Klausur

CARL VON OSSIETZKY universität OLDENBURG

Termin

- Klausur (120 min)
  - Dienstag, 04.02.2020, 14:30-15:30 Uhr, A14 1-101/2 (Hörsaal 1/2)
- Wiederholungsklausur (120 min)
  - Montag, 06.04.2020, 14:30-16:30 Uhr, A11 1-101 (Hörsaal B)

Teilnahmebedingungen

- rechtzeitige Anmeldung zur Klausur
  - Anmeldung zur Klausur möglich bis zum 28.01.2020 (MEZ)
  - Abwesenheit bei der Klausur trotz Anmeldung gilt als Fehlversuch
- Leistungsnachweis (6 KP-Punkte)
- Erreichen von 45% der Gesamtpunkte der Klausur

Wegweiser

Software Fehler

- Oberbürgermeister-Wahl in Neu-Ulm
- Ausfall einer Autobatterie
- Netzwerk-Absturz
- Zusammenbruch eines Aktien-Handelssystems
- Ausfall der USS Yorktown
- Ariane 4
- Patriot-Rakete






Fehlerursachen (Zusammenfassung)

- typische Programmierfehler (z.B. Speicherüberläufe)
- fehlende Qualitätssicherung (z.B. Test nach Erweiterung)
- Inkompatibilitäten verschiedener Systemkomponenten
- schlechtes UI-Design
- inkorrekte/unvollständige Fehlerbehandlung
- Wiederverwendung von Code, der unter anderen Voraussetzungen entwickelt wurde, und deren Verwendungsvoraussetzungen unklar waren
- Eingriffe in Softwaresysteme durch unqualifizierte Mitarbeiter
- Fehlerhafte Nutzung

Folgerungen für die Software-Entwicklung (1)

- Anforderungen an ein Softwaresystem sind präzise, nachprüfbar, und möglichst eindeutig festzulegen
- Vorgehensweisen und Zuständigkeiten zur Software-Entwicklung sind genau festzulegen
- Entwurfsentscheidungen sind für spätere Weiterentwicklungen und Wiederverwendungen zu dokumentieren
- Rahmenbedingungen der Softwarenutzung sind festzulegen und zu dokumentieren

Folgerungen für die Software-Entwicklung (2)

- Tatsächliche Realisierung der gestellten Anforderungen (incl. Usability) durch das Softwaresystem ist zu überprüfen
- Korrektes Zusammenspiel aller Systemkomponenten ist sicherzustellen
- Softwaresysteme sind gegen typische Programmierfehler abzusichern
  (Vorsicht: Fehler in Fehlerbehandlungsroutinen)
- Software ist soweit wie möglich gegen Fehlbedienungen und Fehlinterpretationen abzusichern

1.4 Probleme der Software-Entwicklung

Eigenschaften von Software

CARL VON OSSIETZKY universität OLDENBURG

Software ist immateriell

- Software ist nicht sichtbar
- Software wird entwickelt, nicht fabriziert
- Software-Fertigung erfolgt durch Kopieren
- Software-Produktion
  - wird durch Entwurf und Implementierung bestimmt,
  - wird nicht durch die Fertigung bestimmt

Software ist variabel

- Programmcode ist leicht änderbar (?)
- Entwicklungsfortschritt ist schwer messbar
- Anforderungen ändern sich während der Entwicklung
- Software entwickelt sich weiter (Software-Evolution)

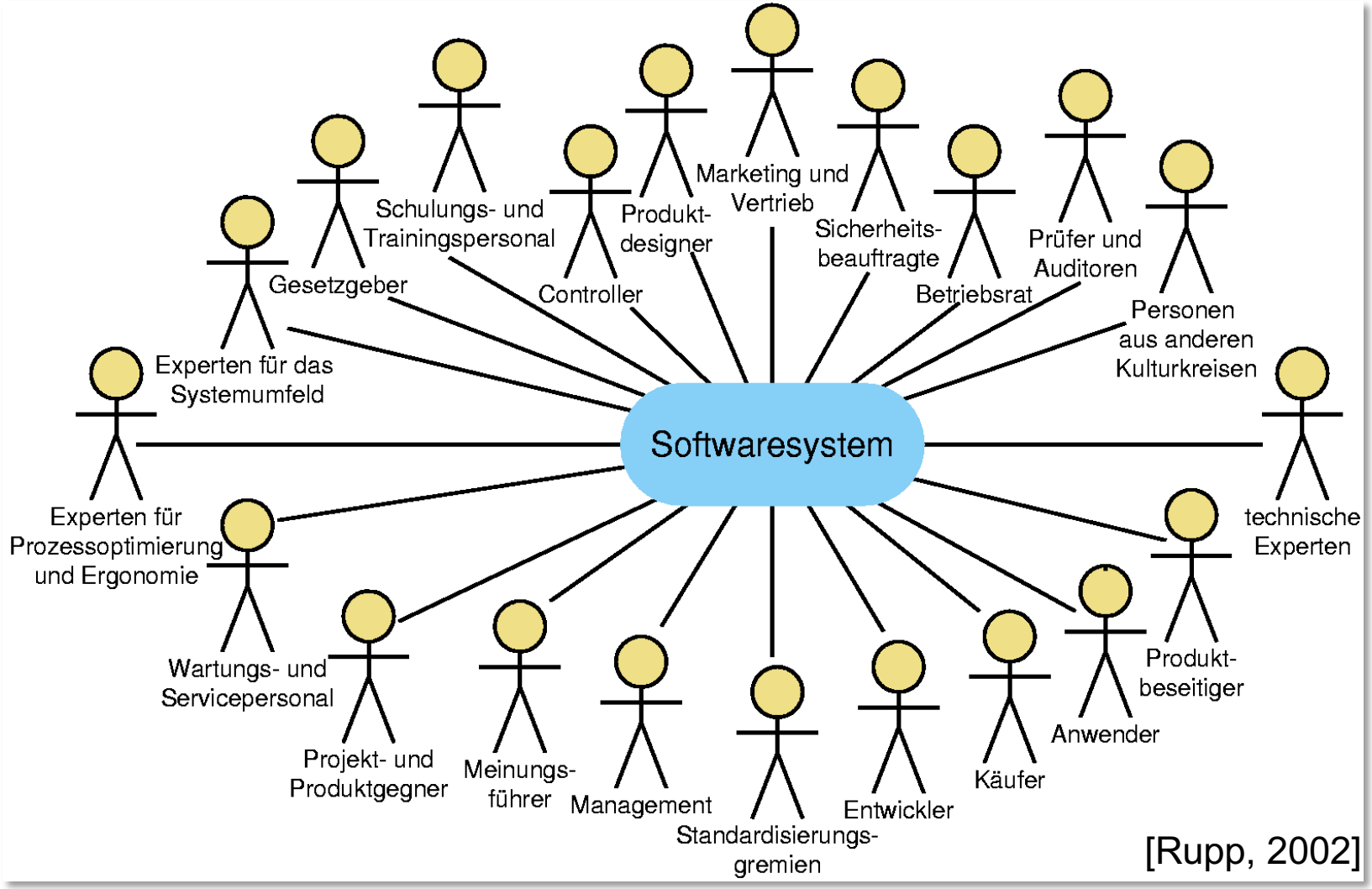
Beteiligte bei der Software-Entwicklung

CARL VON OSSIETZKY universität OLDENBURG

Stakeholder

- (Systembeteiligte, Systembetroffene, Wissensträger, Interessenvertreter ...)
- sind alle Personen und Vertreter von Organisationen, die von Einsatz und Erstellung des zu erstellenden Softwaresystems betroffen sind
- verfolgen unterschiedliche Ziele durch die Entwicklung und Verwendung des Software-Systems
- verwenden unterschiedliche Sprachen zur Formulierung ihrer Ziele und den daraus abgeleiteten Systemanforderungen

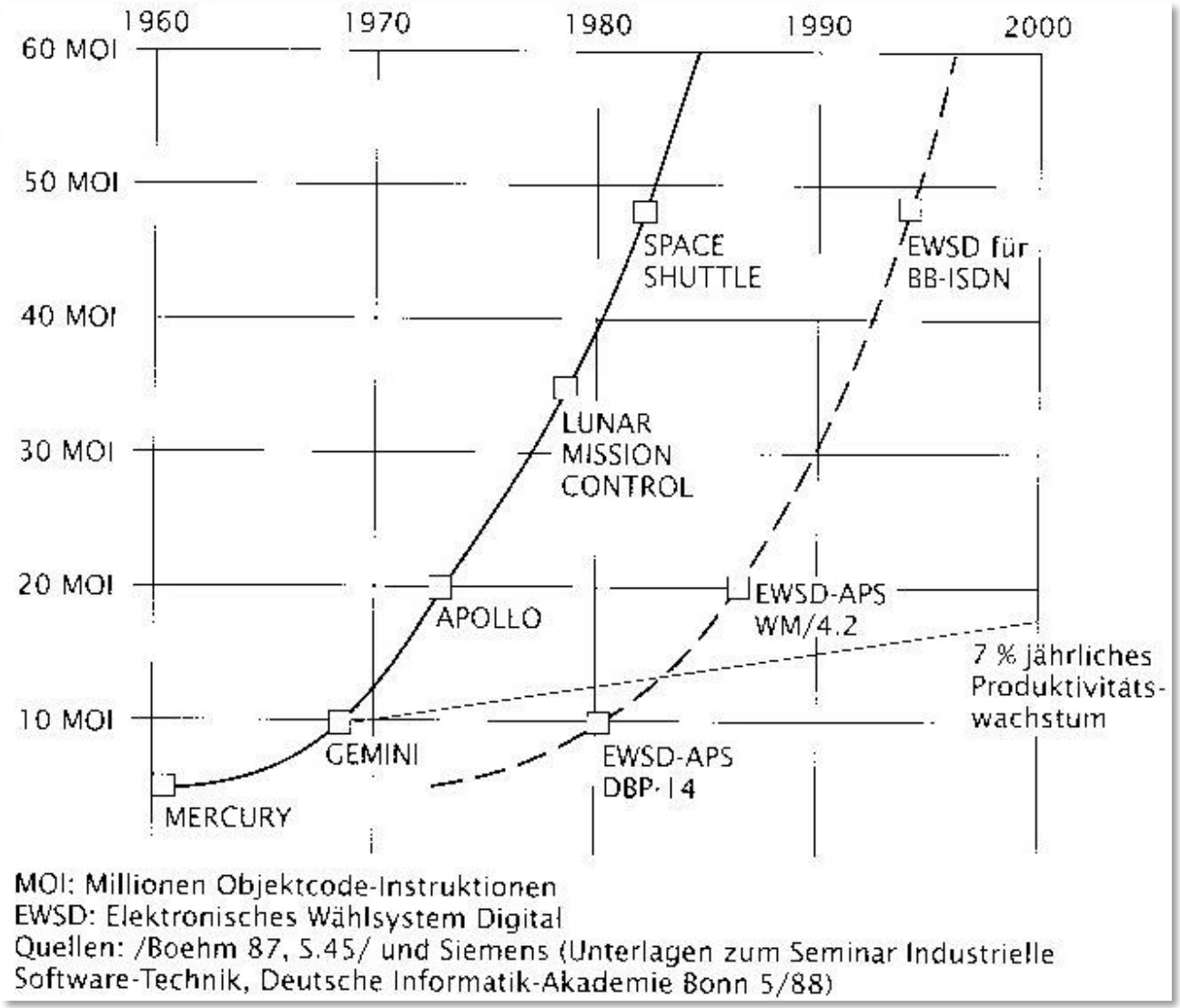
Stakeholder



Komplexität und Umfang

Softwaresysteme

werden zur Bearbeitung immer komplexerer und umfangreicherer Aufgaben eingesetzt



[Balzert, 2001, S. 30]

Anforderungen an Softwaresysteme

CARL VON OSSIETZKY universität OLDENBURG

fehlendes Anwendungswissen bei Entwicklern

- Entwickler kennen den Anwendungsbereich nicht
- Entwickler denken, sie verstehen die Probleme der Nutzer besser als die Nutzer selbst

fehlendes Informatikwissen bei Auftraggebern

- Auftraggeber formulieren unrealistische Anforderungen
- Auftraggeber haben unrealistische Kosten- und Aufwandsvorstellungen

unklare Anforderungen bei Auftraggebern

- Anwender wissen nicht was sie wollen
- Anwender wissen was sie wollen, können das aber nicht artikulieren
- Anwender wissen solange genau was sie wollen, bis sie es bekommen

Veränderungen während der Software-Erstellung

Produktanforderungen

- geänderte Anforderungen durch neue Erkenntnisse
- zusätzliche/geänderte Funktionalität durch zusätzliche/geänderte Aufgaben
- geänderte (gesetzliche) Rahmenbedingungen

Systemumgebung

- zusätzliche/modernere Hardware
- zusätzliche/geänderte Systemsoftware

Entwicklungsumgebung

- neue (modernere) Entwicklungsmethoden
- neue Entwicklungswerkzeuge
- neue Programmiersprachen
- neue Mitarbeiter

Evolution von Softwaresystemen

CARL VON OSSIETZKY universität OLDENBURG

Alterung von Softwaresystemen

- keine Abnutzung von Softwaresystemen
- keine (klassische) Wartung von Softwaresystemen durch Austausch von Verschleißteilen

(Weiter-) Entwicklung von Softwaresystemen

- ständige Anpassung an sich verändernde Anforderungen und Rahmenbedingungen

"Unsere Softwaresysteme werden nicht gewartet, sie werden weiterentwickelt"
(Softwareentwickler zu Wartungsaktivitäten, Oktober 2003)

Anforderungen an den Software-Entwicklungsprozess

CARL VON OSSIETZKY universität OLDENBURG

Funktionstreue:

- geliefertes Softwaresystem muss die definierten Produktanforderungen erfüllen

Qualitätstreue:

- geliefertes Softwaresystem muss den definierten Qualitätsanforderungen genügen

Termintreue:

- Softwaresystem muss zum definierten Fertigstellungstermin ausgeliefert werden

Kostentreue:

- Entwicklung des Softwaresystems muss im definierten Kostenrahmen erfolgen

Folgerungen für die Software-Entwicklung (3)

Bereitstellen und Verwenden von Verfahren, Methoden und Techniken

- zur funktions-, qualitäts-, termin- und kostentreuen Entwicklung und Weiterentwicklung (Wartung, Evolution) von Softwaresystemen
- zum Umgang mit sich ständig ändernden, unvollständigen und nicht ausreichenden Anforderungen und Software-Erstellungsumgebungen
- zur Berücksichtigung der Interessen aller relevanter Stakeholder (Manager, Benutzer, Software-Entwickler)

Schaffen eines Problembewusstsein

- in allen Stakeholder-Gruppen für die speziellen Probleme der Software-Entwicklung

1.4 Ingenieurdisziplin "Softwaretechnik"

Ingenieurdisziplin "Softwaretechnik"

Software-Krise:

- Software Entwicklung in den 60er Jahren
  - Zunahme der Komplexität benötigter Softwaresysteme
  - keine (geeigneten) Programmiersprachen
  - keine geeigneten Methoden, Techniken und Werkzeuge zur Software Entwicklung

Ingenieurdisziplin "Softwaretechnik"

CARL VON OSSIETZKY universität OLDENBURG

"The whole trouble comes from the fact, that there is so much tinkering with software. It is not made in a clean fabrication process, which it should be. [...] What we need is software engineering."

[F. L. Bauer: Software Engineering, Garmisch, October 7-11, 1968]

"[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines."

[F. L. Bauer: Software Engineering, Garmisch, October 7-11, 1968]

Ingenieurdisziplin "Softwaretechnik"

Ingenieur

- wissenschaftlich gebildete Person im technischen Bereich, die technische Gegenstände, Verfahren, Anlagen oder Systeme erforscht, plant, entwirft, konstruiert, fertigt, vertreibt, überwacht oder verwaltet. [Brockhaus]

Ingenieurdisziplin "Softwaretechnik"

CARL VON OSSIETZKY universität OLDENBURG

Ingenieurmäßiges Vorgehen

- folgt etablierten und bewährten Prinzipien
- wendet Methoden gezielt an
- sucht marktorientiert einen optimalen Kompromiss zur Problemlösung
- lehnt "Künstlertum" ab

Ingenieurmäßiges Vorgehen der Softwaretechnik

- Anforderungen erheben und definieren
- Modell spezifizieren
- Lösung(en) konstruieren und bewerten
- Ergebnis prüfen

Definitionen

IEEE (1993):

- Software Engineering:
  "(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1)". [IEEE Std. 601.12-1990, 1993]

Balzert (2001):

- [Softwaretechnik ist die] "zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung von umfangreichen Softwaresystemen" [Balzert, 2001]

Definitionen

CARL VON OSSIETZKY universität OLDENBURG

Boehm (1976):

- "Software Engineering: The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them" [Boehm, 1976]

Parnas (1987):

- "[Software Engineering is] multi-person construction of multi-version software" [nach Ghezzi, et al., 1991]

Hesse et al. (1984):

- "Softwaretechnik ist die Bereitstellung und systematische Verwendung von Methoden und Werkzeugen für die Herstellung und Anwendung von Software".

Definition "Softwaretechnik"

CARL VON OSSIETZKY universität OLDENBURG

Softwaretechnik

- befasst sich mit der
  - wissenschaftlichen Entwicklung von Prinzipien, Techniken, Methoden und Werkzeugen zur Softwareentwicklung und
  - mit der Anwendung dieser Mittel zur Erstellung, zum Betrieb und zur Wartung von umfangreichen Softwaresystemen.

Fähigkeiten des Softwaretechnikers

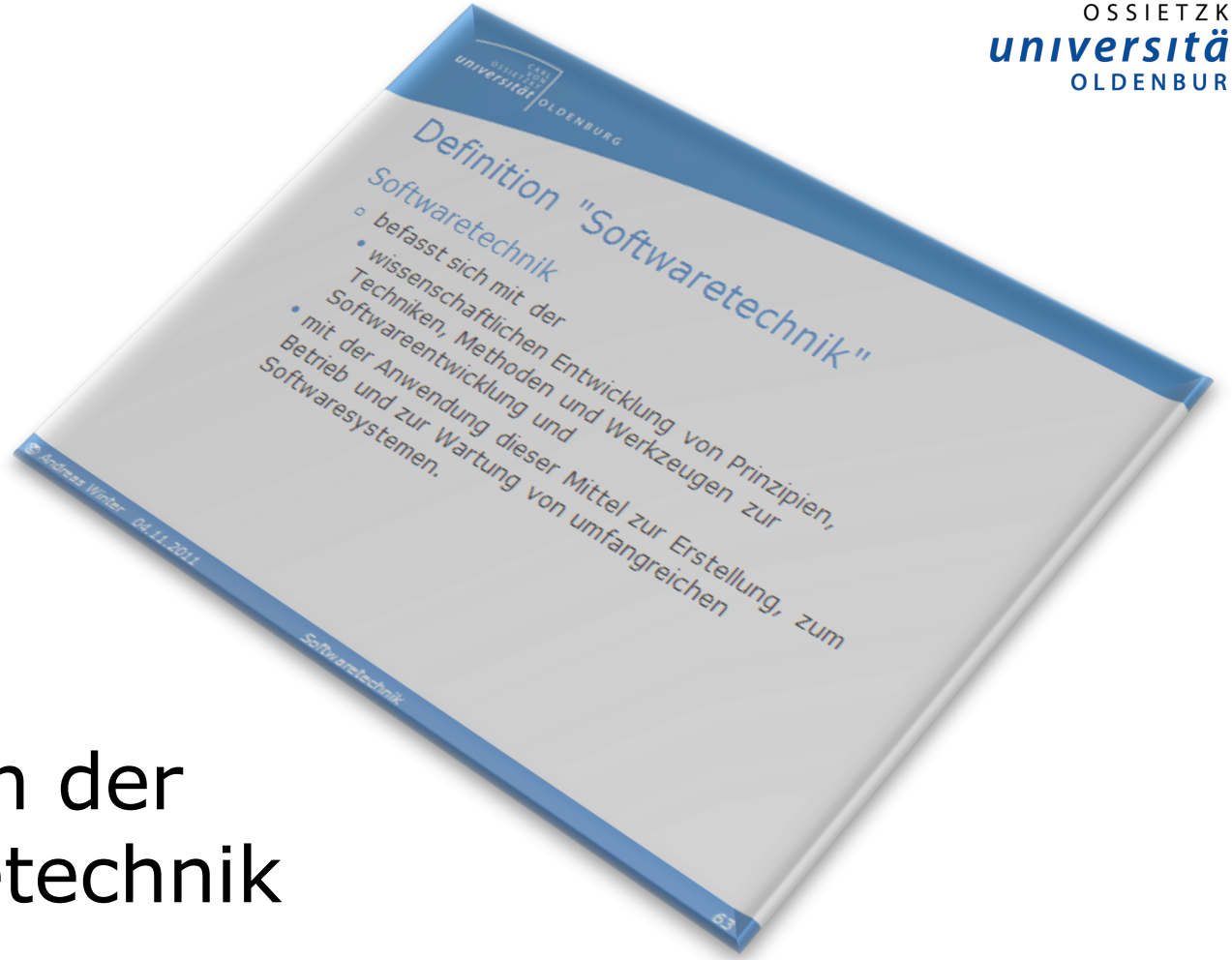
Programmierung im Kleinen

- Programmierfähigkeiten (in mehreren Sprachumgebungen)
- Kenntnissen von Algorithmen und Datenstrukturen

Programmierung in Größen

- hervorragende Fähigkeiten zur Kommunikation mit unterschiedlichen Personengruppen (Auftraggeber, Anwendern und Softwareentwicklern)
- hervorragende Kenntnis verschiedener Methoden und Techniken zur Softwareanalyse und zum Softwareentwurf
- hervorragende Fähigkeiten zur Erstellung von Systemmodellen für Kommunikation und Realisierung
- hervorragende Fähigkeiten zur Projektkoordination auf unterschiedlichen Abstraktionsebenen

[vgl. Ghezzi,1991 , S. 5f]



1.6 Prinzipien der Softwaretechnik

[vgl. Ghezzi,1991, S. 43ff
Balzert 1998, S. 557ff]

Grundlegende Definitionen

Prinzipien

- sind Grundsätze, die man seinem Handeln zugrunde legt

Techniken

- sind planmäßig angewandte, begründete Vorgehens-weisen zur Erreichung von festgelegten Zielen

Methoden

- spiegeln grundsätzliche Denk- und Vorgehensweisen wider und fassen Techniken (und Methoden) geeignet zusammen

Werkzeuge

- dienen der automatischen Unterstützung von Methoden

Prinzipien

Prinzip der Analogie

- ähnliche Dinge sollen ähnlich behandelt werden

Vorteile

- standardisiert Beschreibungen und Vorgehensweisen
- gibt Strukturierungshilfen
- verdeutlicht Parallelen und Gemeinsamkeiten ähnlicher Sachverhalte
- erleichtert Nachvollziehen und Verstehen komplexer Sachverhalte

Prinzipien

Prinzip der Analogie

Lone Pine Koala Sanctuary
Brisbane, QLD, Austalia, 22.11.2000

Prinzipien

Prinzip der Zerlegung

- komplexe Dinge sind geeignet
  - in weniger komplexe Dinge und
  - hierzwischen vorliegende Beziehungen
- zu zerlegen

Vorteile

- erleichtert Umgang mit komplexen Systemen
- erleichtert Einarbeitung und Bearbeitung komplexer Dinge
- erleichtert Festlegung von Verantwortlichkeiten

Prinzipien zur Zerlegung

CARL VON OSSIETZKY universität OLDENBURG

Prinzip der Trennung der Belange (separation of concerns)

- Dinge, die nicht zusammen gehören, sind auch getrennt voneinander zu behandeln

Vorteile

- reduziert Problemkomplexität
- ermöglicht die separate und unabhängige Behandlung individueller Problemaspekte
- fokussiert die Betrachtung auf einzelne Aspekte
- ermöglicht Verteilung von Verantwortlichkeiten gemäß der Belange

Prinzipien zur Zerlegung

CARL VON OSSIETZKY universität OLDENBURG

Prinzip der Abstraktion

- Dinge sind möglichst unter Betonung des Wesentlichen zu betrachten.
- Dinge, die gemeinsame Eigenschaften haben, sind zu identifizieren und gemeinsam zu betrachten

Vorteile

- fördert die Erkennung gemeinsamer Eigenschaften
- trennt Unwesentliches von Wesentlichem
- ermöglicht gemeinsame Betrachtung ähnlicher Dinge

Prinzipien

Prinzip der Abkapselung (Geheimnisprinzip)

- interne Details, die für die aktuelle Betrachtung unwesentlich sind, sind zu verbergen

Vorteil

- trennt Implementierung von Verwendung
- erhöht Zuverlässigkeit des Systems, da die Verwendung ausschließlich über fest definierte Schnittstellen erfolgt
- ermöglicht leichteren Austausch der "Internas"

Prinzipien

Prinzip der Lokalität

- Dinge, die zusammen gehören, sind auch gemeinsam, an einem Ort zu behandeln

Vorteile

- fasst alle relevanten Aspekte eines Sachverhaltes an einem Ort zusammen
- ermöglicht leichteres Auffinden zusammengehörender Aspekte

Prinzipien

Prinzip der Vollständigkeit

- alle möglichen Fälle und Ausprägungen sind zu erheben und zu betrachten

Vorteil

- verhindert, dass einzelne Aspekte übersehen werden
- sorgt für die Betrachtung solcher Aspekte, die (zunächst) als unwesentlich eingestuft werden
- macht Ignorieren von (unwesentlichen) Aspekten bewusst
- dient zur Qualitätssicherung

Prinzipien

Prinzip der Nachweisbarkeit

- Ziele sind hinsichtlich ihrer Zielerreichung zu überprüfen

Vorteil

- erfordert nachprüfbare Zieldefinitionen
- macht Projektfortschritt nachweisbar
- erleichtert den Nachweis über abgelieferte Leistung incl. deren Korrektheit
- dient zur Qualitätssicherung

Prinzipien

Prinzip der Redundanz-Vermeidung

- identische Dinge sollen nur einmal behandelt werden

Vorteile

- Datenbestände bleiben nach Änderung konsistent
- Änderungen müssen nur einmal erfolgen
- Konsistenz-Erhaltung erfordert keinen zusätzlichen Aufwand

Gründe zur bewussten Prinzip-Verletzung

- Verdeutlichung komplexer Sachverhalte
- Optimierung (Bsp.: Bearbeitungszeit, Layout)
- eigene Kontrolle über sich verändernde Versionen

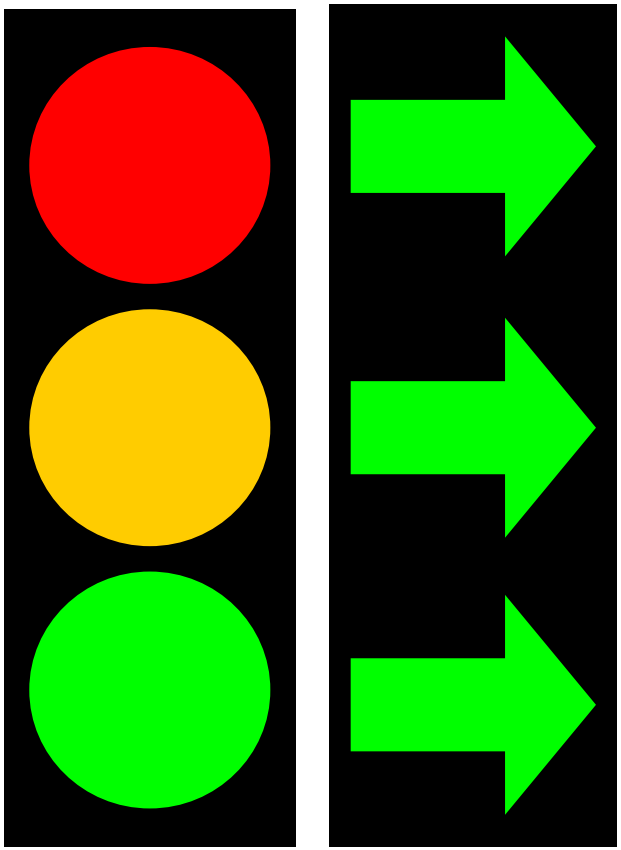
Prinzipien Konflikte

Wohin gehört der grüne Pfeil (Grünpfeil)?

Prinzip der Lokalität

Prinzip der Vollständigkeit

Prinzip der Analogie



Der grüne Pfeil

[Straßen-Verkehrsordnung StVO §37(2)]

"Rot ordnet an: "Halt vor der Kreuzung". Nach dem Anhalten ist das Abbiegen nach rechts auch bei Rot erlaubt, wenn rechts neben dem Lichtzeichen Rot ein Schild mit grünem Pfeil auf schwarzem Grund (Grünpfeil) angebracht ist. Der Fahrzeugführer darf nur aus dem rechten Fahrstreifen abbiegen. Er muss sich dabei so verhalten, dass eine Behinderung oder Gefährdung anderer Verkehrsteilnehmer, insbesondere des Fußgänger- und Fahrzeugverkehrs der freigegebenen Verkehrsrichtung, ausgeschlossen ist."
[http://www.verkehrsportal.de/stvo/stvo_37.php3]

Zusammenfassung: Prinzipien

CARL VON OSSIETZKY universität OLDENBURG

Prinzip der Analogie

- ähnliche Dinge sollen ähnlich behandelt werden

Prinzip der Zerlegung

- komplexe Dinge sind geeignet in weniger komplexe Dinge und hier-zwischen vorliegende Beziehungen zu zerlegen

Prinzip der Trennung der Belange

- Dinge, die nicht zusammen gehören, sind auch getrennt voneinander zu behandeln

Prinzip der Lokalität

- Dinge, die zusammen gehören, sind auch gemeinsam, an einem Ort zu behandeln

Prinzip der Abstraktion

- Dinge, die gemeinsame Eigenschaften haben, sind zu identifizieren und gemeinsam zu betrachten

Prinzip der Abkapselung (Geheimnisprinzip)

- interne Details, die für die Anwendung unwesentlich sind, sind zu verbergen

Prinzip der Nachweisbarkeit

- Ziele sind hinsichtlich ihrer Zielerreichung zu überprüfen

Prinzip der Vollständigkeit

- alle möglichen Fälle und Ausprägungen sind zu erheben und zu betrachten