Software Engineering

Lernskript

Dozent:
Prof. Dr. Stefan Kramer

LaTeX von:
Sven Bamberger

Zuletzt Aktualisiert:
22. Februar 2014

Dieses Skript wurde erstellt, um sich besser auf die Klausur vorzubereiten.

Dieses Dokument garantiert weder Richtigkeit noch Vollständigkeit, da es aus Mitschriften und Vorlesungsfolien gefertigt wurde und dabei immer Fehler entstehen können. Falls ein Fehler enthalten ist, bitte melden oder selbst korrigieren und neu hoch laden.

Inhaltsverzeichnis

1 Introduction to Software Engineering

1.1 Was versteht man unter Software-Engineering?

Software Engineering ist eine technische Disziplin, die sich mit allen Aspekten der Softwareherstellung beschäftigt.

1.1.1 Technische Disziplin:

- Techniker wenden Methoden, Werkzeuge und Theorien an um Dinge zum Laufen zu bringen
- Techniker erkennen an, dass sie mit oganisatorischen und finanziellen Beschränkungen arbeiten müssen

1.1.2 Alle Aspekte der Softwareherstellung:

- SE beschäftigt sich nicht nur mit technischen Prozessen der Softwareentwicklung
- auch Projektverwaltung und Entwicklung von Werkzeugen, Methoden und Theorien, welche die Sofwareherstellung unterstützen

1.1.3 Softwareprozess (grundlegende Aktivitäten von SE):

- Softwarespezifikation
  - Kunden und Entwickler definieren die zu produzierende Software und die Rahmenbedingung fr ihren Einsatz
- Softwareentwicklung
  - Software wird entworfen und programmiert
- Softwarevalidierung (Softwarebeurteilung):
  - Software wird überprüft um sicherzustellen, dass sie den Kundenanforderungen entspricht
- Softwareevolution
  - Software wird weiterentwickelt, damit sie die sich ändernden Bedürfnisse der Kunden und des Marktes wiederspiegelt

1.1.4 Zwei grundlegende Arten von Softwareprodukten:

- Allgemeine Prdoukte:
  - eingeständige Systeme die am freien Markt an jeden Kunden verkauft werden kann, der es sich leisten kann
  - Beispielsweise: Software für den PC, Datenbanken, Textverarbeitungsprogramme
- Angepasste Produkte:
  - Software die im Auftrag eines bestimmten Kunden hergestellt werden
  - Biespielsweise: Steuerungssysteme für elektronische Geräte, System zur Unterstützung eines bestimmten Geschäftprozesses

1.1.5 Unterschiede dieser Softwaretypen:

- Allgemeine Produkte:
  - Unternehmen, das Software entwickelt, bestimmt die Spezifikationen der Software
- Angepasste Produkte:
  - Spezifikation wird von dem Kunden entwickelt und kontrolliert, dass das System kauft

1.1.6 Viele unterschiedliche Anwendungsarten:

- Eigenständige (stand-alone) Anwendungen:
  - Softwware läuft lokal
  - Beinhaltet alle notwendigen Funktionen um ohne Netzwerkverbindung zu laufen
  - z.B. Fotoanwendungen ohne Netzwerkzugriff
- Interaktive transaktionsbasierte Anwendungen:
  - laufen auf externen PC's und werden vom User aufgerufen
  - Webapplikationen über Cloud-Zugriff
- Eingebettete Steuerungssysteme:
  - es gibt daon mehr als von allen andern Anwendungsarten
  - Software für ABS im Auto
  - Kontrollieren die Hardware
- Stapelverarbeitende Systeme (Batch-Verarbeitungssysteme)
  - Systeme die große Datenmengen verarbeiten (z.B. Business-Systeme)
  - Abrechnung von regelmäßigen Zahlungen (Telefonrechnung)
- Unterhaltungssysteme
  - zum persönlichen Gebrauch
  - z.B. Spiele
- Systeme für die Modellierung und Simulation
  - Systeme, die von Wissenschaftlern und Ingenieuren entwickelt wurden
  - z.B. physikalische Prozesse simulieren
- Datenerfassungssysteme
  - System, welches Daten unter extremen Bedingungen mit Hilfe von Sensoren erfassen und weiterleiten
- Systeme von Systemen
  - Zusammensetzung von vielen Softwaresystemen

1.1.7 Wesentliche Merkmale guter Software:

- Wartbarkeit
  - kritisches Merkmal
  - Software sollte so geschrieben werden, dass sie **weiterentwickelt werden kann**, um sich verändernden Kundenbedüfnissen Rechnung zu tragen
  - Softwareanpassungen sind eine unvermeidliche Anforderung einer sich verändernden Geschäftsumgebung
- Verlässlichkeit (dependability) und Informationssicherheit (security):
  - Verlässliche Software sollte keinen körperlichen oder wirtschaftlichen Schaden verursachen, falls das System ausfällt
  - Böswillige Benutzer sollten nicht in der Lage sein, auf das System zuzugreifen oder es zu beschädigen
- Effizienz (efficiency):
  - Software sollte nicht verschwenderisch mit Systemressourcen wie Speicher und Prozessortkapazität umgehen
  - Effizien umfasst somit Reaktionszeit, Verarbeitungszeit, Speichernutzung usw.
- Akzeptanz (acceptability)
  - Software muss von den Benutzern akzeptiert werden, für die sie entwickelt wurde. Das bedeutet, dass sie verständlich, nützlich und kompatibel mit anderen Systemen sein müssen, die diese Benutzer verwenden.

Abbildungsverzeichnis