Software Engineering
1. Einführung und Begriffe

Prof. Dr. Klaus Ostermann

Agenda

- Organisatorisches
- Begriffsklärung: Softwaretechnik
- Aufbau der Vorlesung

Organisatorisches

Organisation der LV

- Umfang: 2 SWS mit ~ 8 Vorlesungen
- Begleitend zum Teamprojekt
- Termine:
  - VL: Mi 14.15 – 15.45 Uhr hier
  - Klausur: 26.06.2018, 14:00 Uhr, Hörsaal N6
  - Nächster Termin (24.4.) fällt aus!
- Scheinkriterien
  - Abschlussklausur
  - Muss zum Abschluss des Programmierprojekts bestanden werden
  - Fliesst in die Note ein (Details folgen)
- Homepage der LV:
  - http://ps.informatik.uni-tuebingen.de/teaching/ss19/se/
    Kopien der Folien, Literaturhinweise, ...

Termin heute

- Vorlesung bis 14.50 Uhr
- 14.50-15.00: Vorstellung „Tübinger Softwareprojekt“
- 15:00-15:05 Pause
- Ab 15:10: Vorstellung der Projekte zum „Tübinger Softwareprojekt“ durch unsere Industriepartner
- Es gibt noch (wenige) freie Plätze!
- Ab 16:10: Software Engineering Messe im Foyer Hörsaalgebäude
- Es gibt **reichlich** zu Essen und zu trinken!

Lehrveranstaltungsstil

- Konzeptvermittlung durch Folien
- Folienkopien sind auf der Homepage verfügbar, Abweichungen (insb. Korrekturen) sind möglich
- Beispiele häufig an der Tafel
- Zwischenfragen und Kommentare während der Vorlesung sind grundsätzlich erwünscht.
- „Einsatz in der Praxis“ im Teamprojekt/TSP
- Gastvorträge von hochkarätigen Dozenten aus der Industrie (22.5. CapGemini zu Microservices)
- Kommen Sie zur Vorlesung!
  - Und werden Sie nicht zur Belastung für Ihre Teampartner

Inhalt

- Software Engineering:
  - Systeme, Projekte, Fachgebiet
- Vorgehens- und Prozessmodelle
- Anforderungsanalyse
- Softwareentwurf: Modellieren von Softwaresystemen
- Systementwurf:
  - SW-Architekturen, Entwurfsmuster und Komponenten
- OO-Konzepte für Fortgeschrittene
- Software-Qualitätssicherung:
  - speziell Testen von Softwaresystemen
- Projektmanagement

Begriffe und Kontext

Vorgehen?

- Woher kommt die Problembeschreibung?
- Beschreibt sie wirklich das Problem des Nutzers?
- Welche Struktur soll das fertige Programm haben?
- Löst das Programm wirklich das Problem? Funktioniert es korrekt?
- Was passiert mit neuen Anforderungen des Benutzers?
- Wie teilt man die Arbeit bei mehreren Entwicklern?

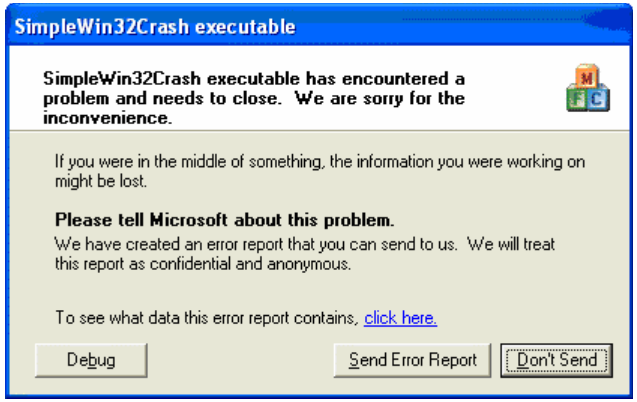
Softwaretechnik (engl. Software Engineering)

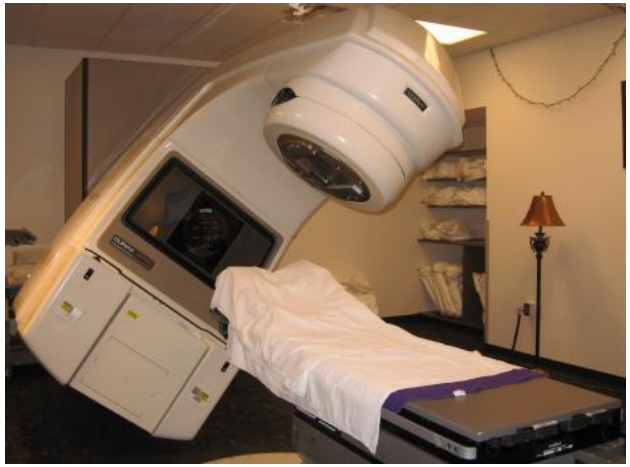
- Etabliertes Teilgebiet der Informatik
  - Anforderungsanalsyse
  - Entwurf und Entwicklung von Software, und Werkzeuge dafür
  - Organisation und Strukturierung der Entwicklung, Projektmanagement
  - Qualitätssicherung
  - Betrieb und Wartung von Systemen
  - uvm.

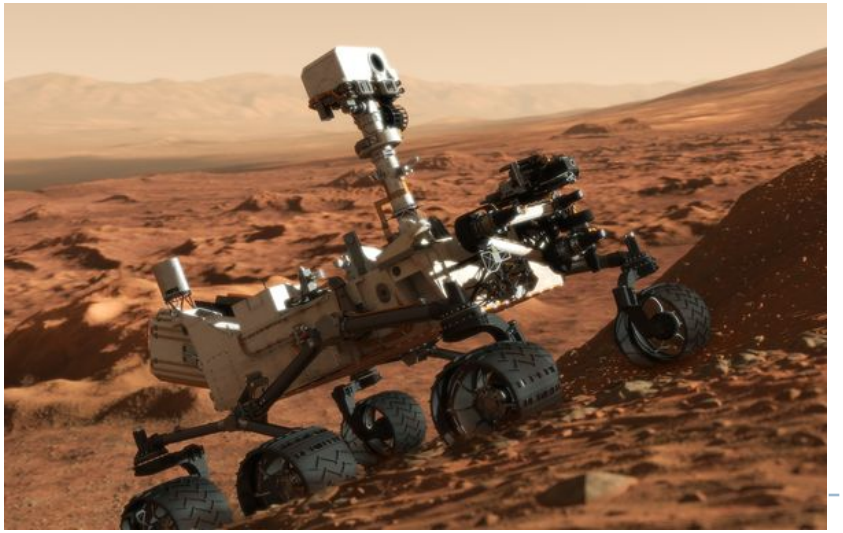
Software-Krise

- Mitte 1960er Jahre
- Mit schnellerer Hardware wurde Software wichtiger
- Steigende Anforderungen, aber
  - Qualifiziertes Personal fehlte
  - Softwareentwicklung durch “Bastelei”
  - Software war unzuverlässig, ständige Wartung
  - Bestehende Systeme intransparent und unübersichtlich, kaum änderbar
  - Kosten und Dauer überstiegen Erwartungen
  - Anforderungen oft nicht erfüllt
- Softwarekosten überstiegen Hardwarekosten
- Große Softwareprojekte scheiterten

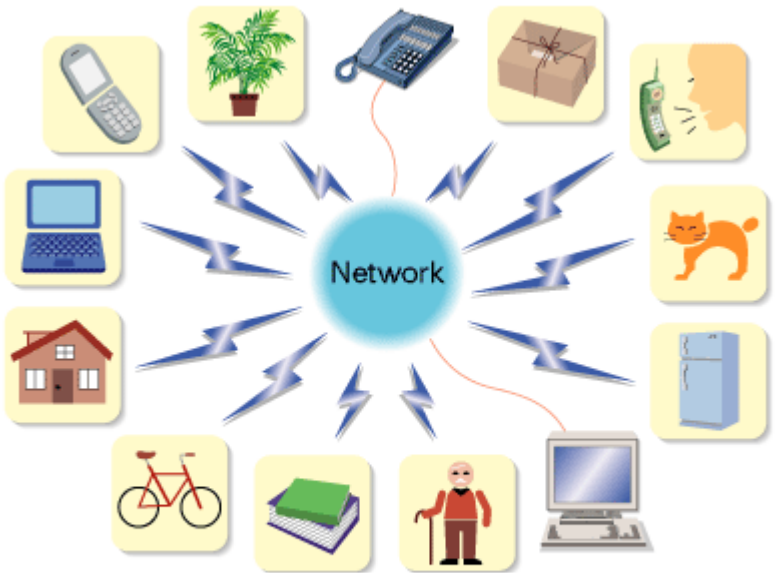
Software-Krise II

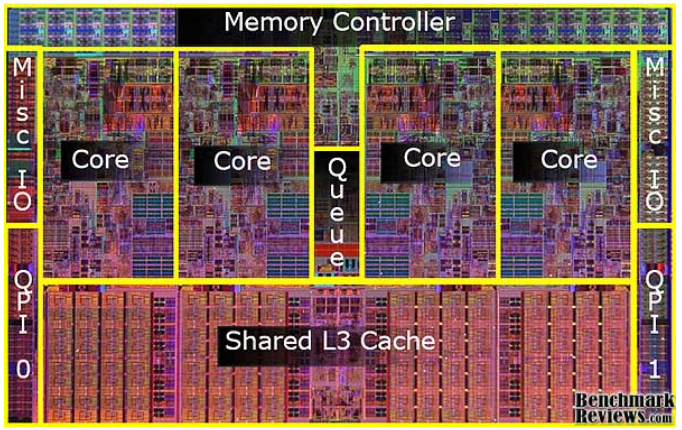


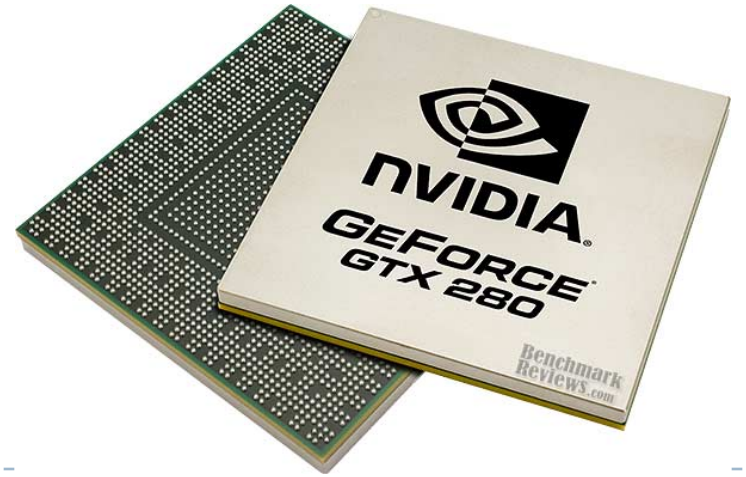




Die Software-Krise von morgen





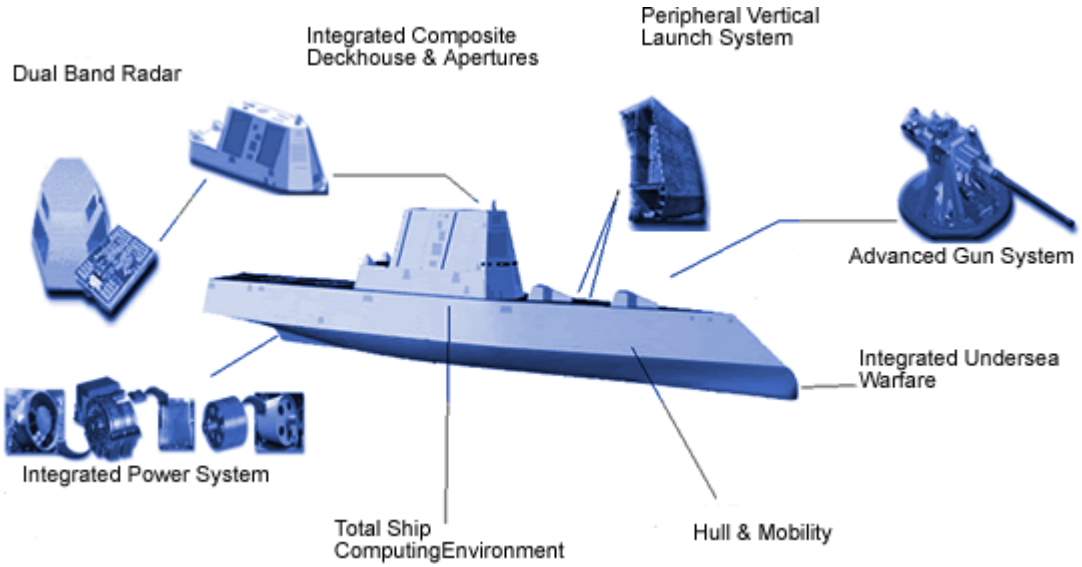


Fehlgeschlagene Softwareprojekte (Beispiele)

- SAGE-System von 1951 zur Aufspürung von Bombern; bei Inbetriebnahme 1963 überflüssig; neue Gefahr Raketen
- Toll Collect: Geplanter Start 31. Aug 2003, tatsächlicher Start 1. Jan 2006; 3.5 Milliarden EUR Einnahmeausfälle
- Kaliforniens Führerscheinstelle stoppt Projekt nach 6 Jahren und 45 M$
- FBI Virtual Case File in 2005 gestoppt, nach 3 Jahren und 170 M$
- Londoner Börse beendet Taurus Projekt 1993 nach 11 Jahren und Budgetüberschreitung um 13200 % (800 M£)

Extreme Komplexität (Beispiel)

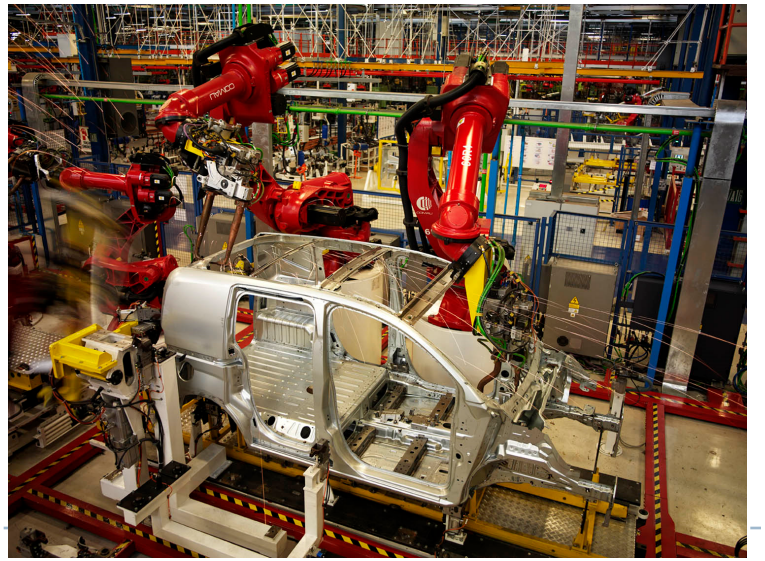
- DDX U-Boot
- Viele eingebettete Systeme
- Zusammen 30.000.000.000 Zeilen Code (Schätzung)
- In 142 Programmiersprachen



Woran liegt es?

- unzureichend spezifizierte Anforderungen
- häufiges Ändern der Anforderungen während des Projekts
- Mangel an Ressourcen
- inkompetente Mitarbeiter
- wenig Benutzer-Einbeziehung
- fehlende Unterstützung durch das Management
- zu große Erwartungen
- falsche Schätzung der Zeit/Kosten
- Managementfehler
- Obsolete Projekte (inzwischen bessere Lösungen)

Engineering?



Der Neid auf die Ingenieure…

- Autoproduktion
  - Kalkulierbare Kosten und Risiken
  - Vorhersehbares Ergebnis
  - Hohe Qualität

• Klare Trennung zwischen Planung und Produktion eines Produkts
• Qualitätskontrolle durch Messungen
• Möglichkeit der Automatisierung, „Industrialisierung“

- Softwareproduktion
  - Kosten können nicht zuverlässig vorhergesagt werden
  - Viele Projekte enden während oder kurz nach der Produktion als Fehlschlag
  - Qualitätssicherungsmaßnahmen garantieren keine quantifizierbare Qualität

Softwaretechnik als Lösungsidee

*Software Engineering: „The Establishment and use of sound **engineering principles** in order to obtain **economically** software that is **reliable** and works **efficiently** on **real** machines."*

[Bauer 1975, S. 524]

- Begriff 1968 geprägt
  - Systematisches Herangehen
  - Publikation von bewährtem Vorgehen und Erfahrung
  - Entwurf, Teile-und-Herrsche
  - Wiederverwendung
  - Qualitätssicherung
- Begriff "Software Engineering" provokativ gewählt

Softwaretechnik in der Informatik

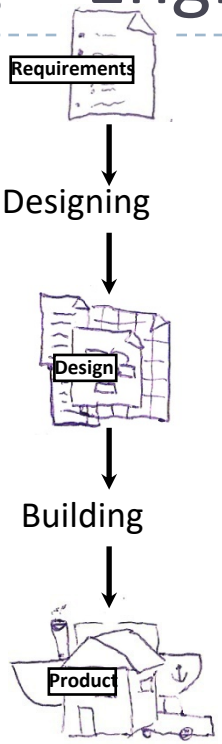
Informatik (computer science)

- Theorien und Methoden für Computer und Softwaresysteme

Softwaretechnik (software engineering)

- Praktische Erstellung von Software

Software Engineering = Engineering?



Ingenieursprodukt

- Produkt ist ein physikalisches Objekt
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist teuer
  - erfordert Arbeit und Material
  - ist langsam
  - teuer zu wiederholen
- Präzise Messung der Qualität

Software

- Das Produkt ist das laufende Programm
- Konstruktion durch Betriebssystem
- Konstruktion
  - ist extrem billig
  - automatisch, kein Materialeinsatz
  - sehr schnell
  - leicht zu wiederholen
- Wenig sinnvolle Metriken zur Messung

Schlussfolgerungen

Software = Design = Plan!
Programmieren ist Entwerfen, nicht Produktion
Simulation von Software ist nicht notwendig.
Agile Techniken sind möglich, “Wasserfallmodell” ist nicht adäquat

Aufbau der Vorlesung

Aufbau der Vorlesung

- 1. Anforderungsanalyse
  - Was will der Kunde?
  - Entwickeln wir das Richtige?
- 2. Vorgehensmodelle
  - Wie plant man das Vorgehen?
  - Wie geht man auf geänderte Anforderungen ein?
  - Zum Beispiel: Wann und wie Testen?
- 3. Teamwork und Projektmanagement
  - Versionsverwaltung: Technische Zusammenarbeit
  - Projektmanagement: Arbeitsteilung, Zeitplanung
  - Risikomanagement

Aufbau der Vorlesung II

- 4. Softwaretechnik im Kleinen
  - Wartbarer Quelltext
  - Modellierung, UML
  - Entwurfsmuster, Trennung von Belangen
  - Refactoring
- 5. Softwaretechnik im Großen
  - Systementwurf, Modellierung
  - Modularität
- 6. Qualitätssicherung
  - Metriken
  - Testen
  - Jenseits von Testen

Literatur

- Viele Softwaretechnikbücher auf dem Markt, z.B.
  - *Software Engineering*. Ian Sommerville. Addison-Wesley Pub Co
- Keine konkrete Empfehlung für die gesamte Vorlesung
  - Zu jedem Abschnitt weiterführende Literatur