

**XION IT SYSTEMS**

AKTIENGESELLSCHAFT

Dresdnerstraße 81-85/8.Stock  
A-1200 Wien

Tel: 0664-8242-600

E-mail: office@xion.at

Web: xion.at

Festnetz: +43/1/333 91 99-0

Fax: +43/1/333 91 99-199

x i o n . i t s y s t e m s . o g



# Software Wartung und Evolution

*Dipl.-Ing. Dr. techn. Johannes Weidl-Rektenwald*  
*Xion IT Systems AG*

**XION IT SYSTEMS**

AKTIENGESELLSCHAFT

Dresdnerstraße 81-85/8.Stock  
A-1200 Wien

Tel: 0664-8242-600

E-mail: office@xion.at

Web: xion.at

Festnetz: +43/1/333 91 99-0

Fax: +43/1/333 91 99-199

x i o n . i t s y s t e m s . o g



# Motivation und Inhalt

## Maintenance

**Wartung der Anwendungen verbraucht 83% des IT Budget**

*"Too much money for maintenance. With just 27% of 2004 spending planned for new development, maintenance costs are choking IT productivity."*

Gartner, Executive Summary, September 2004

**60% des IT-Budget geht auf für Legacy Wartung**

*"Between 60 and 80 percent of an average company's IT budget is spent on maintaining existing mainframe systems and applications."*

Gartner Group, Analyst Report, March 2002

## Legacy

**Legacy gibt es (fast) immer und überall**

- 70% der weltweiten Geschäftsdaten werden mit COBOL prozessiert
- Es gibt über 200 Milliarden COBOL "Lines of Code"
- 30 Milliarden COBOL Transaktionen werden täglich prozessiert – mehr als alle Zugriffe auf Webseiten

**Legacy Modernisierung ist eine strategische Notwendigkeit**

*"It is clear that new technology will no more replace legacy in total than legacy technology alone will satisfy the needs of today's Web-savvy users. The truth is that modernized legacy applications play a crucial role."*

Giga, Analyst Report, September 2002

## 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 managt man Software Wartung?

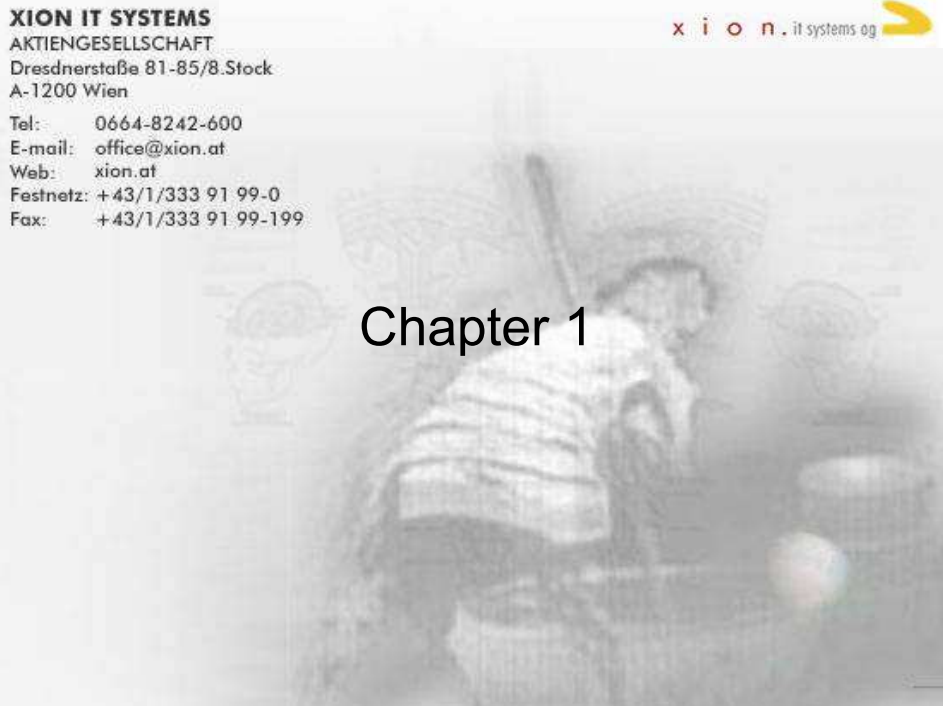
## Inhalt der Vorlesung: Themen

- (1) *Software Wartung*: Motivation, Definition, Arten, Probleme
- (2) *Software Wartung*: Aspekte, Aktivitäten, Wartungskrise, Legacy Systeme, *Reverse Engineering*
- (3) *Restructuring*, *Re-Engineering*, *Organisation der Wartung*: Software Life Cycle Modelle
- (4) *Tool Demos*, *Organisation der Wartung*: Defect Tracking, Software Configuration Management, Produktivstellung
- (5) *Software Evolution*: E-type Programs, Laws of Software Evolution Explained, Change Patterns
- (6) *Spezielle Kapitel der Software Wartung*: Program Comprehension, Change Impact Analysis, Qualitätsmerkmal „Wartbarkeit“
- (7) *Best Practices*: Design for Change, MDA und Wartung, *Software Wartung im unternehmerischen Kontext*: Rollen, Gewährleistung, Software-Wartungsverträge

**XION IT SYSTEMS**  
AKTIENGESELLSCHAFT  
Dresdnerstraße 81-85/8.Stock  
A-1200 Wien

Tel: 0664-8242-600  
E-mail: office@xion.at  
Web: xion.at  
Festnetz: +43/1/333 91 99-0  
Fax: +43/1/333 91 99-199

x i o n . it systems og 



# Chapter 1

## Chapter 1

- Inhalte
  - Motivation: Software Wartung und Evolution
  - Abgrenzung der Begriffe Software Wartung und Software Entwicklung
  - Definition „Software Wartung“
  - Arten der Software Wartung
  - Probleme der Software Wartung

## Software Engineering vs. Software Maintenance

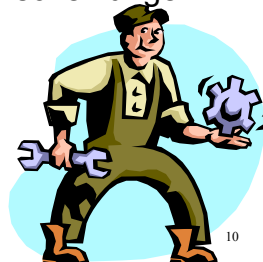
- Suche nach dem Begriff „Software Engineering“ bei amazon.de, Kategorie „Englische Bücher“
  - Treffer: 9218 (2007: 8405; 2006: 1510; 2005: 1405; 2004: 1078)
    - Top Treffer: „Software Engineering: Update (Ian Sommerville et al.; 2006)“
    - Treffer 2: „A Handbook of Software and Systems Engineering.“
    - Treffer 3: „Practical Software Engineering. Analysis and Design for the .NET Platform.“
- Suche nach „Software Maintenance“
  - Treffer: 403 (2007: 374; 2006: 90; 2005: 92; 2004: 92)
    - Top Treffer: „Effective Software Maintenance and Evolution: A Reuse-Based Approach“ (Stanislaw Jarzabek; 2007)
    - Top Treffer 2007: „Macs for Dummies“ (Baig; 2006)
    - Top Treffer 2006: „Troubleshooting your PC“ (1994)
- Search for „Software Evolution“
  - Treffer: 91 (2007: 81; 2006: 23; 2005: 20; 2004: 15)
    - Top Treffer: „Software Evolution and Feedback. Theory and Practice.: Theory and Practice“ (Nazim H. Madhavji, M. M. Lehmann, und J. Ramil; 2006)
    - Top Treffer 2007: „The Old New Thing. Practical Development Throughout the Evolution of Windows“ (Chen et al.; 2007)
    - Treffer #7/2007: „Successful Evolution of Software Systems“ (Yang et al.; 2003)
    - Top Treffer 2006: „Software Engineering: Evolution And Emerging Technologies: 130“ von K. Zielinski (26. Dez. 2005!)

© J. Weidl-Rektenwald since 2002

9

## 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 Abnutzungserscheinungen („wear-out“)

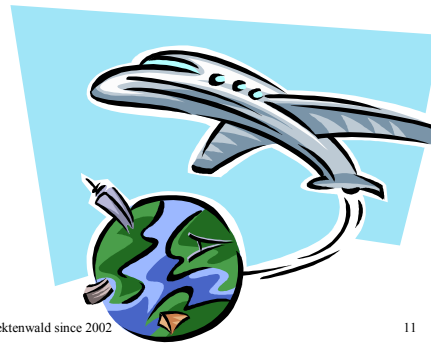


© J. Weidl-Rektenwald since 2002

10

## Annäherung an den Begriff „Software Wartung“

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



© J. Weidl-Rektenwald since 2002

11

## Software ist leicht änderbar?

```
/**
 * Wrong name: should be getRechnungToAuftragIDAndKeyAccountID
 */
public static Rechnung getRechnungToAuftragIDAndPersonID (Integer iAID,
    Integer iKAID) throws ExceptionPersist {

    Vector v;
    Rechnung r = new Rechnung();
    r.setAuftragID(iAID);
    r.setKundenID(iKAID);
    v = r.search();

    [...]
```

© J. Weidl-Rektenwald since 2002

12

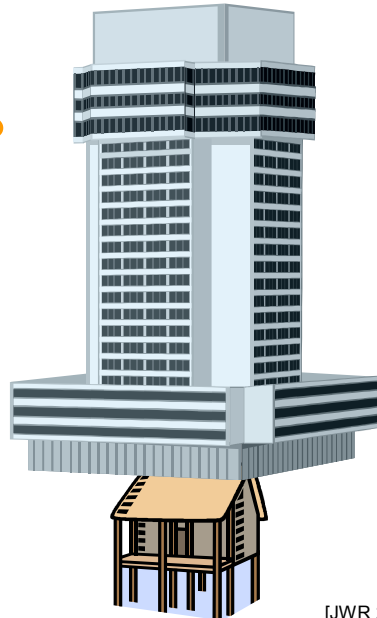
## Software ist leicht änderbar?

```
for(int j=0;j<vtz.size();j++) {  
  
    tzkbez = ((DtZeile)vtz.elementAt(j)).getDtKurzBezeichnung();  
    iZeichPos = tzkbez.indexOf(':');  
  
    if (tzkbez.substring(0,iZeichPos).equals(tzkbez.substring(iZeichPos+1)))  
        ivSortDt = 1;  
    else if (tzkbez.substring(0,iZeichPos).compareTo(tzkbez.substring(iZeichPos+1))>0)  
        ivSortDt = 0;  
    else  
        ivSortDt = 2;  
  
    if (ivSortDt == 0) iZeichPos = 0;  
    else iZeichPos++;  
  
    int i;  
    for(i=0;i<vSortDt[ivSortDt].size();i++) {  
        if  
            (((DtZeile)vSortDt[ivSortDt].elementAt(i)).getDtKurzBezeichnung().substring(iZeichPos).compareTo(tzkbez.substring(iZeichPos))>0)  
            break;  
        }  
  
    if (i<vSortDt[ivSortDt].size())  
        vSortDt[ivSortDt].insertElementAt(vtz.elementAt(j),i);  
    else  
        vSortDt[ivSortDt].add(vtz.elementAt(j));  
}  
}
```

© J. Weidl-Rektenwald since 2002

13

## Software ist leicht änderbar?



[JWR 2002]

© J. Weidl-Rektenwald since 2002

14

## 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?

© J. Weidl-Rektenwald since 2002

15

## 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]

© J. Weidl-Rektenwald since 2002

16

## 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]

© J. Weidl-Rektenwald since 2002

17

## 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]

© J. Weidl-Rektenwald since 2002

18

## „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]

© J. Weidl-Rektenwald since 2002

19

## Software Wartung vs. Software Evolution

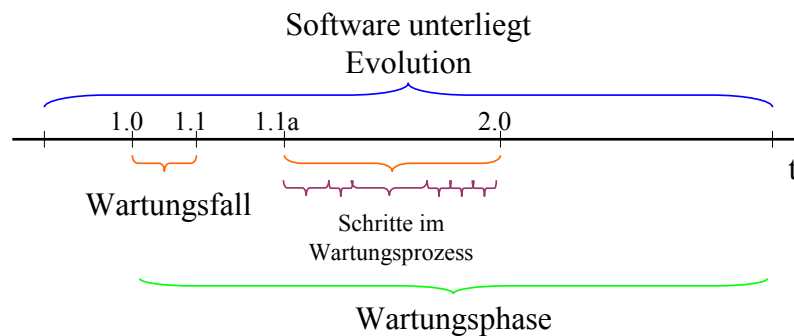
- Software Wartung
  - bezeichnet als **Tätigkeit** eine Änderung an einem Softwaresystem nach dessen Auslieferung (die Durchführung einer Änderung bezeichnet man als „Wartungsfall“)
  - bezeichnet als **Prozess** die Schritte, die in einem Wartungsfall sequentiell durchzuführen sind
  - bezeichnet als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

[JWR 2002]

© J. Weidl-Rektenwald since 2002

20

## Software Wartung vs. Software Evolution



© J. Weidl-Rektenwald since 2002

21

## 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

© J. Weidl-Rektenwald since 2002

22

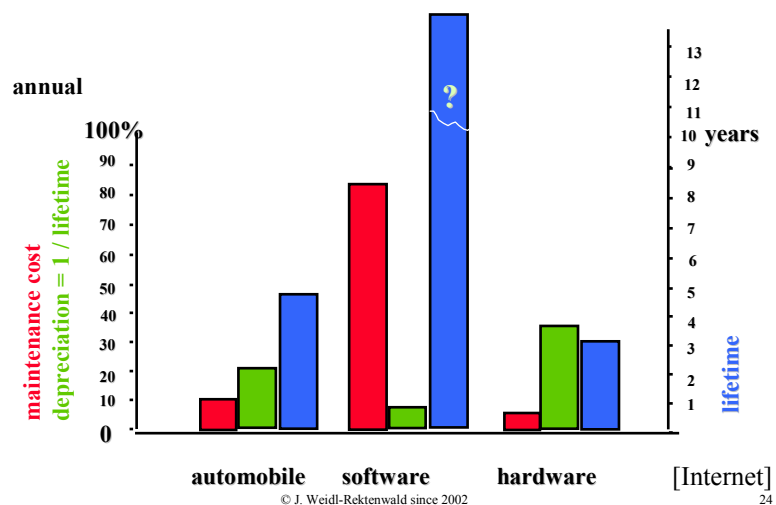
## Warum Beschäftigung mit Software Wartung/Evolution?

- „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]

© J. Weidl-Rektenwald since 2002

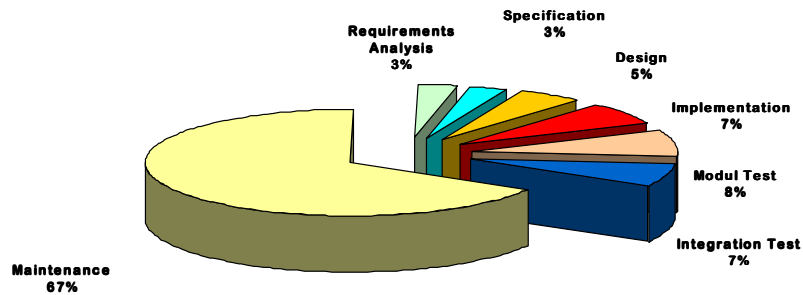
23

## Software Wartung im Vergleich



24

## Cost Distribution in the Software Life-Cycle

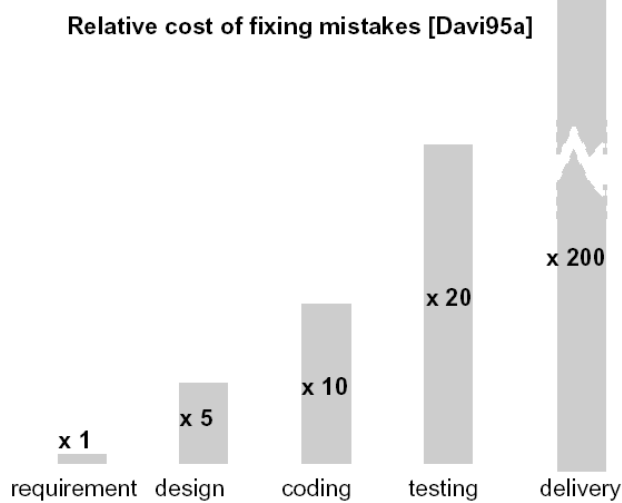


### Cost Distribution in the Software-Life-Cycle

Source: Principles of Software Engineering and Design, Zelkovits, Shaw, Gannon 1979  
© J. Weidl-Rektenwald since 2002

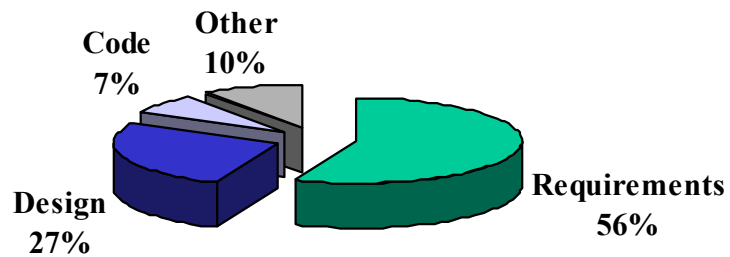
25

## Cost of fixing bugs per phase



26

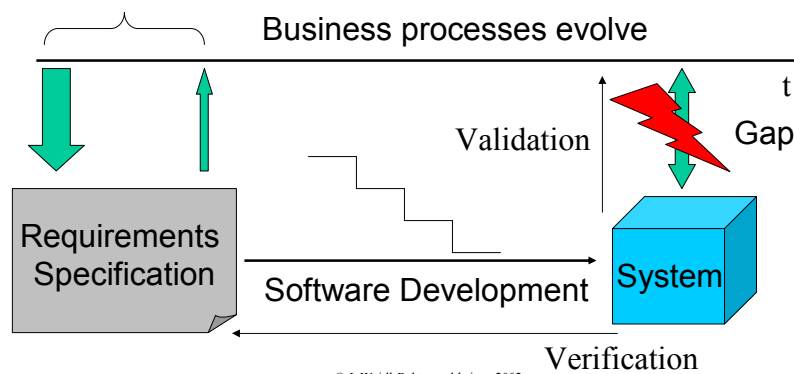
## Distribution of Bugs



© J. Weidl-Rektenwald since 2002 Source: Klösch/Gall, „Objektorientiertes Reverse Engineering“, Springer 27

## Classic Approach to Software Development

„Requirements Fixing“



© J. Weidl-Rektenwald since 2002 28

## Difference software maintenance vs. software 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

[Internet]

© J. Weidl-Rektenwald since 2002

29

## 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

© J. Weidl-Rektenwald since 2002

31

## 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.

© J. Weidl-Rektenwald since 2002

32

## Allgemeinere Definition [SWEBOK]

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

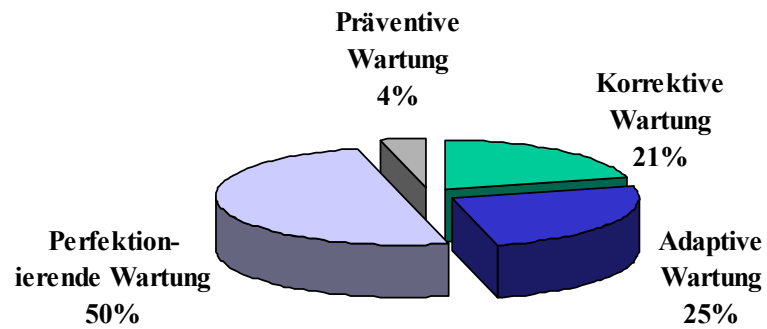
© J. Weidl-Rektenwald since 2002 *Source: Software Engineering Body of Knowledge, <http://www.swebok.org/>* 33

## Arten der Software Wartung

- ❶ Korrektive Wartung
  - „Bug fixing“; reaktive Natur
- ❷ Präventive Wartung
  - Finden von latenten Fehlern, bevor sie effektive Fehler werden
- ❸ Adaptive Wartung
  - Neue Hardware, Betriebssysteme etc.; neue Anforderungen
- ❹ Perfektionierende Wartung
  - Verbesserungen von Performance und Wartbarkeit (Restructuring, Reverse Engineering, Dokumentationspflege, etc.)
- ❶ + ❷: Corrections / ❸ + ❹: Enhancements

© J. Weidl-Rektenwald since 2002 *Source: Klösch/Gall, „Objektorientiertes Reverse Engineering“, Springer* 34

## Arten der Software Wartung



© J. Weidl-Rektenwald since 2002  
Source: Klösch/Gall, „Objektorientiertes Reverse Engineering“, Springer 35

## 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 als für das Programm notwendig bzw. überhaupt sinnvoll

© J. Weidl-Rektenwald since 2002

37

## 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

© J. Weidl-Rektenwald since 2002

38

## 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 Faktorensomme
- Die **Dynamik** des Programmdurchlaufs ist während der statischen Code Inspektion nicht oder nur schwer ableitbar

© J. Weidl-Rektenwald since 2002

39

## Schwierigkeiten in der Software Wartung

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

© J. Weidl-Rektenwald since 2002

40

## 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 meist unter **Zeitdruck** und das Management und die Dokumentation des Wartungsfalls werden vernachlässigt

© J. Weidl-Rektenwald since 2002

41

## Schwierigkeiten in der Software Wartung

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

© J. Weidl-Rektenwald since 2002

42

## 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, FORTRAN, 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
  - Problem der Datenmigration!

© J. Weidl-Rektenwald since 2002

45

## Geschichte der Software Wartung

- Die Wartung von Programmen galt von jeher als *unbeliebte* Tätigkeit im Software Life-Cycle
  - Historisch die Arbeit von neu rekrutierten bzw. nicht so erfahrenen Programmierern
- 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**

© J. Weidl-Rektenwald since 2002

46

## Evolutionäre Probleme der Software 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**“)

© J. Weidl-Rektenwald since 2002

47

## 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)

© J. Weidl-Rektenwald since 2002

48

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