University of Zurich[UZH]

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

Software Wartung und Evolution

Prof. Dr. Harald Gall

Dr. Emanuel Giger

Universität Zürich, Institut für Informatik

http://www.ifi.uzh.ch/seal/teaching/courses/SWEvo13.html

http://tinyurl.com/seal-evolution

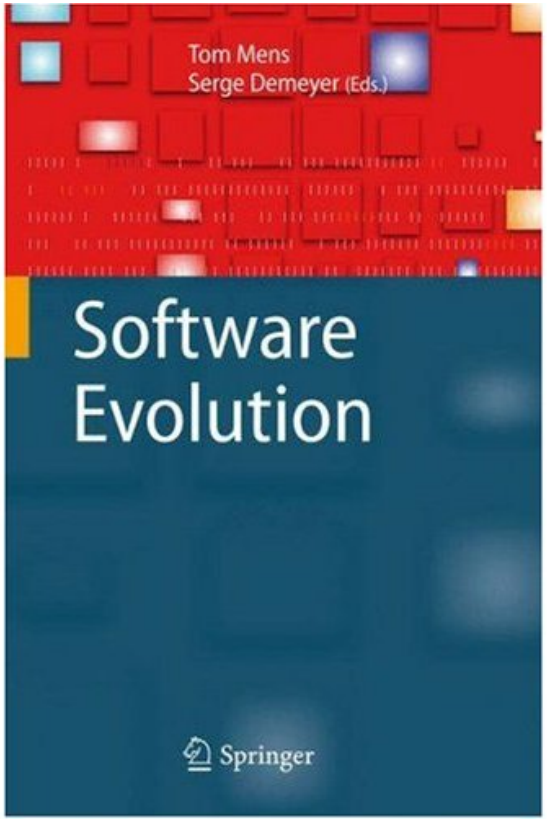
Organisatorisches /1

- LV Info
  - Vorlesung mit Übungen
  - Freitag, 13:00 - 14:45 Uhr, BIN 2.A.10
- Unterlagen
  - Deutsch & Englisch (Folien, wiss. Artikel)
- Voraussetzungen
  - BSc Informatik: Assessmentstufe, Module Software Engineering und Software-Praktikum
- Anrechenbarkeit
  - BSc Informatik (ab 4. Semester)
  - MSc Informatik

Organisatorisches /2

- Leistungsnachweis
  - Erfolgreiche Bearbeitung von Übungsaufgaben während des Semesters und Teilnahme an der schriftlichen Endklausur
- Schriftliche Prüfung am 7. Juni 2013
  - Bei geringer Teilnehmerzahl mündliche Prüfungen (Termine nach Vereinbarung)
- LV-Web-Seite
  - http://tinyurl.com/seal-evolution

Buchempfehlung



- Die Vorlesung folgt mitunter dem Buch und weiteren wissenschaftlichen Papieren gemäss Webseite der LV

Themen der Vorlesung

- Software Wartung: Definition, Arten, Aspekte, Aktivitäten, Wartungskrise, Legacy Systeme
- Reverse Engineering, Re-Engineering, Refactoring, Architecture Reflexion
- Defect Tracking, Software Configuration Management, Produktivstellung, Dokumentation,
- *Software Evolution:* Laws, Software Evolution Analysis
- Change Patterns, Maintainability, Design for Change (e.g. OO, AOP, SOC)
- Best Practices & Tools für Wartung und Testen
- Program Comprehension, Impact Analysis, Change Propagation
- Empirical Studies, Defect Prediction

Teil 1

Motivation: Software Wartung und Evolution

Abgrenzung Software Wartung zu Entwicklung und Evolution

Definition Software Wartung

Probleme und Herausforderungen der Software Wartung

What is Software Maintenance?

- According to the IEEE Standard for Software Maintenance, IEEE 1219, software maintenance is defined as
  "the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment."
- The standard also addresses maintenance activities prior to delivery of the software product.

What is Software Evolution?

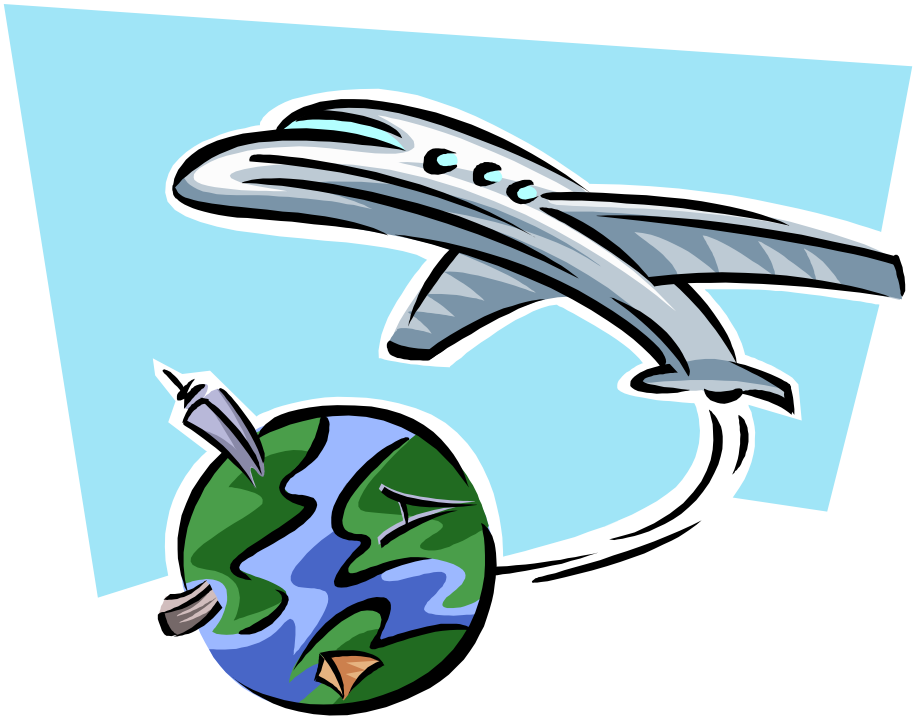
- There is no standard definition, but the Wikipedia wording for it is:
  "Software evolution is the term used in software engineering (specifically software maintenance) to refer to the process of developing software initially, then repeatedly updating it for various reasons."

Software ist verschleissfrei

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

Änderbarkeit von Software

- 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));
}
```

„Programs, like people, get old“

- „Software aging will occur in all successful products“
  - David L. Parnas, “Software Aging,“ in Proceedings of ICSE 1994

- Es ist einsichtig, dass, was altert, irgendwie up-to-date gehalten werden muss.
- Wie altert Software und warum?
- Was sind effektive Methoden und Werkzeuge, um Alterung hintanzuhalten oder gar zu vermeiden?

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

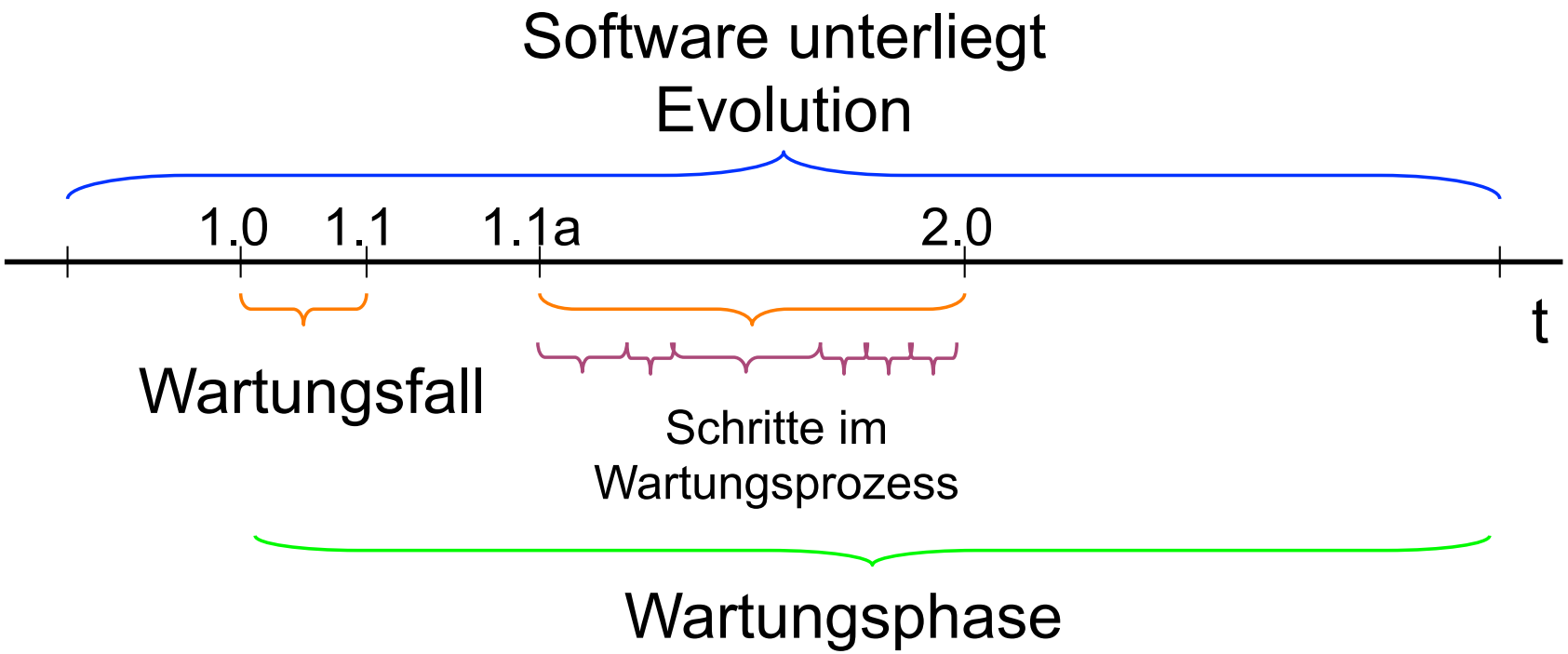
[D. L. Parnas, 1994]

Software 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 Software-Systems von der Erstellung bis zur Stilllegung
  - umfasst: Entwicklung, Wartung, Migration, Stilllegung

Software Wartung vs. Software Evolution



Warum Evolution?

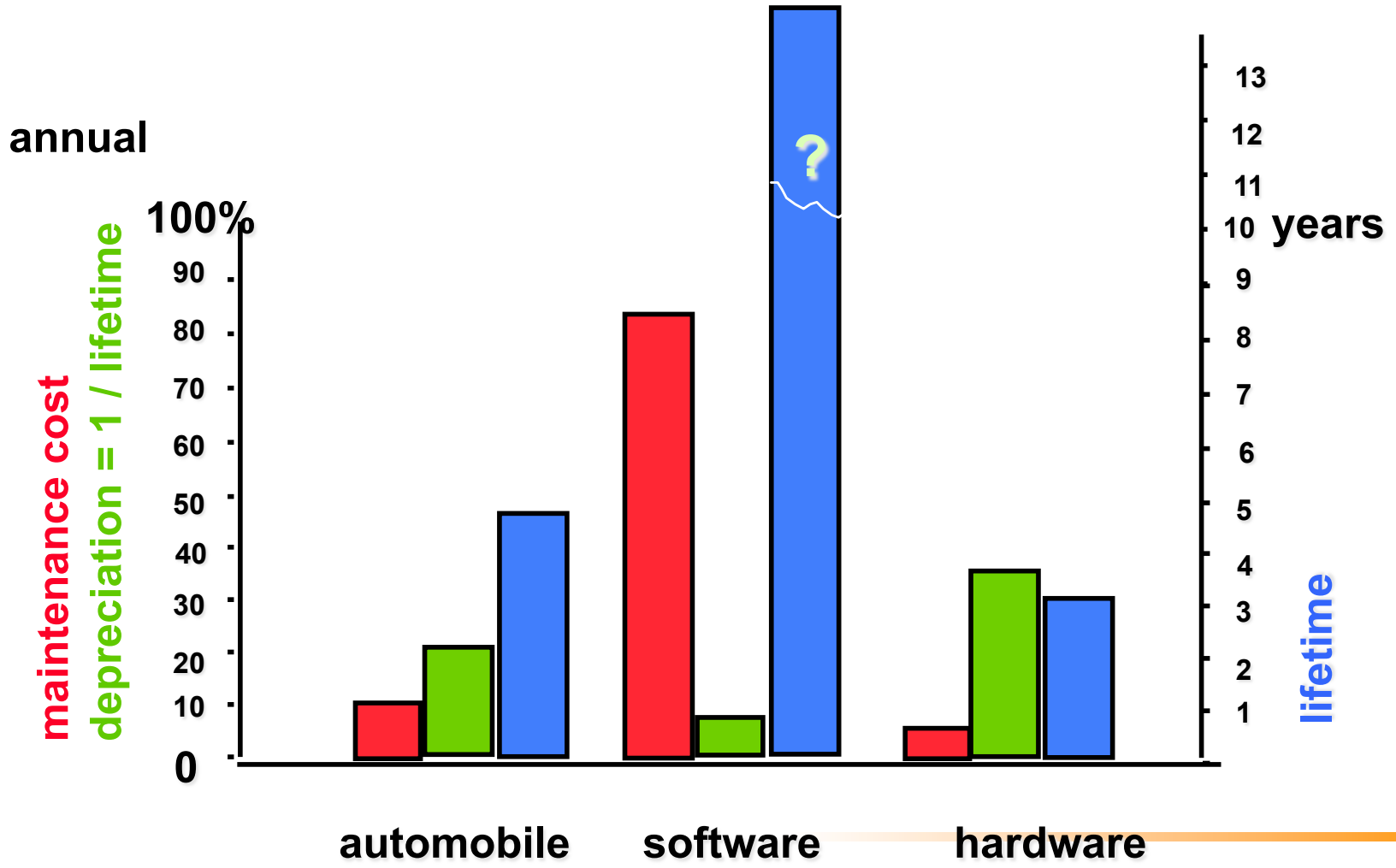
- Viele Software Systeme bilden Geschäftsprozesse der realen Welt nach
- Geschäftsprozesse unterliegen ständigen Änderungen
  - passiv durch Adaption an neue Gegebenheiten (neue rechtliche Gegebenheiten, Marktsituation, Euro, Basel II, …)
  - aktiv durch die Einführung neuer Produkte, neuer Prozesse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

Warum Analyse von Software Evolution?

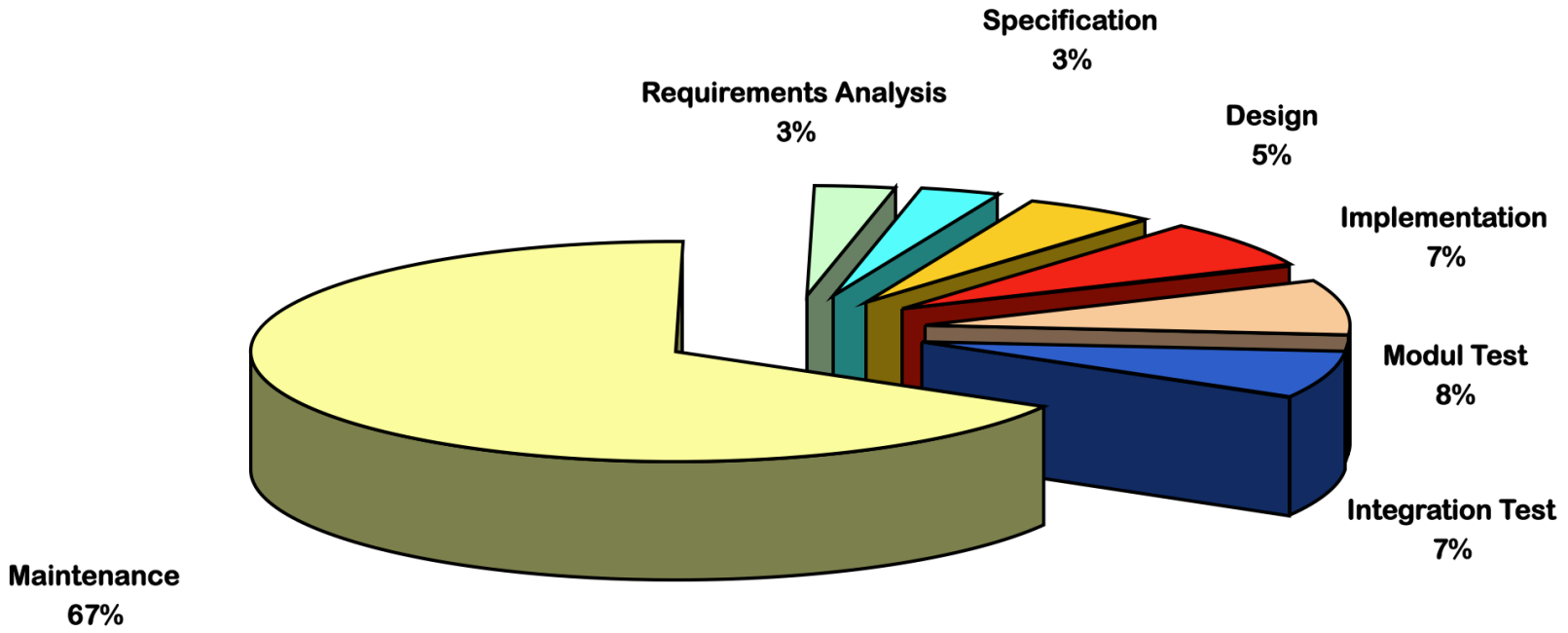
- „Nevertheless, the industrial track record raises the question, why, despite so many advances, [...]
  - satisfactory functionality, performance and quality is only achieved over a lengthy evolutionary process,
  - software maintenance never ceases until a system is scrapped
  - software is still generally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

[Lehman et al., 1997]

Software Wartung im Vergleich



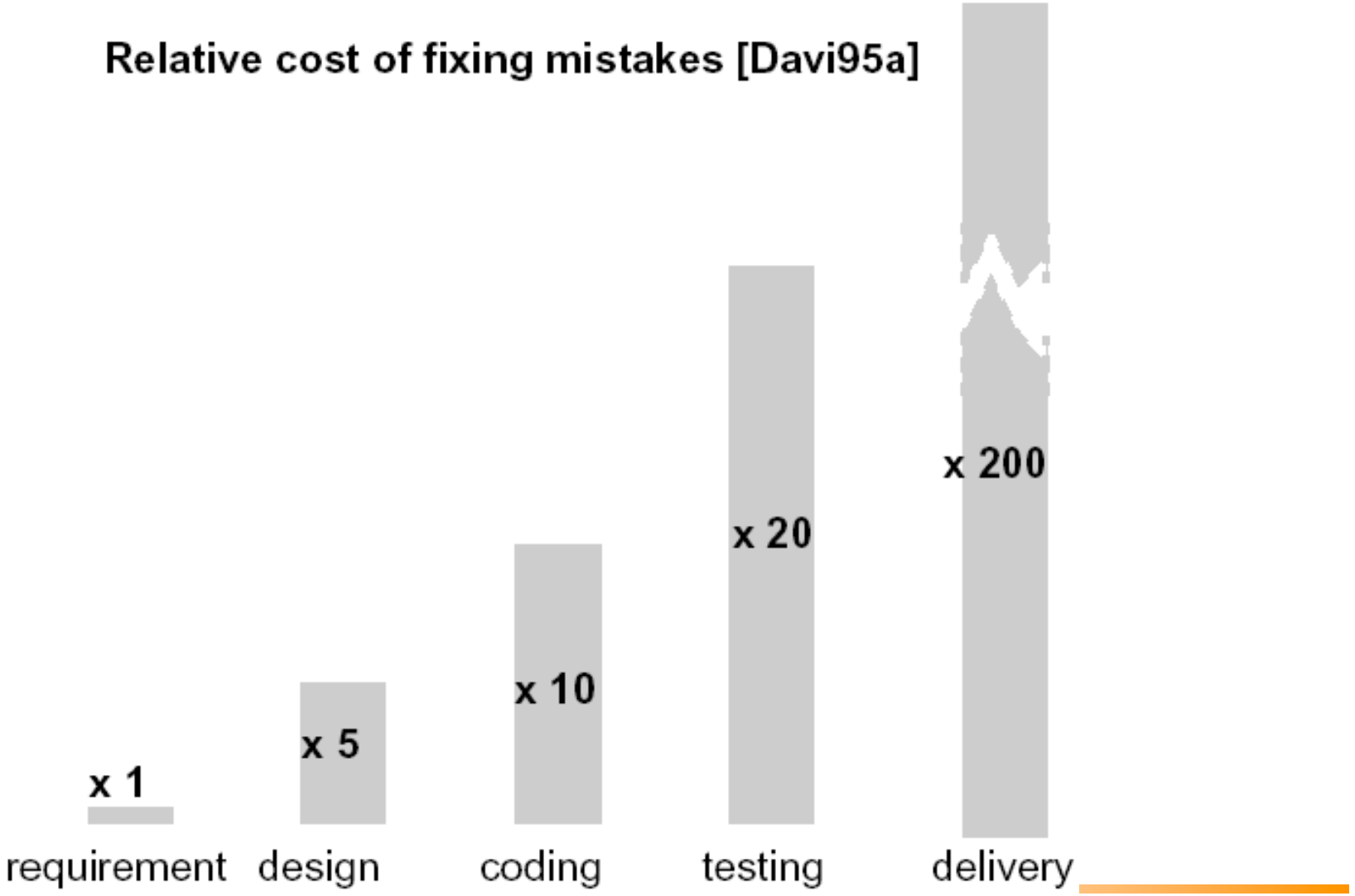
Cost Distribution in Software Life-Cycle



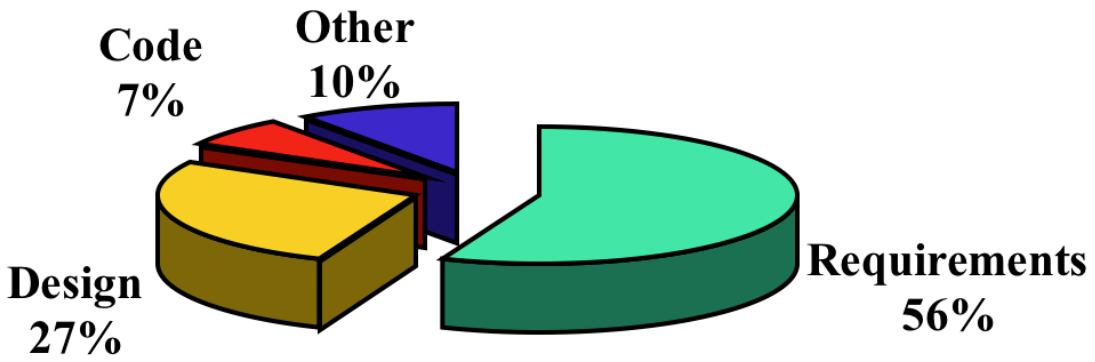
Cost Distribution in the Software-Life-Cycle

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

Cost of fixing bugs per phase

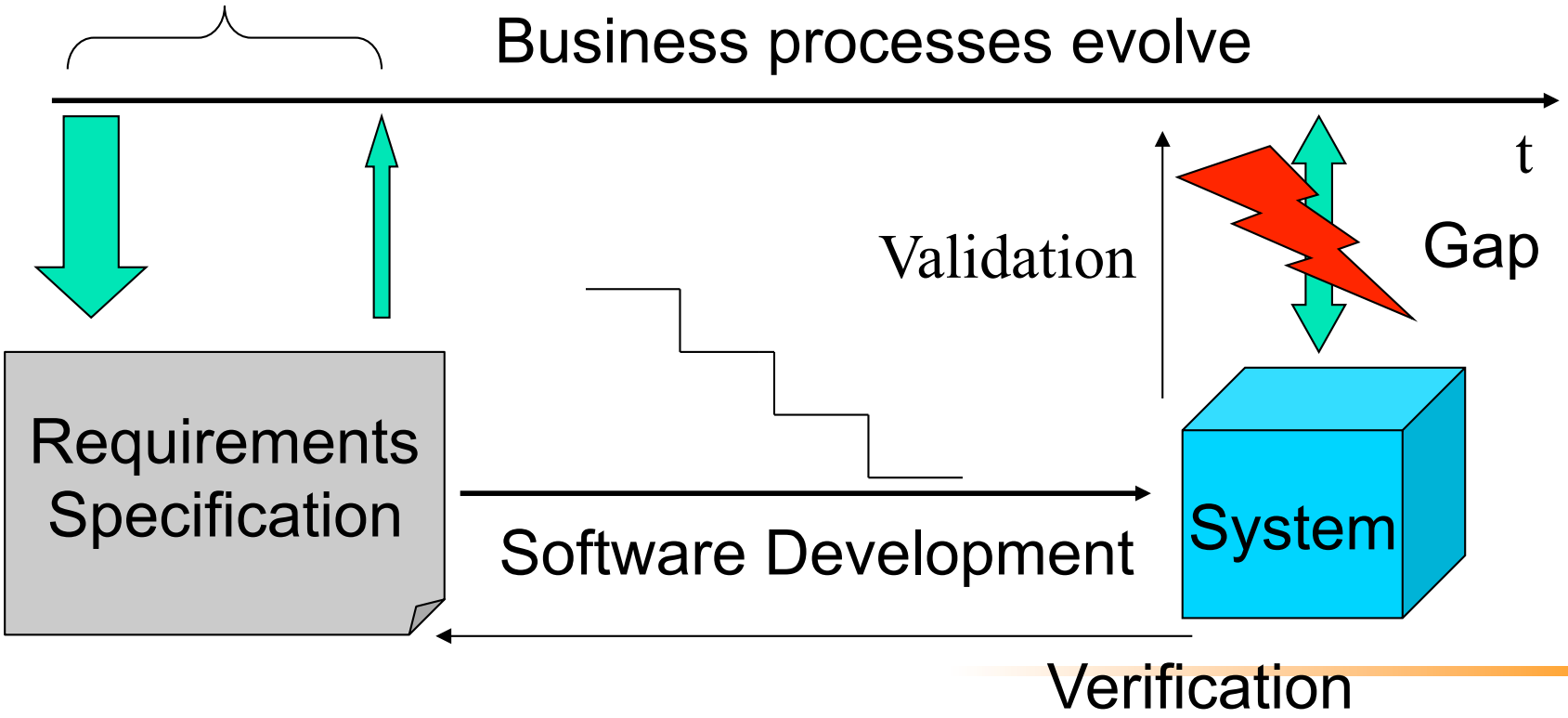


Distribution of Bugs



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

University of Zurich[UZH]

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

Software Wartung

Definition: Software Wartung

- Nach IEEE IEEE 1219-98
  „... the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.“
- IEEE/EIA 12207 standard for software life cycle processes depicts maintenance as one of the primary life cycle processes,
  „... as the process of a software product undergoing modification to code and associated documentation due to a problem or the need for improvement. The objective is to modify the existing software product while preserving its integrity.“

Arten der Software Wartung

- (1) Korrektive Wartung (21%)
  - „Bug fixing“; reaktive Natur
- (2) Präventive Wartung (4%)
  - Finden von latenten Fehlern, bevor sie effektive Fehler werden
- (3) Adaptive Wartung (25%)
  - Neue Hardware, Betriebssystem; neue Anforderungen
- (4) Perfektionierende Wartung (50%)
  - Verbesserungen in Performance und Wartbarkeit (Restructuring, Reverse Engineering, Dokumentationspflege, etc.)
- Corrections = (1) + (2) ~ 25%
- Enhancements = (3) + (4) ~75%

Warum ist Software Wartung nicht trivial?

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 Wartung

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

Schwierigkeiten in der 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 Wartung

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

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

Legacy Systeme

- 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

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 in der Software Entwicklung

- 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.