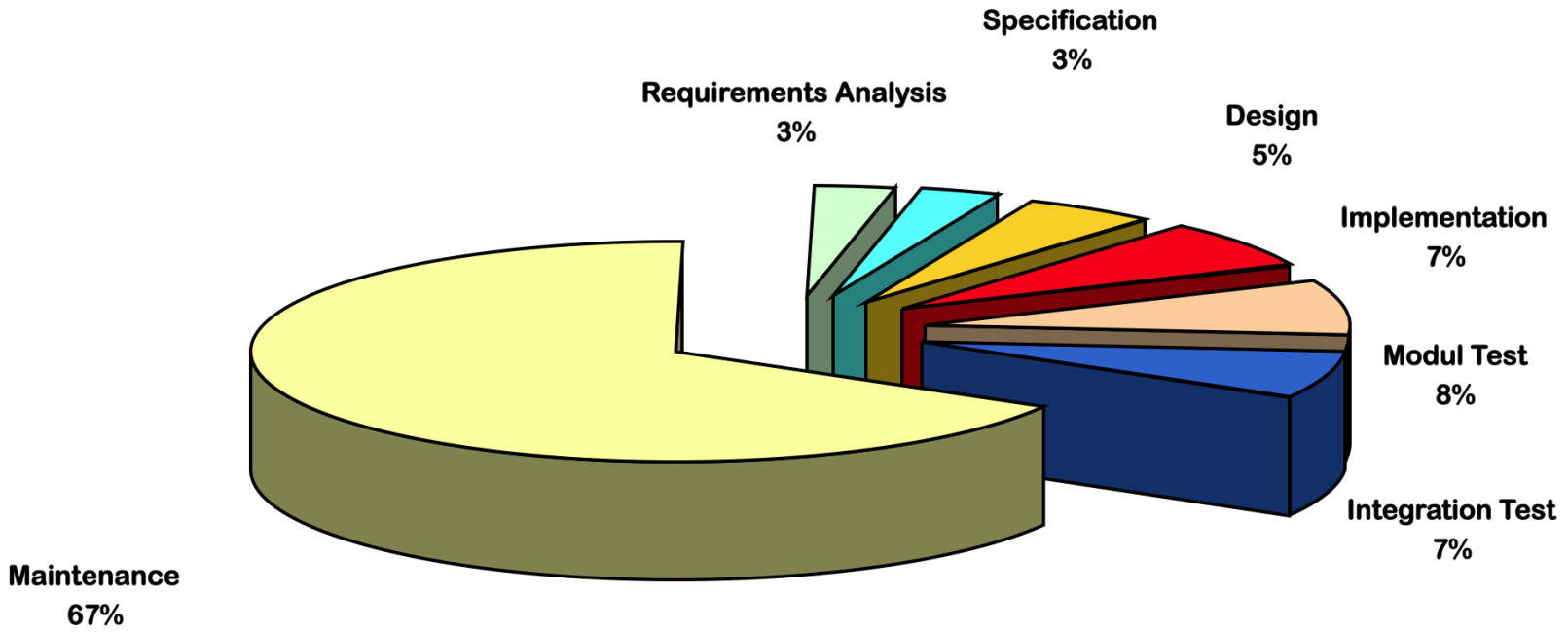
- „Nevertheless, the industrial track record raises the question, why, despite so many advances, [...]
  - satisfactory functionality, performance and quality is only achieved over a *lengthy evolutionary process*,
  - software maintenance *never ceases* until a system is scrapped
  - software is still generally regarded as the *weakest link* in the development of computer-based systems“.

  [Lehman et al., 1997]

# Software Wartung im Vergleich



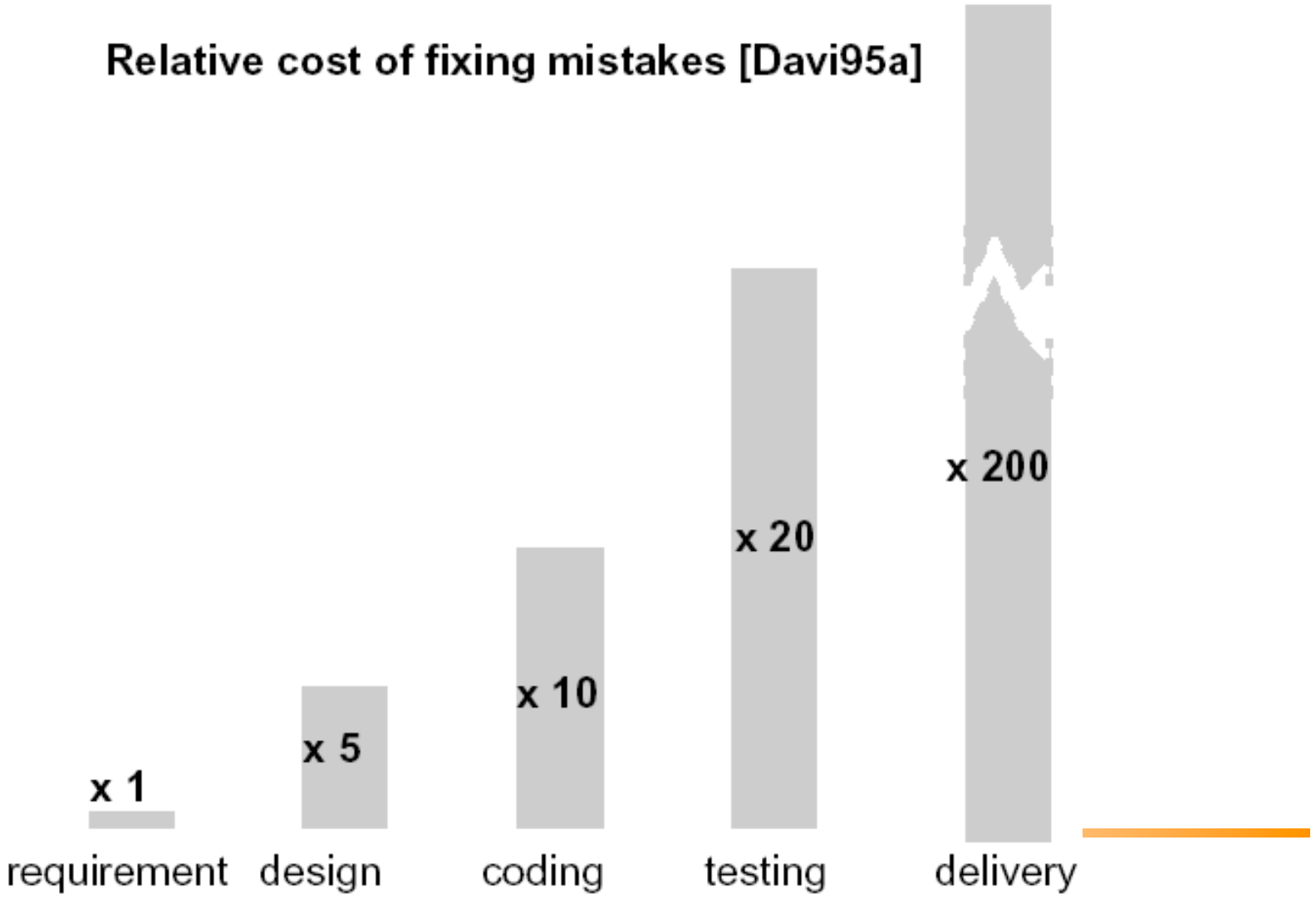
# Cost Distribution in Software Life-Cycle



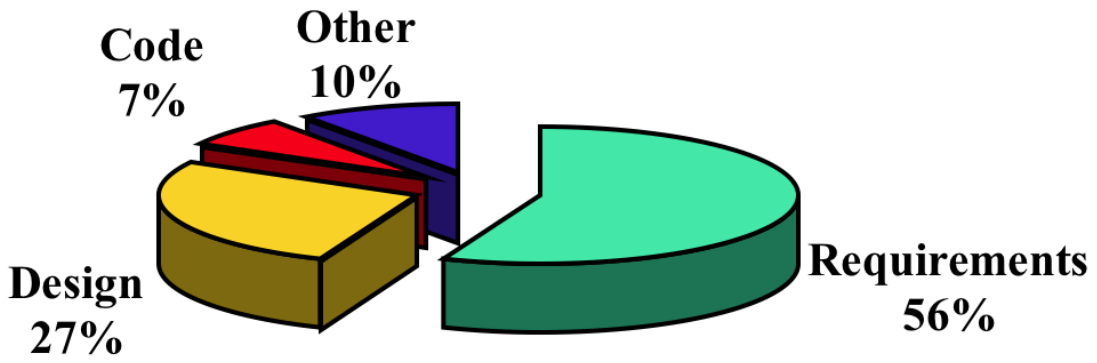

**Cost Distribution in the Software-Life-Cycle**

Source: Principles of Software Engineering and Design, Zelkovits, Shaw, Gannon 1979

# Cost of fixing bugs per phase

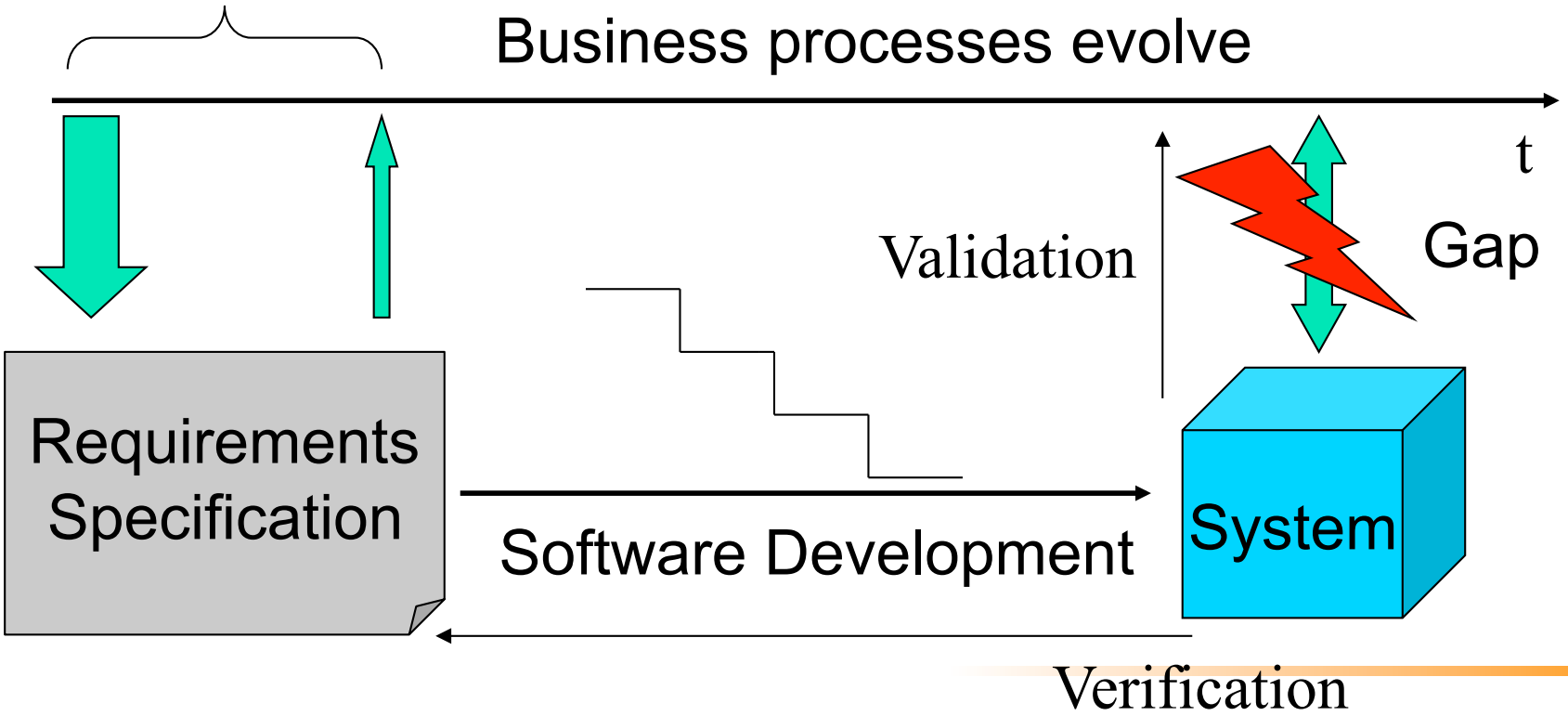


# Distribution of Bugs



# Classic Approach to SW Development

- „Requirements Fixing“



# Software maintenance vs. development

- Maintenance is similar to software development
  - Some unique skills and processes are employed:
    - Have intimate knowledge of system structure and content
    - Perform impact analysis and know the ripple effect
    - Problem solving skills
      - „Programmers have become part historian, part detective, and part clairvoyant.“ (Corbi 1989)
    - Track and control changes
    - Maintenance Outsourcing
    - Maintenance Cost Estimation

# Software Wartung: Definition

# Definition: Software Wartung

- Nach IEEE Std. 610.12-1990 bzw. IEEE Std. 1219-1998
  - Software Wartung ist die Modifikation eines Software-Produktes oder einer Komponente, nach der Auslieferung, mit dem Zweck
    - der Fehlerkorrektur
    - der Verbesserung der Performance oder anderer Systemattribute
    - der Adaptierung an eine geänderte Umgebung

# Definition: Software Wartung

- Nach Barry Boehm
  - “The process of modifying existing operational software while leaving its primary function intact”
- Abstrakt
  - „Preserve the value of software over time“
- Center for Software Maintenance
  - **Software maintenance** is the set of activities, both technical and managerial, that ensures that software continues to meet organizational and business objectives in a cost-effective way.

# Allgemeinere Definition [SWEBOK]

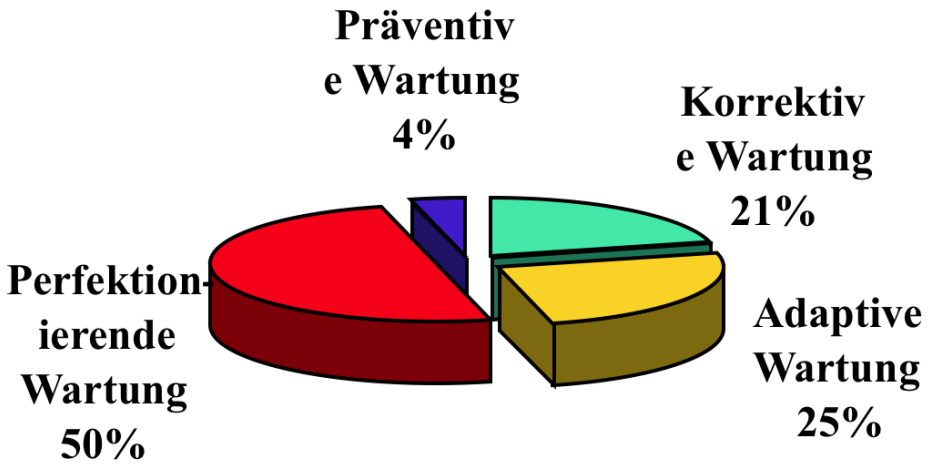
- Software Maintenance
  - The totality of activities required to provide cost-effective support to a software system.
  - Activities are performed during the pre-delivery stage as well as the post-delivery stage.
  - Pre-delivery activities include planning for the post-delivery operations, supportability, and logistics determination.
  - Post-delivery activities include software modifications, training, and operating a help desk.

*Source: Software Engineering Body of Knowledge, http://www.swebok.org/*

# Arten der Software Wartung

- (1) Korrektive Wartung
  - „Bug fixing“; reaktive Natur
- (2) Präventive Wartung
  - Finden von latenten Fehlern, bevor sie effektive Fehler werden
- (3) Adaptive Wartung
  - Neue Hardware, Betriebssystem; neue Anforderungen
- (4) Perfektionierende Wartung
  - Verbesserungen in Performance und Wartbarkeit (Restructuring, Reverse Engineering, Dokumentationspflege, etc.)
- Corrections = (1) + (2)
- Enhancements = (3) + (4)

# Arten der Software Wartung




d.h. ca. 80% sind keine korrektiven Aktivitäten!

# Warum ist Software Wartung nicht trivial?

Motivation

# Typische Eigenschaften von Software

- Programme sind nur selten in sinnvolle Modulstrukturen aufgeteilt, und wenn, ist diese Aufteilung meist sehr willkürlich
- Redundanzen in Daten bzw. Funktionen sind ein steter Bestandteil eines Programms
- Die Sichtbarkeitsbereiche von Daten und Funktionen sind meist weiter ausgedehnt aus für das Programm notwendig bzw. überhaupt sinnvoll

# Typische Eigenschaften von Software

- Trace Ausgaben sind spärlich, haben keinen Timestamp und keine Quellenangabe
- Durch Änderungen am Code verändert sich das Laufzeitverhalten durch Gleichzeitigkeitsprobleme („race conditions“) nichtdeterministisch
- Dieselbe Methode wird durch ein Flag für unterschiedliche Berechnungen genutzt
- Methoden bleiben bei der Klasse für die sie ursprünglich geschrieben wurden, bei einer Änderung der Funktionalität werden sie nicht zur am Besten geeigneten Klasse verschoben

# Typische Eigenschaften von Software

- Variablennamen sind semantisch wertlos
  - int work = 0;
- Variablen werden kontext-abhängig verwendet
  - Fall 1: work speichert Kundennummer
  - Fall 2: work speichert Fakturensumme
- Die Dynamik des Programmdurchlaufs ist während der statischen Code Inspektion nicht oder nur schwer ableitbar

# Schwierigkeiten in der Software Wartung

- Missing
  - Development environment (tools, scripts, etc.)
  - Build environment
  - Source code
  - Documentation
  - Design Decisions
  - Domain knowledge
  - Original programmer team / analysts

# Schwierigkeiten in der Software Wartung

- Wartung ist ereignisgesteuert (planbar?)
- In der korrektiven Wartung wird nach Meldung eines Fehlers verlangt, die Ursache so schnell als möglich zu finden und den Fehler zu beseitigen (also quick fix!), trotz des weiterlaufenden Tagesgeschäftes
- Das Wartungspersonal steht daher meistens unter Zeitdruck und das Management und die Dokumentation des Wartungsfalls werden vernachlässigt

# Schwierigkeiten in der Software Wartung

- Laufende Wartung erhöht die „Software Entropie“
  - Verklärt
    - Architektur
    - Design
    - Modularisierung
  - Erhöht
    - Abhängigkeiten („Coupling“)
  - Vermindert
    - Orthogonale Trennung („Cohesion“)

# Schwierigkeiten in der Software Wartung

- Änderungen an einem Software System sind zwar operativ leicht durchführbar, die Schwierigkeit ist aber, die richtige Änderung durchzuführen - und **nur** diese.

# Schwierigkeiten in der Software Wartung

- Viele Probleme der Software Wartung treten erst im Zusammenhang mit großen, alten, komplexen Software Systemen auf, so genannten „*Legacy Systemen*“.
- Diese wurden mit Methoden konzipiert und in Sprachen erstellt, die heute kaum mehr benutzt (und noch weniger gelehrt) werden
  - Strukturierte Analyse, proprietäre Analyseansätze
  - Mumps, CICS, PL/I

# State-of-the-Art: „7x24“

- 7 Tage in der Woche 24 Stunden online
- Hardware Hersteller werben mit 99,999 Prozent Verfügbarkeit ihrer Systeme (entspricht 5 Minuten 20 Sekunden Downtime im Jahr)
- Wann werden dann neue Versionen eingespielt?
- Trend zu Applikationsservern, die Wartung zur Laufzeit unterstützen
  - 2. Instanz mit adaptierter Software fährt hoch
  - Die 2. Instanz übernimmt zur Laufzeit
  - Die ursprüngliche Instanz geht außer Betrieb

# Software Wartung als Tätigkeit

- Die Wartung von Programmen galt von jeher als *unbeliebte* Tätigkeit im Software Life-Cycle
  - Historisch die Arbeit der Ferialpraktikanten bzw. neu rekrutierten Programmierer
- Das heisst
  - wenig Know-How
  - keine Werkzeuge
- Folgen
  - „quick fix“ Modell wird angewandt
  - Wartung führt zu noch mehr Fehlern
  - Wartung kann aufgrund von steigender Komplexität gar nicht mehr durchgeführt werden

# Evolutionäre Probleme der Wartung

- Die Komplexität wächst mit jedem Wartungseingriff
- Daher vergeht zwischen den Wartungseingriffen immer mehr Zeit
- Die Produktivität der Wartungseingriffe sinkt, die Kosten pro Eingriff steigen
- Dies alles geschieht nach Gesetzmäßigkeiten („Laws of Software Evolution“)

# Das Pareto Prinzip im Software Engineering bzw. in der Wartung

- 20% der Requirements bedingen 80% der Komplexität
- 80% des Systems sind in 20% der Zeit fertig gestellt
- 20% des Codes beinhalten 80% der Fehler
- 80% der Fehler werden in 20% der Zeit behoben
- Nach Vilfredo Pareto (1848-1923)
  - Italienischer Ökonom und Gesellschaftstheoretiker
  - Dieses Prinzip besitzt auch in vielen anderen Bereichen Gültigkeit (z.B. Zeitmanagement, Verkauf)

# POEM - David H. H. Diamond

- The fellow who designed it,
  Is working far away;
  The spec's not been updated,
  For many a livelong day.

- The guy who implemented it is
  Promoted up the line;
  And some of the
  enhancements
  Didn't match to the design.

- They haven't kept the flowcharts,
  The manual's a mess,
  And most of what you need to
  know
  You'll simply have to guess.

- We do not know the reason,
  Why the bugs pour in like rain,
  But don't just stand here gaping,
  Get out there and MAINTAIN.