Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

Bachelor of Science (BSc) in Informatik

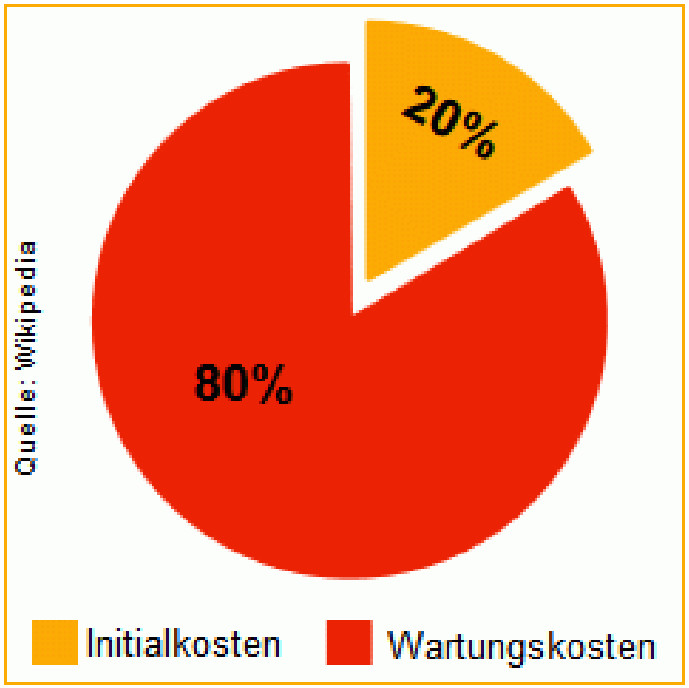
Modul Advanced Software Engineering 2 (ASE2)

LE 12 – Software Maintenance and Operations

Softwarewartung

Institut für Angewandte Informationstechnologie (InIT)
Walter Eich (eicw) / Matthias Bachmann (bacn)
https://www.zhaw.ch/de/engineering/institute-zentren/init/

Um was geht es?



Aufwand für Softwarewartung ist abhängig von:

- der betreffenden Software (Fehlerdichte, Wartbarkeit),
- ihrem Einsatz (Fehleroffenbarung durch unterschiedliche Anwendungsszenarien;
- dem Wunsch, bestimmte Attribute zu verbessern) und der Einsatzdauer (änderndes Umfeld).

Agenda

1. Begriffswelt der Softwarewartung
2. Evolution von Software
3. Management der Softwarewartung
4. Techniken der Softwarewartung
5. Wrap-up

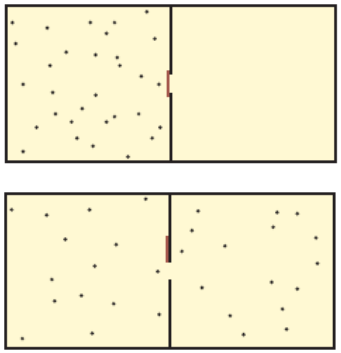
Lernziele

Sie sind in der Lage,

- die Gründe für Softwarewartung und deren Bedeutung zu diskutieren.
- die wichtigsten Begriffe wie Software, Softwarewartung, Wartbarkeit und Softwareevolution zu erklären.
- die wichtigsten Kategorien bzw. Aufgaben der Softwarewartung zu beschreiben.
- die Gesetze der Softwareevolution nach Lehman und deren Konsequenzen zu erklären.
- den Wartungsprozess und dessen Einbettung im Softwarelebenszyklus darzustellen.
- wichtige Techniken der Softwarewartung wie Softwareanalyse und -visualisierung, Reengineering, Sanierung und Migration in der Softwarewartung zu benennen.

Warum eigentlich ist Softwarewartung nötig?

- Als allgemeine Erklärung gilt die Entropie aus der Physik.
- Die Entropie ist das Mass der Unordnung oder Zufälligkeit eines Systems und findet in der Physik im 2. Hauptsatz der Thermodynamik Verwendung.
- Überlässt man ein Software-System sich selbst, so nimmt ohne jegliches Zutun die Unordnung bzw. Software Entropie zu.
- Die Software Entropie wird erhöht durch die Akkumulation von technischen Schulden («technical debts»).
- Andrew Hunt and David Thomas benutzen «fixing broken windows» als eine Metapher zur Vermeidung der Software Entropie in der Softwareentwicklung.
- Ein konsequentes Code Refactoring kann in einer schrittweisen Reduktion der Software Entropie resultieren!

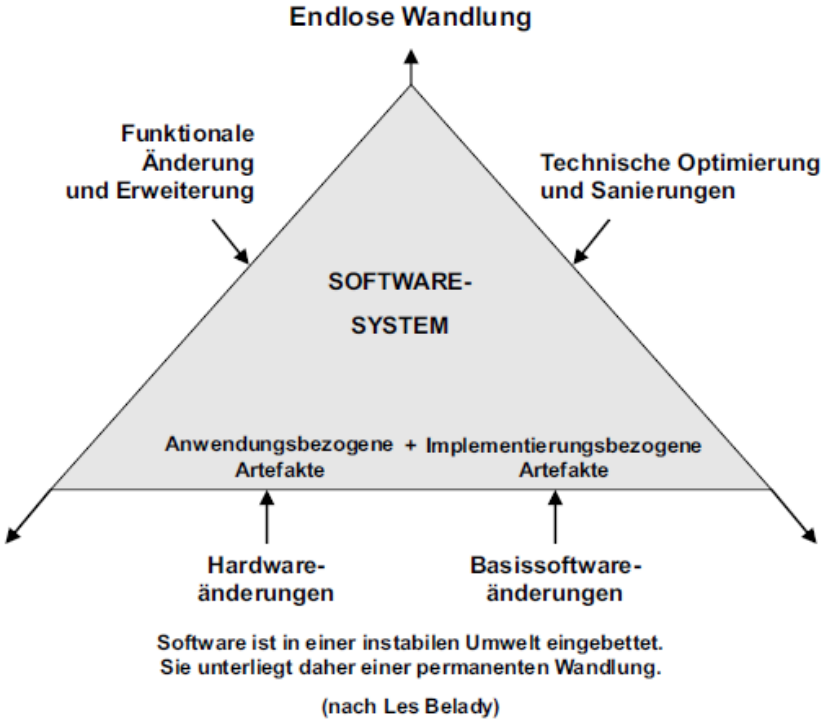


Warum eigentlich ist Softwarewartung nötig?

- Der Begriff Software Entropie ist zwar korrekt, aber etwas allgemein.
- David L. Parnas (1994) nennt zwei konkrete Gründe für die Alterung von Software-Systemen, die Wartung nach sich ziehen:
  - lack of movement, also das Nicht-Anpassen der Software an die sich verändernde Umgebung,
  - ignorant surgery, das Anpassen von Code, ohne die Kenntnisse der zugrunde liegenden Struktur. Dies führt zum inneren Zerfall der Software.

Software lebt in einer instabilen Umwelt

- Unterlässt man diese Änderungen, veraltet das System ohne eigenes zutun gegenüber der sich veränderten Umgebung.
- Es entsteht eine Lücke gegenüber dem Wandel (lack of movement).

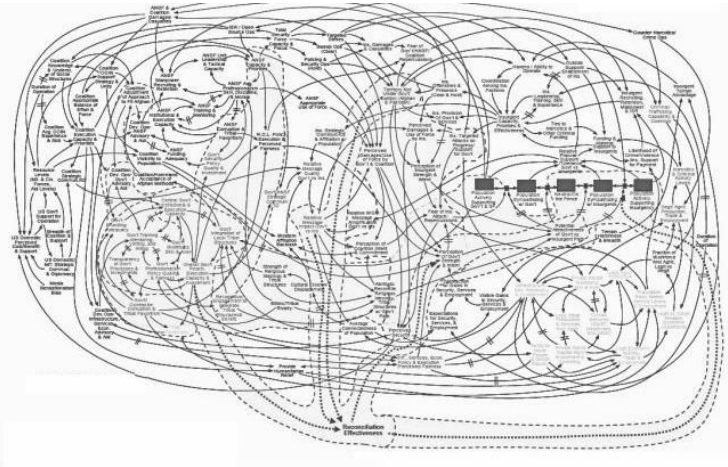


Software ist in einer instabilen Umwelt eingebettet.
Sie unterliegt daher einer permanenten Wandlung.

(nach Les Belady)

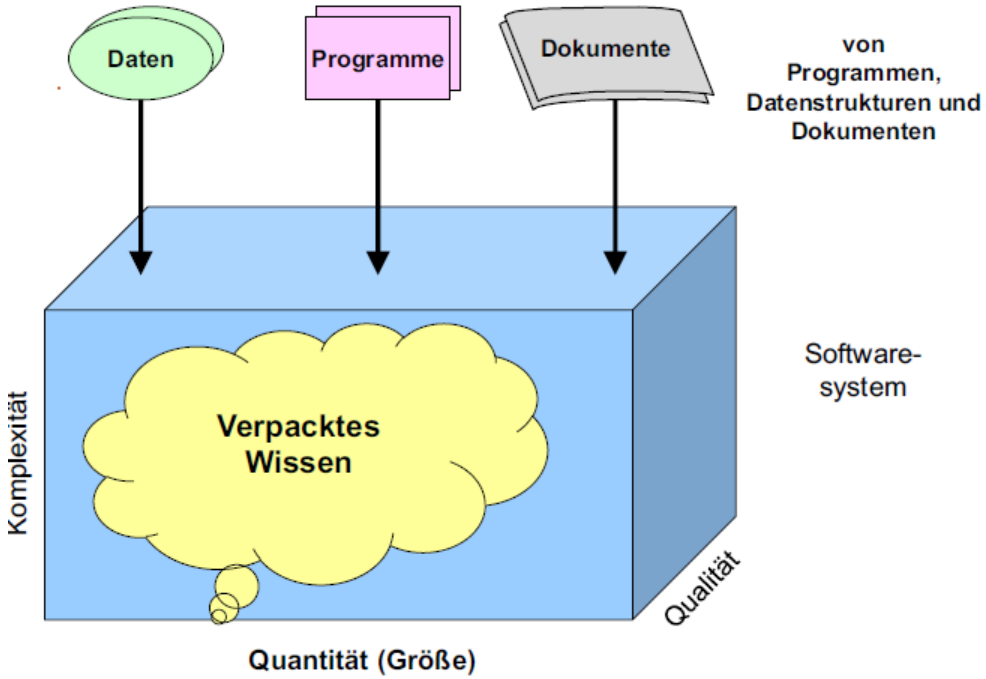
Anpassen der Software ohne Kenntnisse der zugrundeliegenden Struktur

- Wenn Änderungen von Leuten vorgenommen werden, die die zugrundeliegenden Architektur- und Design-Prinzipien eines System nicht kennen, führt dies zu einer Verwässrung der Konzepte und einer Erhöhung der internen Komplexität (ingnorant surgery).
- Stellen Sie sich einmal vor, ein Chirurg würde ohne die Grundkenntnisse der Anatomie, also der Architektur- und Designgrundlagen des Menschen, an Ihnen rumschneiden!



Software

- **Software** sind Programme, Verfahren, zugehörige Dokumentation und Daten, die mit dem Betrieb eines Computersystems zu tun haben. [IEEE Std. 610.12-1990]



Software ist ein Kapitalgut mit drei Dimensionen = Quantität x Komplexität x Qualität.

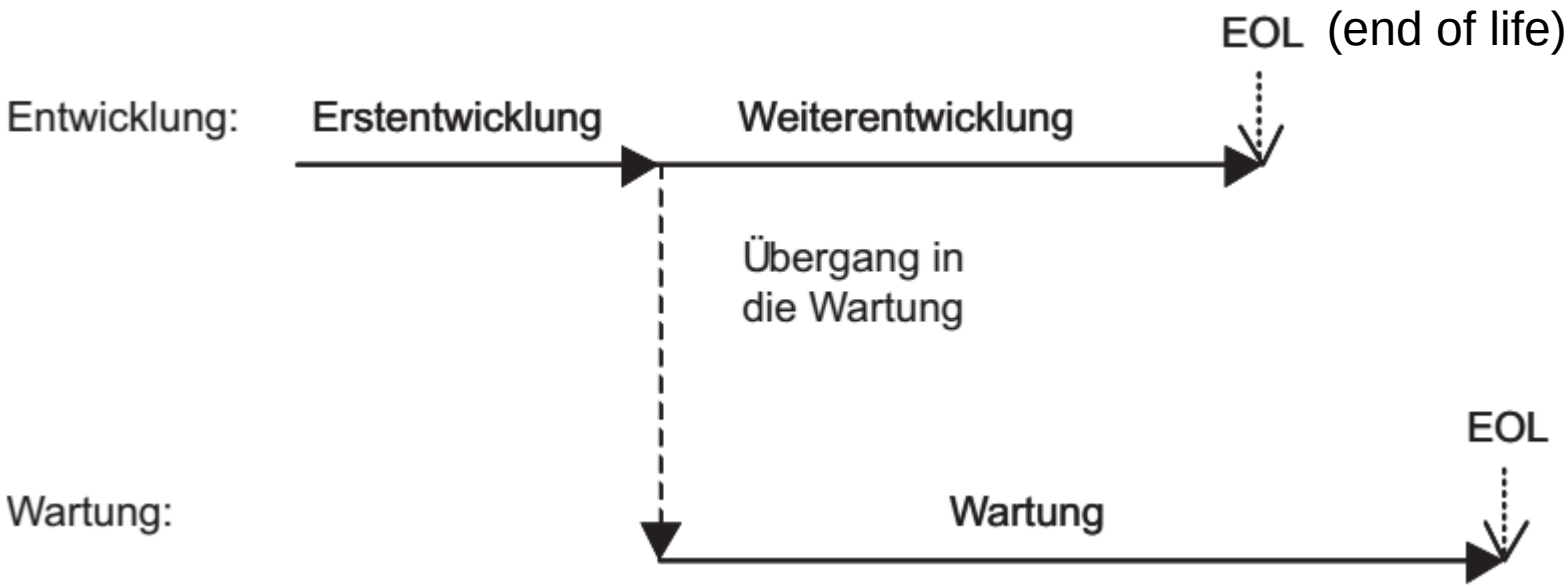
Software ist immateriell:

- nichts daran ist natürlich
- wird nur entwickelt – keine Fertigung
- Kopie und Original sind gleich
- verschleisst nicht
- Fehler entstehen nicht durch Abnutzung
- realisiert keine stetige Funktion
- Wiederverwendung extrem lukrativ

Softwarewartung

- **Softwarewartung** ist die Veränderung eines Softwareprodukts nach dessen Auslieferung, um Fehler zu beheben, Performanz oder andere Attribute zu verbessern oder Anpassungen an die veränderte Umgebung vorzunehmen. [IEEE Std. 610.12-1990]

Der Software-Lebenszyklus – Grundlage der Wartung



- Softwarewartung macht nur Sinn, wenn man den gesamten Lebenszyklus der Software betrachtet – von der Idee für die Software bis zum Ende der Anwenderunterstützung.

Wartbarkeit

- **Wartbarkeit** ist die Einfachheit, mit der ein Softwaresystem oder eine Komponente modifiziert werden kann, um Fehler zu beheben, Performanz oder andere Attribute zu verbessern oder Anpassungen an die veränderte Umgebung vorzunehmen**.**
  [IEEE Std. 610.12-1990]

Legacy Systeme (Altsysteme)

- Viele Probleme der Softwarewartung treten erst im Zusammenhang mit grossen, alten, komplexen Software-Systemen auf, sogenannten «Legacy Systemen».
- Diese wurden mit Methoden konzipiert und in Sprachen erstellt, die heute kaum mehr benutzt (und noch weniger gelehrt) werden.
  - Strukturierte Analyse und Design, proprietäre Analyse- und Designansätze
  - CICS, COBOL, PL/I

- *Aber Achtung: Was sagen unsere Nachkommen in 30 Jahren über Java- oder C#-Programme?*

Softwareevolution

- Als **Softwareevolution** bezeichnen wir den Prozess der Veränderung eines Softwaresystems von der Erstellung bis zur Ausserbetriebnahme. Dieser Prozess umfasst somit die Erstellung, Wartung, Weiterentwicklung, Migration und Stilllegung**.** (nach Lehmann und Belady, 1985)

Kategorien der Wartung

- Folgende Kategorien von Wartung sind nach IEEE Std. 610.12-1990 definiert:
  - **Korrektive Wartung**
  - **Adaptive Wartung**
  - **Perfektionierende Wartung**
  - **Präventive Wartung**

- Daraus lassen sich die Aufgaben der Softwarewartung ableiten:

	Korrektur	**Verbesserung**
Reaktiv	Korrektive Wartung (Restfehler)	Adaptive Wartung (lack of movement)
Proaktiv	Präventive Wartung (Restfehler)	Perfektionierende Wartung (ignorant surgery)

Korrektive Wartung

- Korrektive Wartung ist eindeutig Reparatur und der Anteil an der Wartung, den man unter Softwarewartung gemeinhin versteht.
- Die Aufgabe der korrektiven Wartung ist das Beheben von Softwarefehlern, die nach der Auslieferung aufgetreten sind.
- Hierbei handelt es sich also um das Entfernen von Restfehlern.
- Nach der schwere der Fehler kann unterschieden werden:
  - Instandhaltung des Systems, also der Behebung schwerer Mängel, und
  - der Mängelkorrektur, also der Beseitigung mittelschwerer und leichter Mängel.

Präventive Wartung

- In der präventiven Wartung geht es darum, latente Fehler nach der Auslieferung der Software zu beheben, bevor sie im Feld als effektive Fehler aufgetreten sind.
- Dies ist eine proaktive Massnahme und daher planbar.
- Auch hierbei entfernen wir Restfehler, genau wie bei der korrektiven Wartung.

Adaptive Wartung

- Während der Lebensdauer einer Software kann sich die ursprüngliche Systemumgebung verändern, was dazu führt, dass man die Software diesen Veränderungen anpassen muss.
- Dabei muss zwischen der fachlichen und technischen Umgebung unterschieden werden.
- In der fachlichen Umgebung ändern sich Dinge wie Geschäftsabläufe oder Gesetze, was zu Anpassungen z.B. der Mehrwertsteuersätze führen kann.
- In der technischen Umgebung ändern sich Datenbanksysteme, Betriebssysteme oder Rechner und deren Komponenten.
- Plötzlich ist eine Betriebssystemversion nicht mehr erhältlich, und es muss auf die Nachfolgeversion umgestellt werden.
- Somit wirkt die adaptive Wartung dem «lack of movement» entgegen.

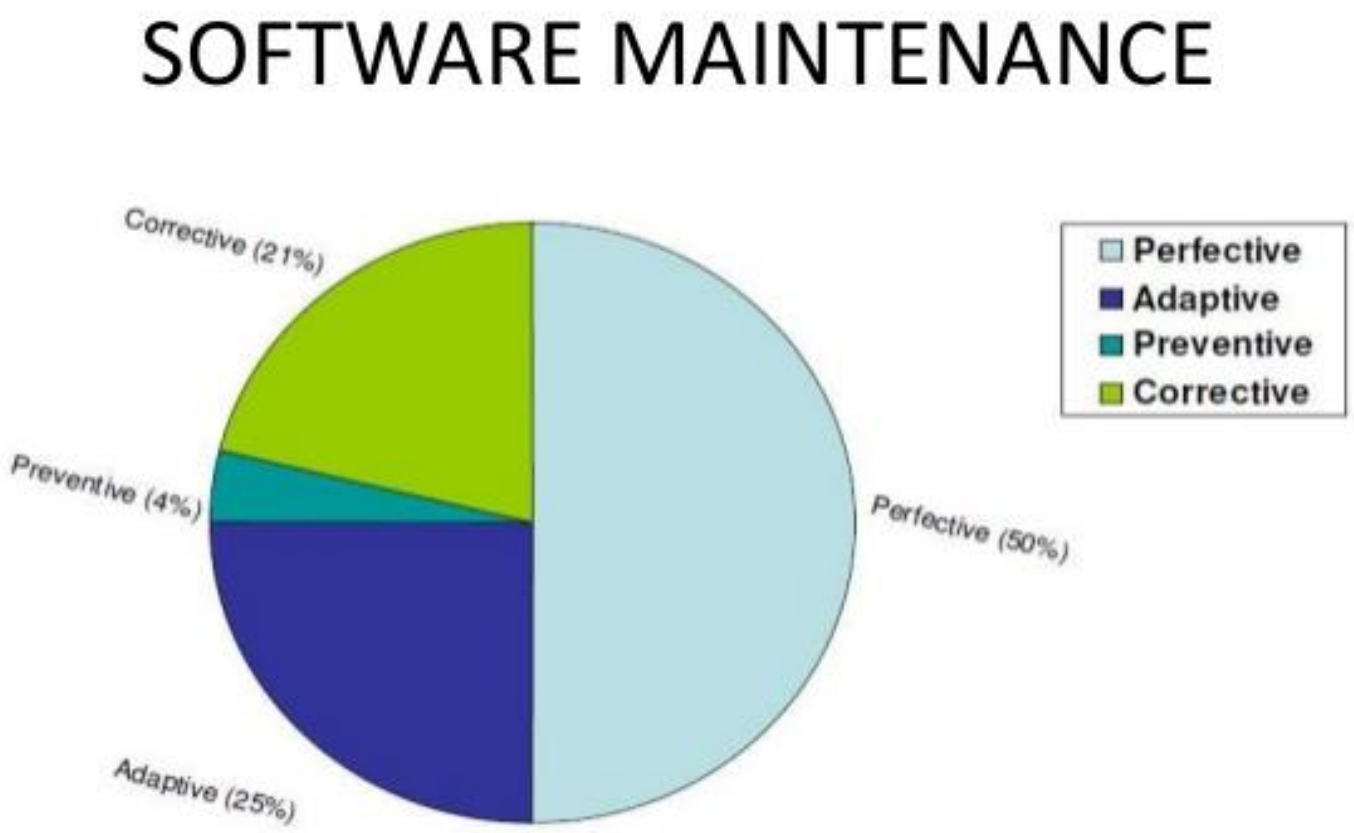
Perfektionierende Wartung

- Das Ziel der perfektionierenden Wartung ist die Verbesserung der Software. Dies kann gemäss der Norm [IEEE Std. 610.12-1990] «…die Performanz oder andere Attribute …» betreffen.
- Der Begriff andere Attribute öffnet ein weites Feld von Möglichkeiten, die die Wartbarkeit, die (Re-)Dokumentation oder andere Eigenschaften der Software betreffen.
- Hiermit sind auch Strukturverbesserungen im Rahmen von Refactoring, Reengineering und Kapselungen gemeint, um dem «ignorant surgery»-Problem entgegenzuwirken.
- Im Falle der proaktiven Wartung ist es wichtig, Indikatoren festzulegen, die die entsprechenden Wartungstätigkeiten auslösen. Unterlässt man dies, findet die proaktive Wartung kaum statt!

Support

- Die Supporttätigkeit umfasst im Wesentlichen den Aufwand, um Fragen und Probleme rund um den Einsatz der Software zu beantworten.
- Es handelt sich dabei also um eine Dienstleistung.
- Im Gegensatz dazu stehen die vier klassischen Tätigkeiten (korrektive, präventive, adaptive und perfektionierende Wartung), die den Code verändern.
- Deswegen wurde die Supporttätigkeit wohl in der Norm nicht als Teil der klassischen vier Wartungstätigkeiten erwähnt.
- Betrachtet man das Ganze aber aus Ressourcensicht, müssen sehr wohl Aufwände, und somit Mitarbeiter, für Supportaktivitäten berücksichtigt werden.

Wartungsaufwand – Klassische Verteilung

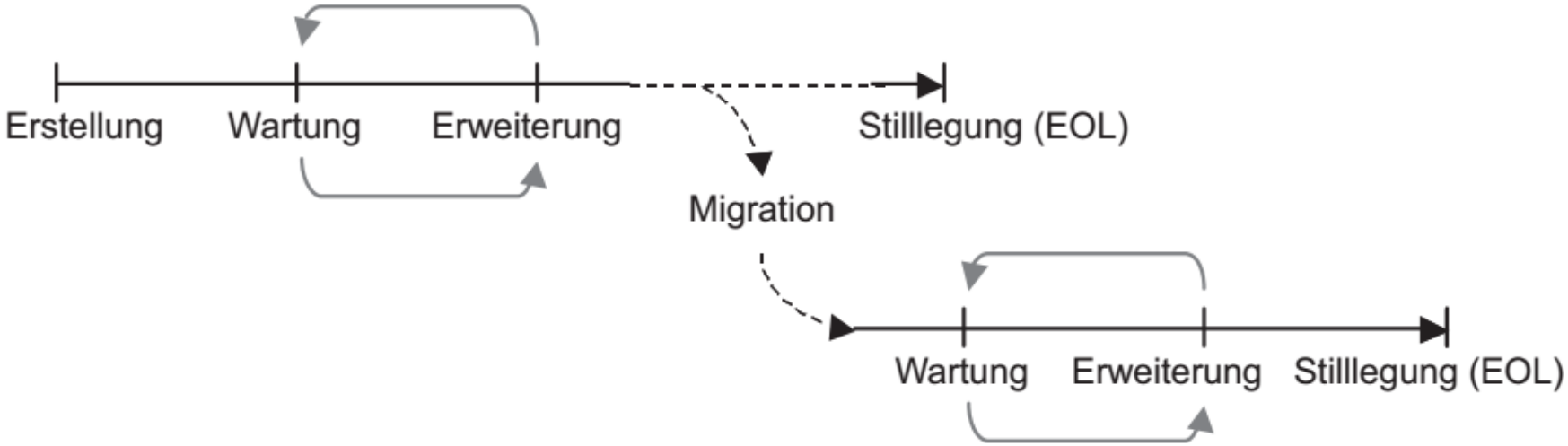


nach Nosek und Plavia, 1990

Agenda

1. Begriffswelt der Softwarewartung
2. **Evolution von Software**
3. Management der Softwarewartung
4. Techniken der Softwarewartung
5. Wrap-up

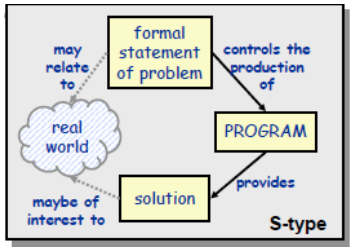
Softwareevolution

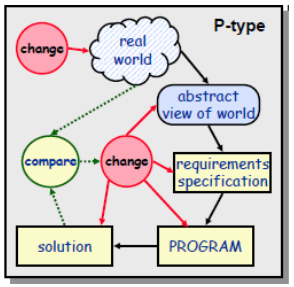


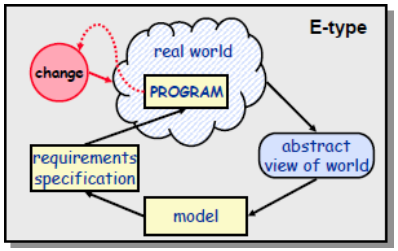
- Als **Softwareevolution** bezeichnen wir den Prozess der Veränderung eines Softwaresystems von der Erstellung bis zur Ausserbetriebnahme. Dieser Prozess umfasst somit die Erstellung, Wartung, Weiterentwicklung, Migration und Stilllegung**.** (nach Lehmann und Belady, 1985)

Drei Klassen von Software (Meir M. Lehman 1980)

- **S-Typ** (specifable): Vollständig durch formale Spezifikation beschreibbar (z.B. Sortieralgorithmus).
  - Erfolgskriterium: Spezifikation nachweislich erfüllt
- **P-Typ** (problem solving): Löst ein abgegrenztes Problem
  (z.B. Schachprogramm).
  - Erfolgskriterium: Problem zufriedenstellend gelöst
- **E-Typ** (embedded): Realisiert eine in der realen Welt eingebettete Anwendung (z.B. CRM-Applikation).
  - Erfolgskriterium: Anwender zufrieden







Quelle: adaptiert von Programs, Life Cycles, and Laws of Software Evolution, Meir M. Lehmann, 1980

- Software vom S-Typ ist stabil
- Software vom P- und E-Typ ist einer Evolution unterworfen durch den Wandel des Umfelds

Eigenschaften von S-, P- und E-Software

Aspekt	S-Typ	P-Typ	E-Typ
Komplexität	klein	mittel – hoch	klein – hoch
Größe	klein	groß	groß
Fokus	Problem	Problem	Prozess
Problemänderung	keine	keine	ja
Äußerer Druck	nein	ja	ja
Druck auf Umgebung	nein	nein	ja

- *Es ist unmöglich, Software vom P- oder E-Typ so zu entwickeln, dass sie während einer mehrjährigen Lebensdauer ohne Änderungen ihre Aufgaben zur vollen Zufriedenheit der Benutzer erfüllt.*

Gesetze der Softwareevolution (1/2)

- I. Gesetz der kontinuierlichen Veränderung
  Ein E-Typ-System, das genutzt wird, muss kontinuierlich anpasst werden, ansonsten wird es sehr schnell nicht mehr nutzbar sein.
- II. Gesetz der zunehmenden Komplexität
  Die Komplexität einer Software wächst stetig an, ausser es wird Arbeit investiert, um diese Komplexität zu reduzieren.
- III. Gesetz der Selbstregulierung
  Die Evolution von Software unterliegt einer sich selbstregulierenden Dynamik mit statistisch bestimmbaren Tendenzen und Invarianzen.
- IV. Gesetz der invarianten Arbeitsrate
  Die durchschnittliche Aktivitätsrate, die für die Wartung eines Systems erbracht wird, ist während der gesamten Lebenszeit nahezu konstant.
- V. Gesetz der Erhaltung der Ähnlichkeit
  Die Inhalte von aufeinanderfolgenden Releases sind während der gesamten Lebenszeit statistisch konstant.

Gesetze der Softwareevolution (2/2)

- VI. Gesetz des kontinuierlichen Wachstums
  Der Funktionsumfang eines Systems muss kontinuierlich zunehmen, um den Ansprüchen der Nutzer auf Dauer gerecht zu werden.
- VII. Gesetz der abnehmenden Qualität
  Die Qualität der Software wird abnehmend empfunden, solange die Software nicht rigoros instand gehalten und an die sich ändernde Umgebung angepasst wird.
- VIII. Gesetz des Feedback-Systems
  E-Typ-Entwicklungsprozesse bilden ein System mit mehreren Feedback-Ebenen und müssen als solche wahrgenommen und behandelt werden, um sie erfolgreich ändern und verbessern zu können.

Der Einfluss der Erstentwicklung

- Die Gesetze von Lehman und Belady beziehen sich immer auf E-Typ-Systeme im eingeschwungenen Zustand.
- Hier spielen die beiden Kräfte der Erweiterung und der vorbeugenden Pflege eine zentrale Rolle.
- Bei der Betrachtung dieser Gesetze fehlt aber der Einfluss der Erstentwicklung auf den weiteren Verlauf der Evolution.
- Wie bei einem Menschen, sind auch bei einem Softwaresystem die «richtigen» Gene – sprich eine gute Erstentwicklung – das A und O.
- Folgende Faktoren beeinflussen also die Lebenserwartung:
  - *Die Qualität der Erstentwicklung (tiefe Entropie = besser)*
  - *Die Wachstumsrate (kleiner = besser)*
  - *Die vorbeugende Pflege (mehr = besser)*

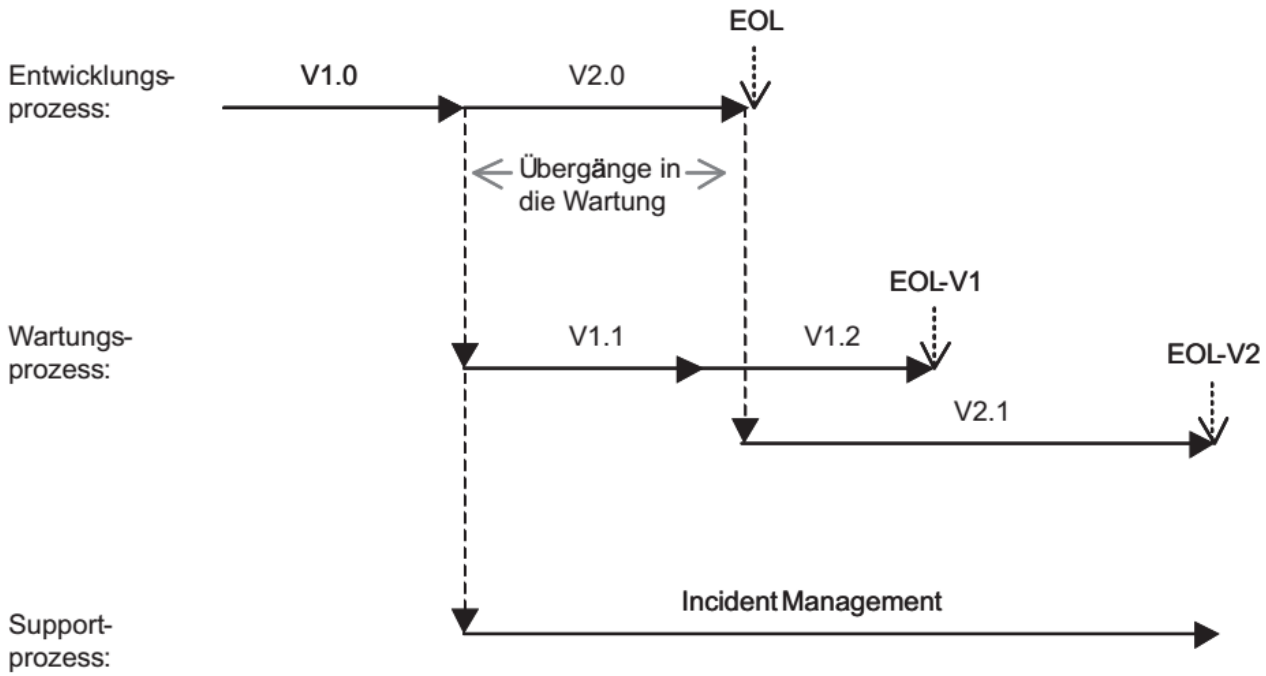
Konsequenzen für P- und E-Software aus den Gesetzen

- Bewegliche Ziele sind kein Unfall, sondern naturbedingt (unsere Projekte sind meistens vom E-Typ).
- Bei innovativen, neuen Produkten und Systemen müssen die Anforderungen zuerst gefunden werden – vielfach durch Prototyping und partielle Lösungen.
- Bei längeren Projekten sind Änderungen der Anforderungen während der Entwicklung wahrscheinlich.
- Software ist nie über mehrere Jahre hinweg gebrauchstauglich ohne Veränderung.
  - Softwarewartung ist kein Unfall, sondern unvermeidlich.
  - Überlegungen zu Software immer auf den ganzen Softwarelebenszyklus beziehen.
- Software, die für den Kunden einen dauerhaften Nutzen darstellen soll, muss gleichermassen gepflegt und weiterentwickelt werden.

Agenda

1. Begriffswelt der Softwarewartung
2. Evolution von Software
3. **Management der Softwarewartung**
4. Techniken der Softwarewartung
5. Wrap-up

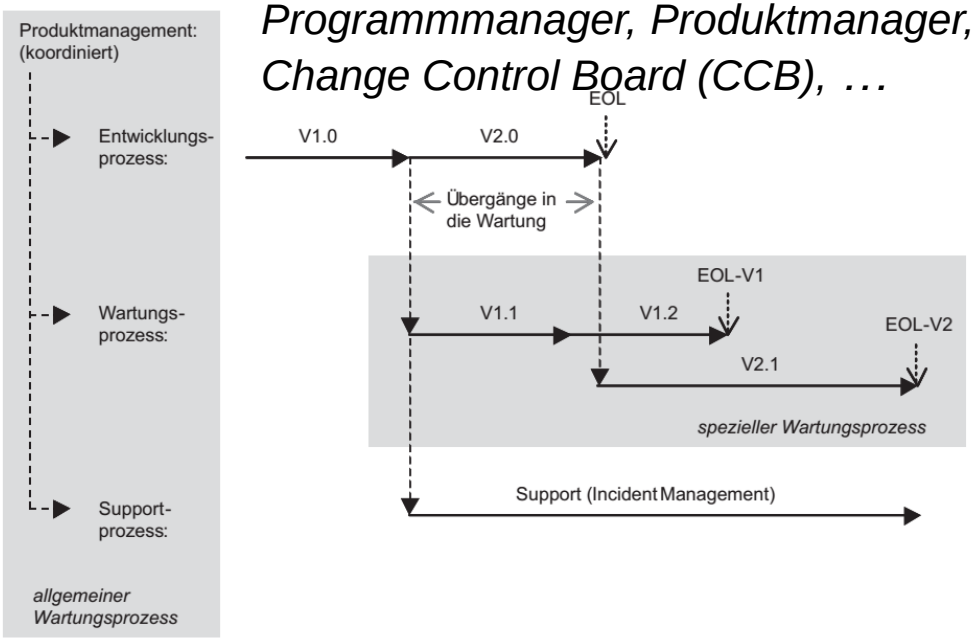
Der Wartungsprozess im Kontext



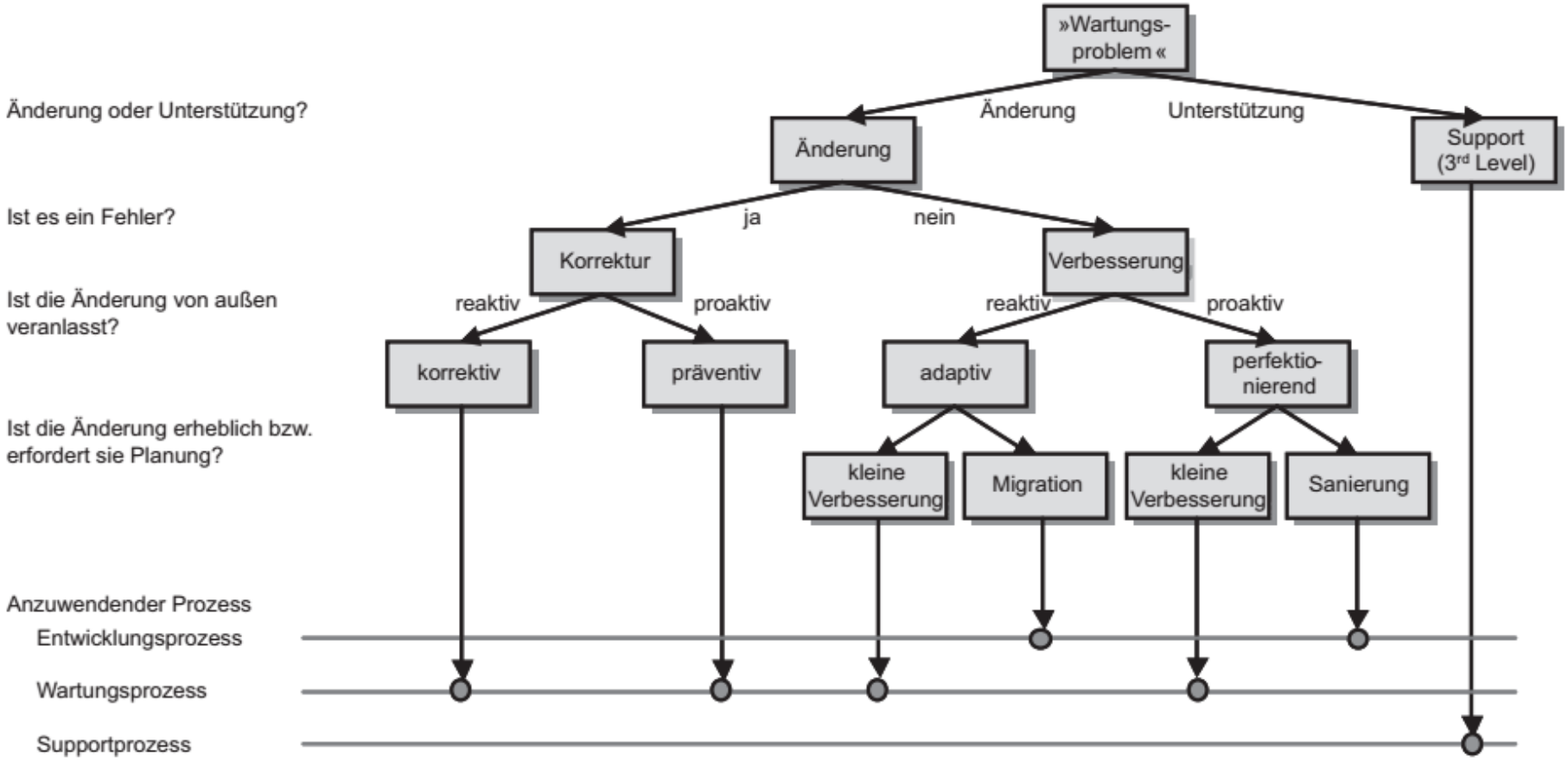
- Bei Software mit einem langen Lebenszyklus müssen Weiterentwicklung, Wartung und Betrieb bzw. Support erfolgreich in Einklang gebracht werden.

Was genau ist der Wartungsprozess?

- Allgemeiner Wartungsprozess: der Prozess, der die Gesamtheit derjenigen Tätigkeiten organisiert, die der Wartung zugeordnet werden können.
- Spezieller Wartungsprozess: der Prozess, der die Tätigkeiten rund um die Wartungsäste des Codes organisiert.



Grundgerüst eines allgemeinen Wartungsprozesses



Fazit Wartungsprozess

- Wir kennen nun die drei Lebenslinien und ihre Hauptziele, die es in der Wartung in Einklang zu bringen gilt:
  - Entwicklung sorgt durch Funktionserweiterung für Marktfähigkeit.
  - Wartung sorgt für die Lieferfähigkeit.
  - Support sorgt für die Kundenzufriedenheit.
- Wichtige Aspekte des allgemeinen Wartungsprozesses sind:
  - Der Produktmanager ist das Oberhaupt des allgemeinen Wartungsprozesses.
  - Das CCB (Change Control Board) ist das wichtigste Organ der Wartung, da es über den weiteren Verlauf jedes einzelnen Wartungsfalls entscheiden muss.
- Der spezielle Wartungsprozess hat u.a. folgende Eigenschaften:
  - Er ist eine geeignet zugeschnittene Form eines Entwicklungsprozesses.
  - Er greift auf dieselben unterstützenden Prozesse wie der Entwicklungsprozess zu (Qualitätssicherung, Dokumentation etc.).

Agenda

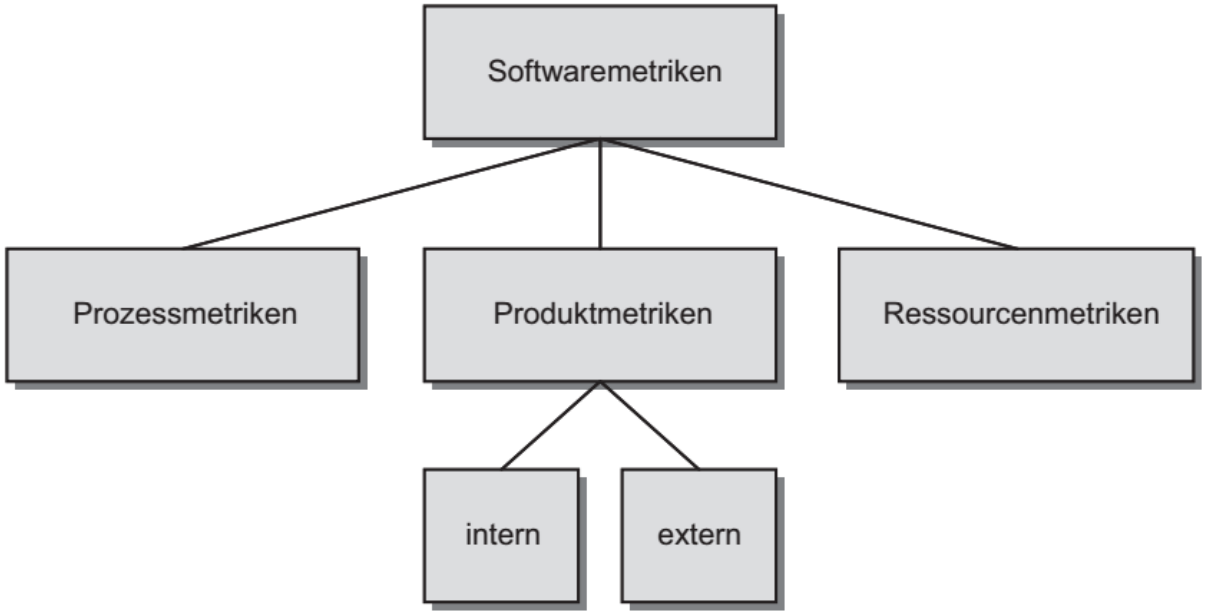
1. Begriffswelt der Softwarewartung
2. Evolution von Software
3. Management der Softwarewartung
4. **Techniken der Softwarewartung**
5. Wrap-up

Übersicht über Techniken der Softwarewartung

- Metriken
- Softwareanalyse und -visualisierung
- Reengineering, Sanierung und Migration

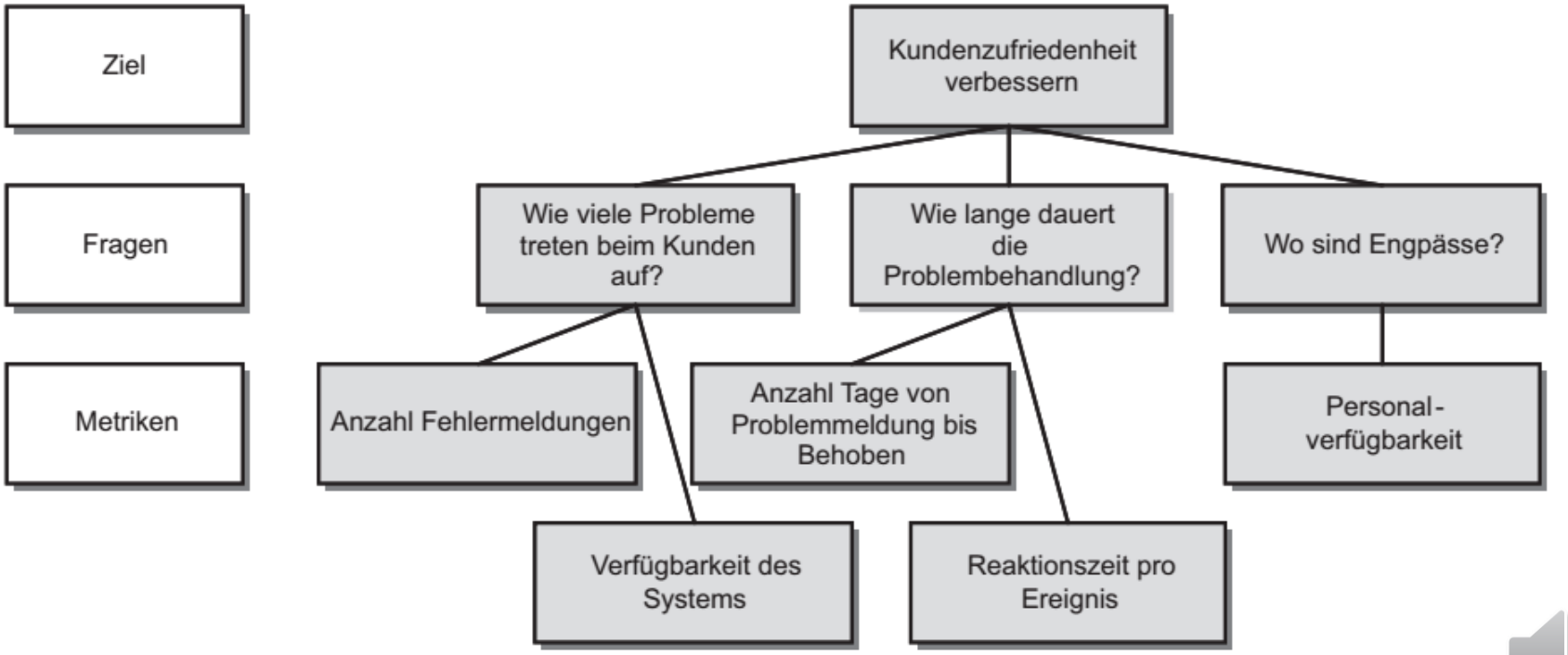
Metriken

- Eine Softwarequalitätsmetrik ist eine Funktion, die eine Softwareeinheit in einen Zahlenwert abbildet.
- Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Softwareeinheit.

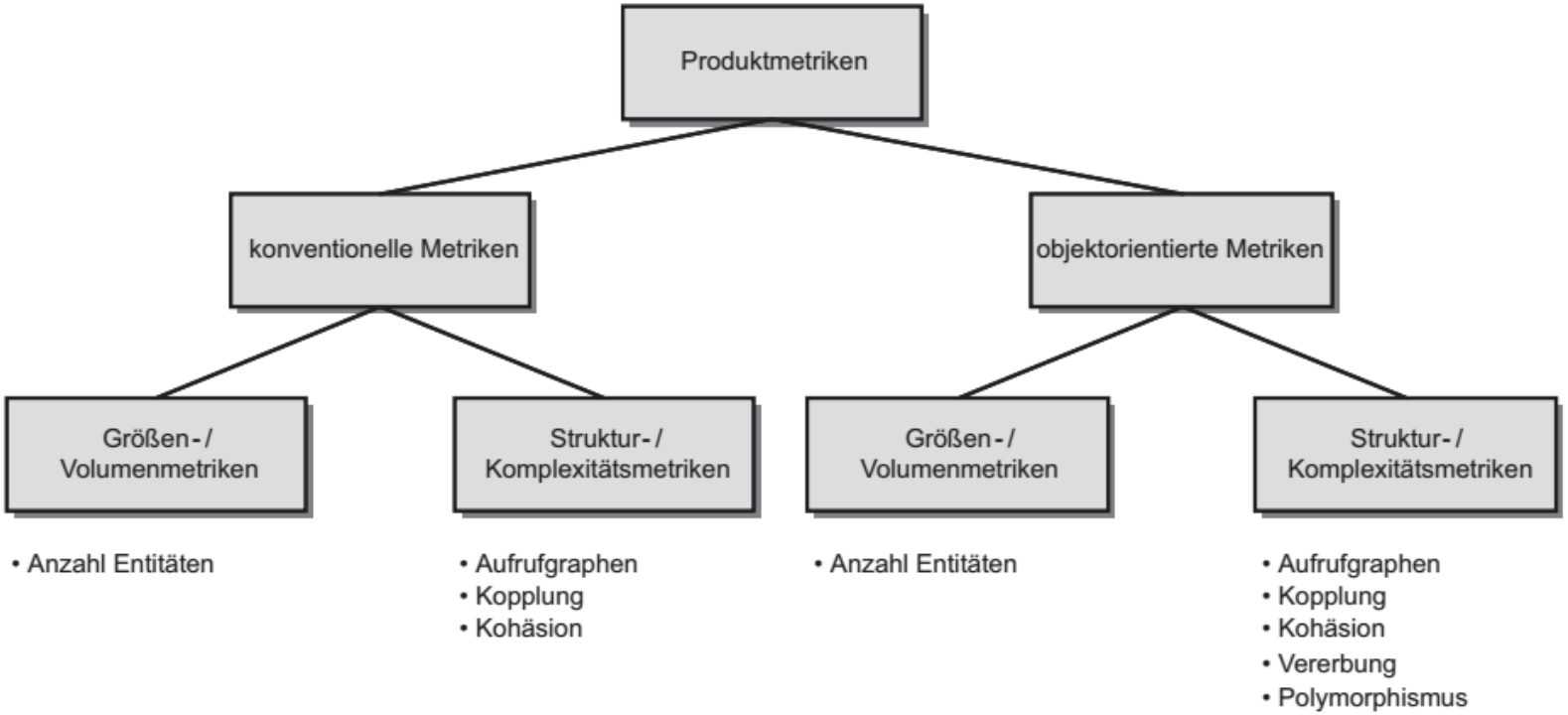


Wie findet man eine geeignete Metrik?

- Der GQM-Ansatz (Goal-Question-Metric) hilft uns, zielgerichtete Metriken zu finden, und verhindert das Erheben von irrelevanten und unsinnig vielen Daten, ist aber mit einigem Aufwand verbunden.

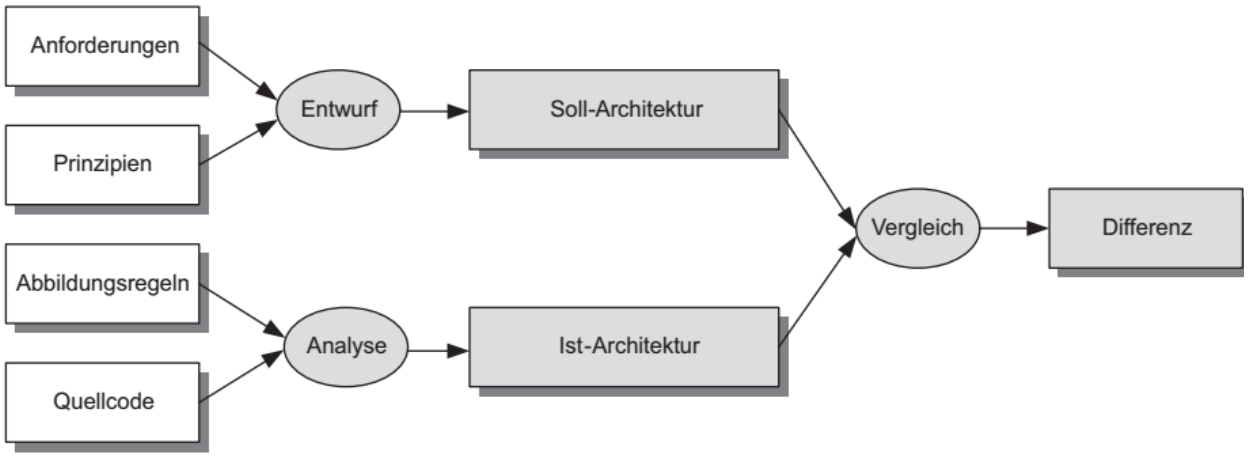


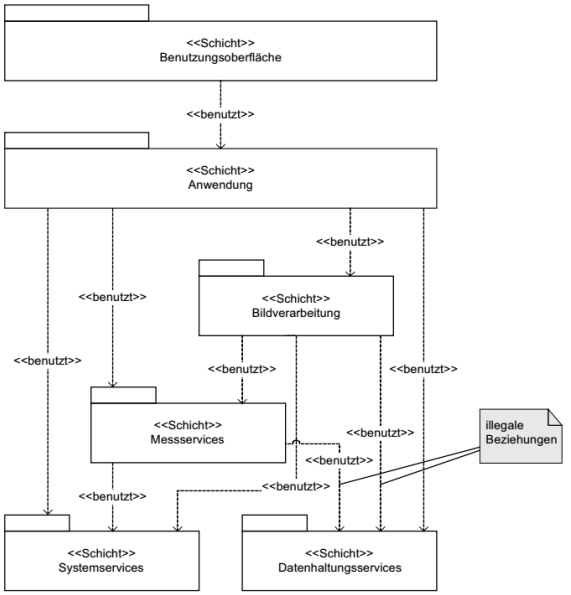
Beispiel Produktmetrikarten



- Sie unterstützen bei folgenden Vorhaben:
  - Feststellen von Degenerationen (ignorant surgery lässt grüssen)
  - Überwachen des Fortschritts von Sanierungsmassnahmen

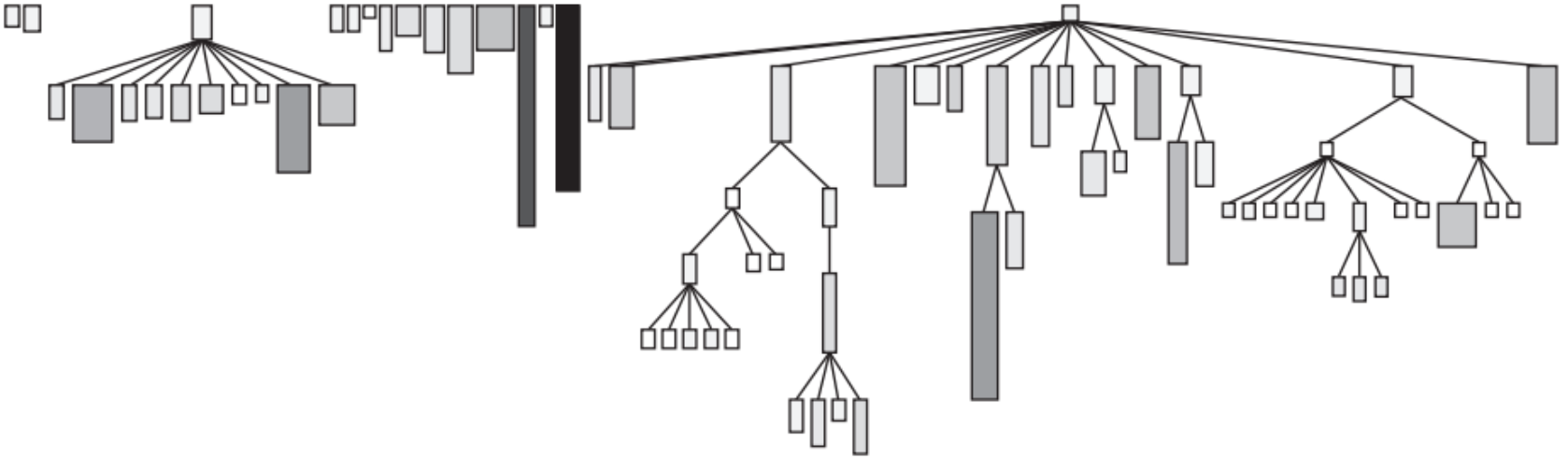
Softwareanalyse und –visualisierung
Architekturanalyse





- Die Architekturanalyse ist ein Mittel, strukturelle Degeneration zu erkennen, dessen Nutzen umso grösser ist, je früher mit der Analyse im Lebenszyklus der Software begonnen wird.
- Werkzeuge: Sonargraph (ex. Sotograph) u.a.

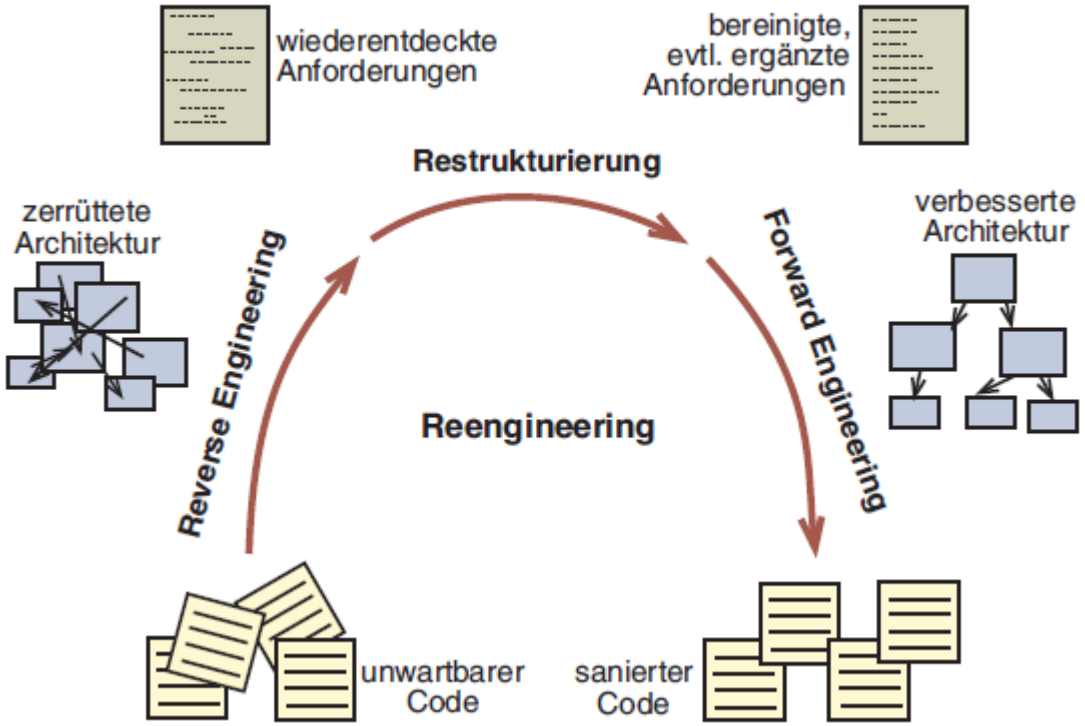
Softwareanalyse und –visualisierung
Codeanalyse



- Codeanalysen helfen dabei, Eigenschaften der Software direkt auf der Ebene Quellcode zu verstehen (z.B. Systemkomplexität als polymetrische Ansicht).
- Werkzeuge: CodeCrawler, GraphViz u.a.

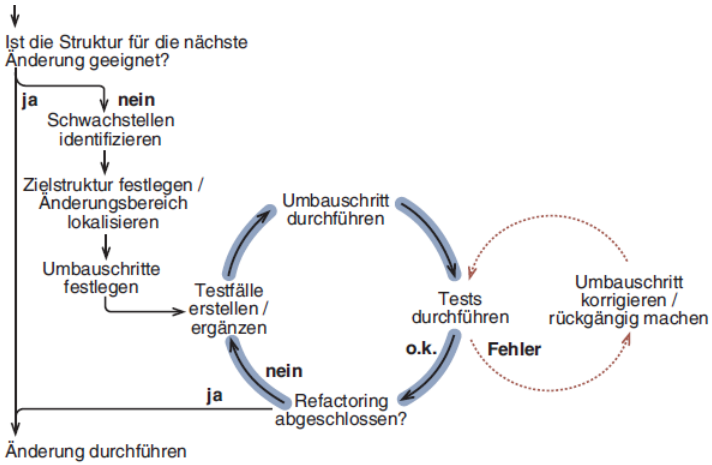
Reengineering

- Reengineering ist der Prozess der Überarbeitung eines Softwaresystem oder einzelner Teile davon.



Verbesserung der Wartbarkeit mit Refactorings

- Refactorings sind kontinuierliche Restrukturierungsmassnahmen. Sie haben das Ziel, die Produktivität bei zukünftigen Wartungs- und Erweiterungstätigkeiten zu verbessern. Damit sind sie als perfektionierende Wartung zu verstehen.
- Refactorings werden häufig als Reaktion auf sogenannte «Bad Smells» durchgeführt. Bad Smells bedeutet etwa so viel wie «Das riecht nach einem Problem» (Anomalie).
- In der agilen Softwareentwicklung werden Bad Smells schnellstmöglich beseitigt (keine Anhäufung von «*technical debts*»!).



Sanierung und Migration

- Sanierung und Migration sind grössere Wartungsvorhaben.
- Softwaresanierung hat als primäre Ziele die Komplexitätsreduktion und Qualitätsverbesserung.
- Durch die Sanierung wird also die Wirtschaftlichkeit und Leistungsfähigkeit des Softwaresystems wiederhergestellt und damit seine Überlebensfähigkeit gesichert – oder anders ausgedrückt: Der Lebenszyklus des Systems wird verlängert.
- Bei einer Migration wird die Software von einer bestehenden in eine andere Umgebung versetzt.
- Auslöser für eine Migration sind meistens geänderte Anforderungen an das Softwaresystem, die in der bestehenden Umgebung nicht mehr oder nur mit grossem Aufwand realisierbar sind.
- Migration ist ein Beispiel adaptiver Wartung.
- Ziel der Migration ist es, die Software auf einen meist einheitlichen, zeitgemässen technologischen Stand zu bringen.

Agenda

1. Begriffswelt der Softwarewartung
2. Evolution von Software
3. Management der Softwarewartung
4. Techniken der Softwarewartung
5. **Wrap-up**

Wrap-up

- Der Anteil der Softwarewartung macht einen beträchtlichen Aufwand an den Gesamtkosten über den ganzen Software-Lebenszyklus aus.
- Softwarewartung wird in die Kategorien korrektive, präventive, adaptive und perfektionierende Wartung unterteilt.
- Die Treiber des Softwareevolutionsprozesses sind die Erweiterung bzw. Weiterentwicklung und Wartung.
- Aus den Gesetzen der Softwareevolution folgt, dass ein benutztes Softwaresystem sich permanent ändert und aktiv dem Zerfall der Software entgegengewirkt werden muss.
- Der Entwicklungs- Wartungs- und Supportprozess müssen von Anfang miteinander geplant werden.
- Techniken der Softwarewartung sind Metriken, Softwareanalysen und -visualisierung sowie Reengineering, Sanierung und Migration.

Quellen- und Literarturverzeichnis

- C. Bommer, M. Spindler, V. Barr: Softwarewartung – Grundlagen, Management und Wartungstechniken, dpunkt.verlag, 2008
- H. M. Sneed, R. Seidl: Softwareevolution – Erhaltung und Fortschreibung bestehender Softwaresysteme, dpunkt.verlag, 2013
- M. M. Lehman: Programs, Life Cycles, and Laws of Software Evolution, Proceedings of the IEEE, 1980
- M. M. Lehmann, L. Belady: Program Evolution: Processes of Software Change, Academic Press, 1985
- IEEE Std 14764-2006 Software Engineering - Software Life Cycle Processes – Maintenance
- IEEE Std 1219-1998 Standard for Software Maintenance
- IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology