

**Methodische und Praktische
Grundlagen der Informatik (MPGI 3)
WS 2008/09**

Softwaretechnik

Steffen Helke

Andreas Mertgen (Organisation)

Rojahn Ahmadi, Georgy Dobrev, Daniel Gómez Esperón,
Simon Rauterberg, Jennifer Ulrich (Tutorien)

Veranstaltungshinweis

Studierendenversammlung

Mittwoch, 22.10.08, 14:15
im H 105 (Audimax)

Einladung des Studiendekans
Herr Heiß
für alle Studiengänge der Fakultät

Was machen wir heute?

- Organisatorisches
- Thematischer Überblick
- Motivation: **Beispiele für Softwarefehler**
- Versuch: Definition von **Softwaretechnik**

Organisatorisches: Übung / Tutorien

- **Reservierte Termine**

Montags und Dienstags von 12-18 Uhr

→ **Erste Tutorien: 27./28.10. 2008**

- **Anmeldung und Info übers Web-Portal**

<http://www.isis.tu-berlin.de/course/view.php?id=1232>

Anmeldung ist bis zum 23.10.2008 möglich!

Inhalte der Übungsaufgaben

6 Übungsblätter zu den Themen

- Objektorientierte Analyse und Design mit UML nach dem Rational Unified Process (RUP)
- Java-Implementierung für eine Teilfunktionalität aus dem UML-Entwurf
- Spezifikation von Vor- und Nachbedingungen mit der Spezifikationssprache Object-Z
- Modellierung reaktiver Systeme mit Statecharts

Organisatorisches: Übungsaufgaben

- Bearbeitung **aller** Aufgaben ist erforderlich!
- Notwendig zur Klausurvorbereitung
- Gruppenarbeit: voraussichtlich 4 Personen
- Pünktliche Abgabe beim Tutor

Verspätete Abgaben können nicht berücksichtigt werden!

Organisatorisches: Klausuren

Voraussetzung:

- erfolgreiche Bearbeitung der Übungsaufgaben

Es sind **zwei** Klausuren zu schreiben!

- Klausur A: Analyse und Design mit UML
- Klausur B: Object-Z und Statecharts

1. Termin **19.02.2009**

- Beginn Klausur A: 8.30 Uhr
- Beginn Klausur B: 10.00 Uhr

2. Termin **09.04.2009** (normal zur Wiederholung)

Organisatorisches: Gesamtnote

Prüfungsäquivalente Studienleistung

- Übungsnote (50%)
- Klausurnoten (2*25%)

Hinweis:

Wer ohne triftigen Grund an beiden Klausurterminen fehlt, hat das Modul nicht bestanden!

Wiederholung von Prüfungsleistungen

Zweiter Versuch:

- erst im folgenden Jahr (WS 09/10)
- Wiederholung **aller** Teilleistungen erforderlich!

Dritter Versuch:

- mündlich, beim Professor

Ausnahme:

Wer am 19.02.09 die Klausur A oder B schreibt, darf diese bereits am 09.04.09 wiederholen (Übung wird angerechnet).

Diverse Prüfungsmodalitäten

- **Bachelor-Studiengänge**

- Informatik (Softwartechnik + Praktikum → **MPGI3**)
- Technische Informatik (Softwaretechnik → **MPGI3TI**)

- **Diplom-Studiengänge**

- Informatik
- Technische Informatik

- **Service-Teilnehmer**

- Techno- und Wirtschaftsmathematik (TWM)
- Wirtschaftsingenieure (Wi-Ing)
- Informationstechnik im Maschinenwesen (ITM)
- Sonstige (Mathe, E-Technik usw.)

Buchempfehlungen

- Sommerville: Software Engineering. Addison-Wesley. 2007.
- Balzert; Lehrbuch der Softwaretechnik. Spektrum. 2001.
- Seemann, Guldenberg: Software Entwurf mit UML2. 2006.
- Brügge, Dutoit: Objektorientierte Softwaretechnik mit UML, Entwurfsmuster und Java. 2004.
- Kruchten: The Rational Unified Process: An Introduction. Third Edition. Addison-Wesley. 2003.
- diverse Bücher über UML2, Object-Z und Statecharts

Unterlagen von unserem Lehrstuhl

→ **Besuch der Vorlesung wichtig!**

- kein explizites Skript
- Folien aus der Vorlesung
- Beispiele aus den Tutorien
- methodische Anleitungen in skriptähnlicher Form

Themen der Vorlesung

- Einführung in die Softwaretechnik
- Prozessmodelle (z.B. Rational Unified Process)
- Textuelle Anforderungsanalyse
- Objektorientierte Analyse und Entwurf mit UML
- Übergang zur Implementierung mit Java
- Architekturmodelle, Entwurfsmuster
- Formale Modellierung mit Object-Z,
- Modellierung reaktiver Systeme mit Statecharts

Softwaretechnik?

„ brauchen wir nicht “

Wir können **programmieren**,

Wir haben einen **Internetanschluß**

→ Wir können eine Softwarefirma gründen

und damit reich werden 😊

... oder?

Vorbemerkungen

Was geht schief? (nach J. Ludewig)

Menschen schaffen erstaunliche Artefakte

- Pyramiden Ägyptens
- Gotische Kathedralen
- Relativitätstheorie
- Verkehrsflugzeuge
- Platos Philosophie



Leider ist der Mensch **dumm**.

- Dummheit nicht als Gegensatz zur Intelligenz
- Sondern als Lücke zwischen Möglichkeiten und Kompetenz

Wer die Möglichkeiten ausschöpft und dabei seine Kompetenz überschreitet, ist **dumm**.

Wer sich innerhalb seiner Kompetenz bewegt, ist **weise**.

So haben die Menschen die Möglichkeit

- gigantische Verkehrsmittel zu bauen
- riesige Staaten und große Verwaltungen zu organisieren
- immer gewaltigere Konzerne zu schaffen
- Rechner durch ein umspannendes Netz zu verbinden

Oft reicht die Kompetenz aber nicht aus. Darum

- sind Titanic und Sowjetunion untergegangen
- trauern Daimler-Chrysler-Aktionäre um ihr Geld
- beschert uns das Web Viren und Spam
- stürzt Windows ab

**Manchmal wächst Kompetenz später,
leider ist das nicht sicher.**

Fehlschläge in der Softwaretechnik

Ariane 5: Der Unfall des Fluges Ariane 501

- Am 4.6.96 endete der Jungfernflug der Ariane 501 nach 40 Sekunden mit einer **Explosion**.
- Die **Herstellungskosten** für die Rakete und die Fracht (4 Forschungssatelliten) sind **500 Mio US\$**.
- Der Flug war **nicht versichert**.
- Die Entwicklung der Rakete hat 10 Jahre und **7 Milliarden US\$** beansprucht.
- Die eingesetzte Untersuchungskommission legte am 19.7.96 einen Bericht vor.

Ariane 5: Der Unfall des Fluges Ariane 501



Rückläufige Ereignisabfolge

- 39s nach Start lösen sich die Booster durch hohen aerodynamischen Druck ab.
 - **Selbsterstörung wird eingeleitet!**

Die eigenen Düsen lösten diesen Druck aus.
- Die Düsen wurden vom On-Board-Computer (OBC) gesteuert. Die Steuerung basierte auf Daten des Inertial Reference Systems (IRS).
- Das IRS enthielt **ungültige Flugdaten**. Das Backup-IRS produzierte auch ungültige Daten.

Overflow Run Time Error

Ursache: Eine Exception wurde ausgelöst bei der
Konvertierung einer 64-bit floating point Variable
in einen 16-bit signed Integerwert.

```
begin
  sensor_get(vertical_veloc_sensor);
  sensor_get(horizontal_veloc_sensor);
  vertical_veloc_bias := integer(vertical_veloc_sensor);
  horizontal_veloc_bias := integer(horizontal_veloc_sensor);
exception
  when numeric_error => calculate_vertical_veloc();
  when others        => use_irs1();
```

Unterschiede zwischen Ariane 4 und 5

- Die Flugbahn der Ariane 5 weicht von der Flugbahn der Ariane 4 ab → **Horizontalgeschwindigkeit** verschieden.
- Deshalb enthielt die Variable einen Wert für die Horizontalgeschwindigkeit außerhalb ihres gültigen Wertebereichs.
- Der Fehler wurde nicht abgefangen und sondern ans **Zweitsystem übergeben**.
- Der Teil der Software, in dem der Fehler geschah, sollte bei der Ariane 5 nach dem Start nicht mehr arbeiten.

Welche Ursachen führten zum Unglück?

1. Codefehler führt zu Laufzeitfehler

- Zu großer Wert der Variablen → Speicherüberlauf
- Design des IRS wurde **von der Ariane 4 übernommen.**

2. Implementierungsentscheidung / Optimierung

- IRS-Computer sollte auf eine Arbeitslast von max. 80% eingestellt werden.
- Drei ungeschützte Variablen wurden statisch auf Fehlermöglichkeiten überprüft
- Fehlerbehandlung wurde für überflüssig befunden (**Analyse benutzte Flugdaten der Ariane 4**)

Weitere Ursachen

3. Änderung der Umgebung

- Programm war nach dem Start nicht notwendig, lief aber noch 40 Sekunden weiter
- Grundsatz: **ein bisher funktionierendes System nicht ändern!**

4. Globale Kontrollstruktur

- Das **IRS schaltete sich nach einer Exception ab** und vertraute auf Redundanz.
- Die ungültigen Daten wurden vom OBC ausgewertet.
- Die **Redundanz der Systeme schützte nicht!**

Aber es wurde doch getestet ...

5. Unzureichende Tests

- Das IRS und das Restsystem wurden getrennt getestet.

6. Spezifikationsänderung: Flugbahn

- Die Spezifikation für die IRS umfasste nicht die realen Flugbahndaten.

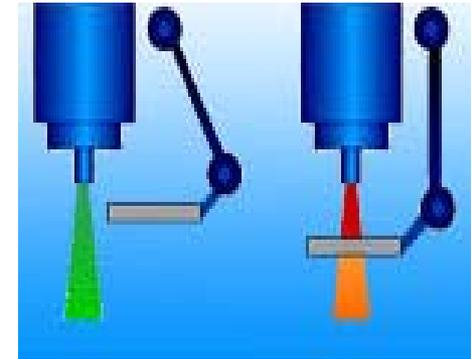
Empfehlungen der Kommission

- Es sollte **keine Software**-Funktion während des Fluges laufen, **die nicht erforderlich ist.**
- Testen sollte unter realistischen Bedingungen stattfinden
→ komplette Simulationen vor jeder Mission
- Spezifische Software-Qualifikations-Berichte für jedes Ausrüstungseinzelteil einführen
- Externe Experten bei den Überprüfungen einbeziehen
- Flugbahndaten in die Spezifikationen und Testanforderungen einfügen

**Es gibt unzählige andere fatale
Softwarefehler in der Geschichte
der Informatik.**

Therac-25-Beschleuniger (1985 - 87)

- Geräte zur Strahlentherapie
- Bestrahlung mit Beta- und Gamma-Strahlen
- Erzeugung von Gamma-Strahlen durch Elektronenbeschuss eines Metallblocks
- Durch einen Softwarefehler ist Bestrahlung ohne Metallblock möglich.



**Softwarefehler kostet fünf
Menschen das Leben!**

Netzausfall bei AT&T (1990)

- 15.1.1990 keine Vermittlung von 70 Millionen Ferngesprächen (9h Ausfall)
- Schaden bei AT&T ca. **75 Millionen US\$**
- Folgeschäden bei Kunden mehrere 100 Millionen US\$

Ursache: falsch gesetzter *break*-Befehl in neuer Software

- Aufspielen von älterer Software löste Problem

U-Bahn in London

- Am 10.4.1990 hatte der Fahrer einer U-Bahn Probleme.
- Er hatte den Startknopf bereits gedrückt.
- Zug verweigerte Abfahrt, da Tür klemmte.
- Fahrer stieg aus und behob das Problem.



Der Zug fuhr ohne Fahrer los!

Quelle: Neumann, *Computer-Related Risks*, ACM Press, 1995.

Spektakulärer Softwarefehler im TV



Warum ist es so schwierig, korrekte Software zu entwickeln?

Konflikte eines Softwareentwicklers

- Wie soll ich gleichzeitig Software wiederverwenden und weiterentwickeln?
- Wie soll ich Anforderungen festlegen und trotzdem auf geänderte Rahmenbedingungen reagieren?
- Wie können mehrere Leute getrennt entwickeln und trotzdem den Überblick behalten?

Konflikte eines Softwaretesters

- Ergeben viele korrekte Einzelteile wirklich ein korrektes Gesamtsystem?
- Wie soll ich ein komplettes System (Rakete) vorher testen und analysieren?
- Welche Annahmen über Ausfälle von Hardware und Software sind gerechtfertigt?

23. Februar 1984,
Business Week, 1. Seite

Software, The New Driving Force ...

- Wirtschaftsfaktor
- Verantwortung
- Zuverlässigkeit

Pressman, 1986

- why does it take so long to get programs finished ?
- why are costs so high ?
- why can't we find all errors before we give the software to our customers ?
- why do we have difficulty in measuring progress as software is
being developed

Software Engineering....

... ist die Anwendung von Prinzipien, Fähigkeiten und Kunstfertigkeiten auf den Entwurf und die Erstellung von Programmen und Systemen von Programmen.

Jack Dennis, 1975

... ist eine Technologie, mit der es möglich wird, ökonomisch Software zu erhalten, die zuverlässig ist und effizient auf realen Maschinen arbeitet.

F. L. Bauer, 1969

Softwareentwicklungsprojekte

Jedes Projekt ist einmalig

- Entwicklung muss planbar/überprüfbar sein

4-Variablen-Modell

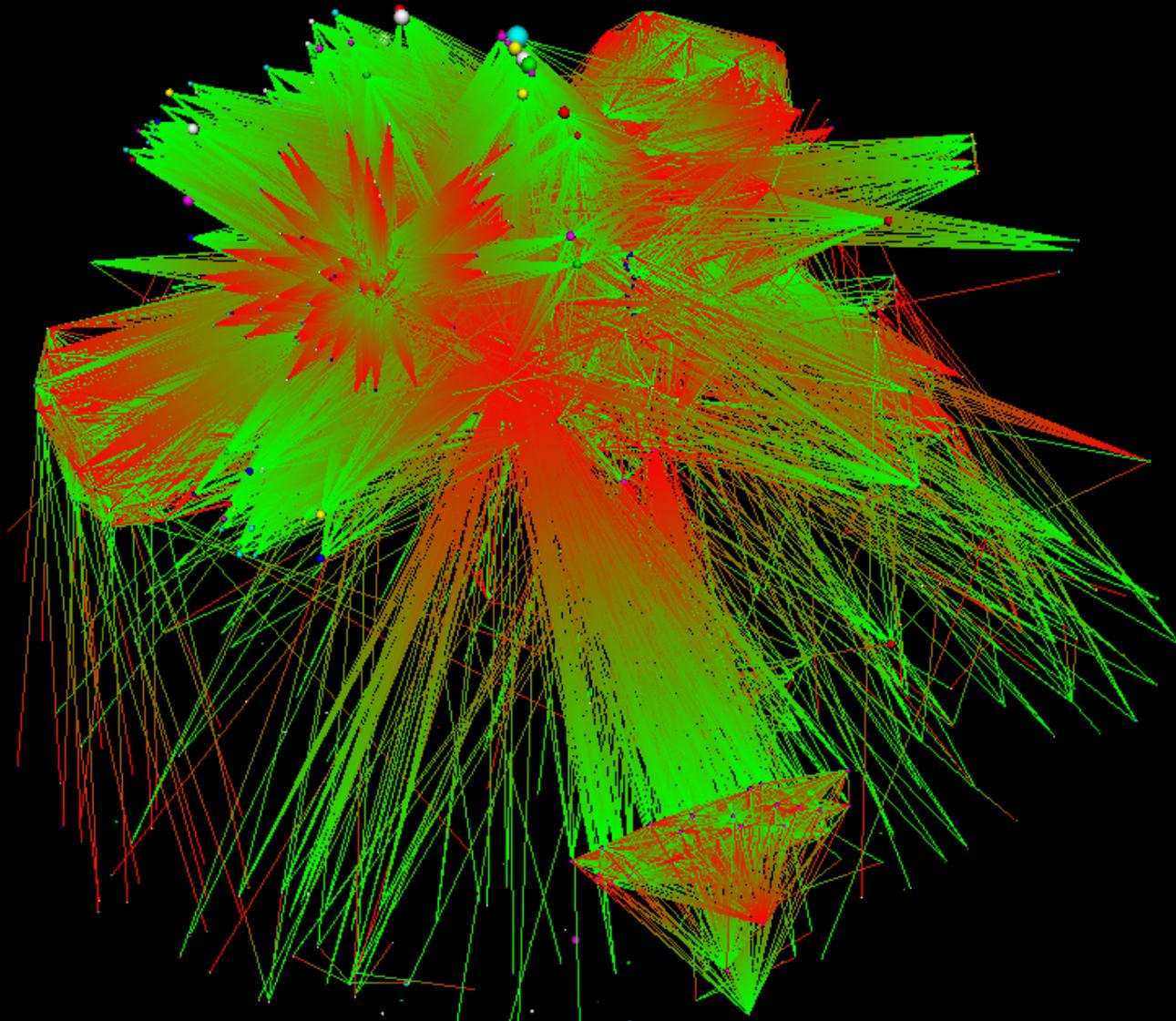
- Einflussfaktoren, die ein Projekt bestimmen:
 1. Umfang
 2. Ressourcen
 3. Qualität
 4. Zeit

[Kent Beck – Extreme Programming]

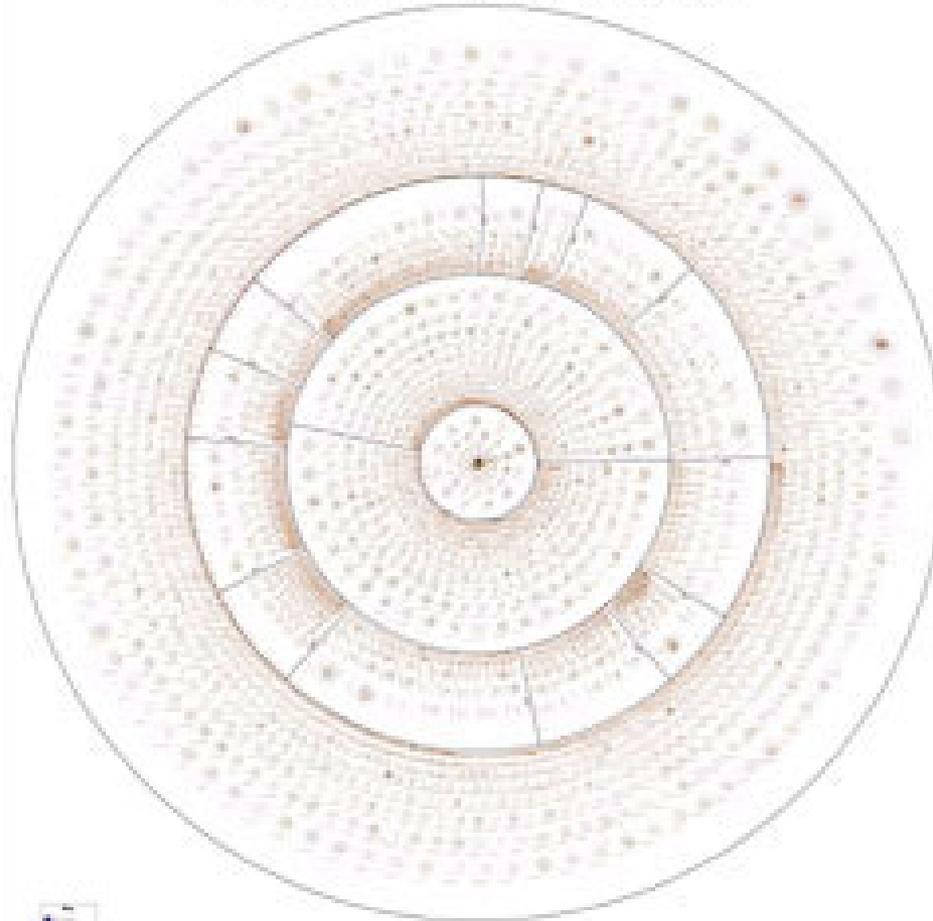
4-Variablen-Modell: Umfang (1)

- Ab wann ist Software „groß“?

Bild eines 220 kLOC Programmes

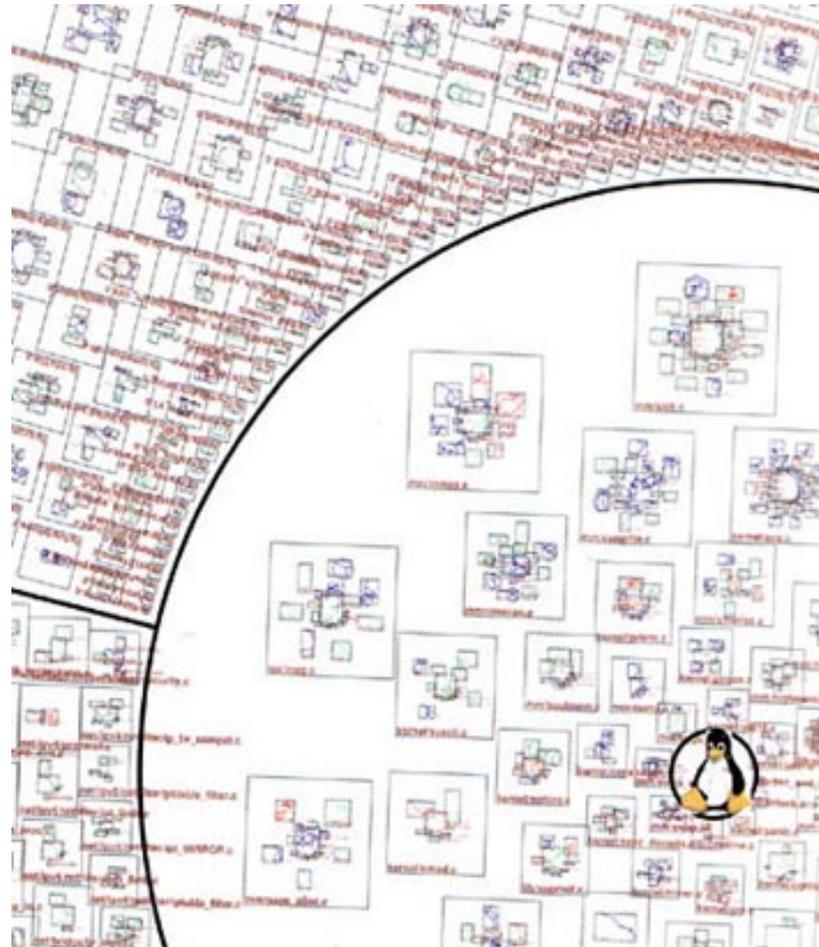


Linux Kernel v2.4.0

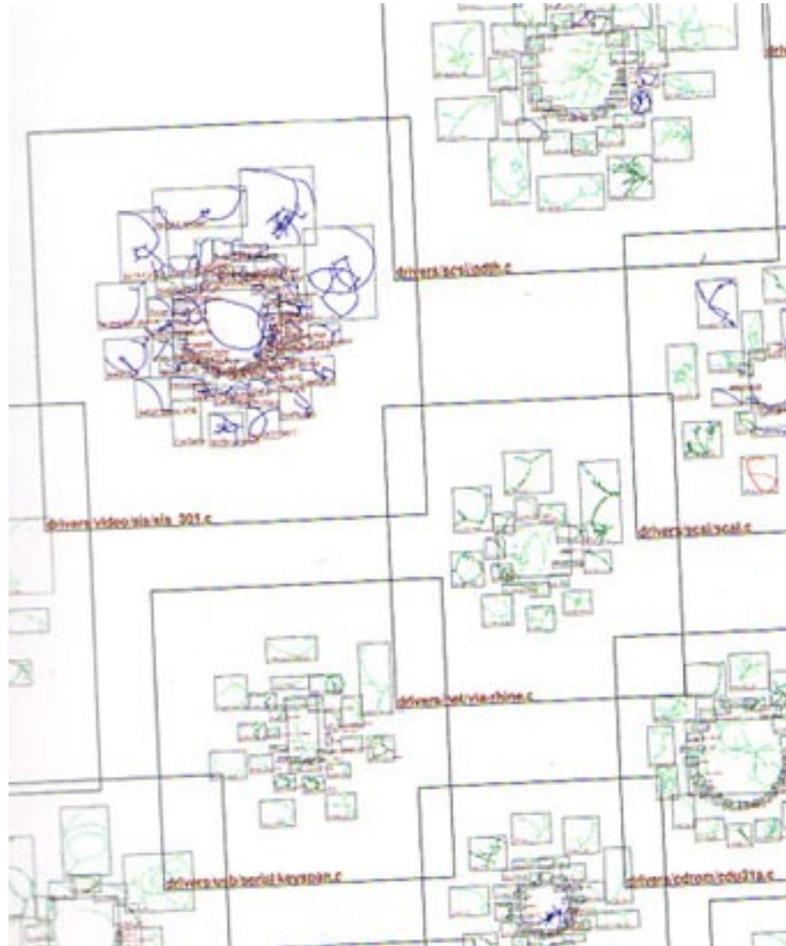


Copyright © 2004 by Red Hat, Inc.

Linux Kernel 2.4 (Ausschnitt)



Linux Kernel 2.4 (Ausschnitt)



4-Variablen-Modell: Umfang (1)

- Ab wann ist Software „groß“?

„I consider a system to be large if it is larger than one mind.“

- Nicht die reine Größe (z.B. LOC) entscheidet, sondern die **Komplexität**

4-Variablen-Modell: Ressourcen (2)

- Große Software wird von großen Entwicklungsteams geschrieben.
- Team-Arbeit erfordert Koordination
 - gleichmäßige Auslastung
 - wie plant man die Auslastung von Menschen?
 - Arbeitsteilung erfordert Kommunikation
 - Kommunikation ist verlustbehaftet
- Projektmanagement!

4-Variablen-Modell: Qualität (3)

- Wie kann man Softwarequalität messen?
- Qualitätmerkmale
 - correctness Korrektheit
 - performance Performanz
 - usability Benutzbarkeit
 - extensibility Erweiterbarkeit
 - ...
- Unterschiedliche Qualitätsinteressen!
- Qualitäten sind nicht additiv

4-Variablen-Modell: Zeit (4)

- **Aufwandsabschätzung?**
- **Zeitdruck durch**
 - Konkurrenz
 - mangelnde Transparenz des Prozesses
 - Pannen im Prozess
- **Wo wird Zeit verloren?**

4 Variablen im Zusammenhang

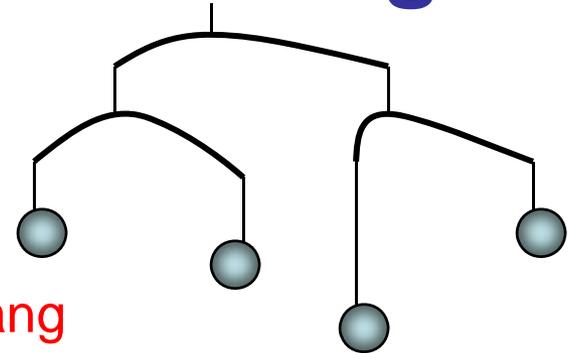
- Abhängigkeiten, z.B.

- Bessere Qualität

- » mehr Zeit oder weniger Umfang

- Mehr Funktionalität

- » mehr Ressourcen oder weniger Qualität ...



- Manche Kombinationen sind (derzeit) unmöglich
 - „Vollständige Korrektheit bis morgen!“
- Nur selten können alle 4 Variablen zu Projektbeginn festgelegt werden
(...und dann später eingehalten werden...)