

Softwarerekonstruktion

Prof. Dr. Falk Howar
Lehrstuhl XIV – Software Engineering
TU Dortmund, Fakultät Informatik

Team

Falk Howar

Simon Dierl

Frederik Gossen

Oxana Warkentin

Martin Fitzke

Lilly Mielke

Kurze Vorstellung

- Professor für Software Engineering am LS14
 - Seit WS2017/2018
- Davor
 - 2014 – 2017: TU Clausthal
 - Enge Zusammenarbeit mit Volkswagen AG
 - 2012-2014: Carnegie Mellon University (SV)
 - Mitglied der „Robust Software Engineering“ Gruppe beim NASA Ames Research Center
 - 2012: Promotion an der TU Dortmund
- Interessen
 - Engineering Methoden für sichere autonome Systeme

Agenda

- Organatorisches
 - Studienordnung: Einordnung / Kompetenzen / Struktur
 - Vorlesung: Bildungsvertrag, Termine, Feedback
 - Übung: Konzept / Termine
 - Klausur
- Vorstellung des Lehrstuhls
- Kurze Einführung: Was ist Software Engineering?
- Vorlesungsinhalte

Struktur laut Modulhandbuch (Bachelor)

- Bachelor Informatik / Angewandte Informatik: **Wahlpflichtmodul.**
- Teilnahmevoraussetzungen:
 - Erfolgreich abgeschlossene Module: „Software-Technik“ (SWT)
 - „Software Praktikum“ ist nicht mehr Teilnahmevoraussetzung
 - Kenntnisse: Objektorientierung, Programmierpraxis, Mathematische Grundlagen der Informatik
- Umfang:
 - 3 SWS (2 SWS Vorlesung, 1 SWS Übung)
 - 4 Credits: 3 Credits Vorlesung, 1 Credit Übung
- Aufwand:
 - 120 Stunden über 15 Semesterwochen
 - 45 Stunden Präsenz ($15*(2+1)$)
 - 75 Stunden Vor-/Nachbereitung und Hausübungen ($15*5$)
- Veranstaltungssprache: Deutsch.

Vorlesung

- Vorlesungstermin:
 - Mo 12:15 bis 13:45,
 - Otto-Hahn-Str. 14 – E23
- Webseite(n):
 - <https://ls14-www.cs.tu-dortmund.de/cms/de/Lehre/Lehrveranstaltungen/2018WS/SWK/index.html>
 - <https://moodle.tu-dortmund.de/course/view.php?id=12923>

Aktuelle Informationen zur Vorlesung



Folien etc. bei Moodle (anmelden mit Unimail-Account) und in Kurs
“Softwarekonstruktion WS18/19” einschreiben.

Vorlesung (II)

- Folien:
 - Vorlesungsfolien werden im Moodle zur Verfügung gestellt
 - Planmäßig spätestens 18:00 Uhr am Vortag der Vorlesung.
- Material zur Klausurvorbereitung:
 - Folien, teilw. Notizfolien und Anhänge
 - Fragen zur Selbstkontrolle
 - Übungszettel
 - Probeklausur mit Musterlösung

Literatur

- Liste online im Laufe der Woche
- Wo möglich: online verfügbar

Prüfung (Studienordnung)

- Bachelor Informatik / Angewandte Informatik:
 - Studienleistung: **Übungsschein**
 - Modulprüfung: **Klausur** (bei bestandener Studienleistung)
- Diplom:
 - Prüfungsleistung: **Klausur**

Klausur

- Prüfung: Klausur
 - schriftlich
 - 90 Minuten
- Klausurtermine:
 - Erster Prüfungstermin: 06.02.2019
 - Zweiter Prüfungstermin: 14.03.2019

Übungsschein

- Bachelor: Bearbeitung der Übungsaufgaben ist Voraussetzung zur Teilnahme an der Modulprüfung!
- 6 Zettel.
 - Min. 50% aus allen Zetteln.

Übungen (Formalia)

- Ablauf der Übungen und Erwerb der Studienleistung
 - siehe Webseite zur Vorlesung und Hinweise auf 1. Übungsblatt
- Übungsblatt online Montags nach Vorlesung
 - (ab 15.10., 14 täglich)
- Übungstermine zum Vorstellen der Lösungen
 - Zettel zurück in Übungen
 - Keine Ausgabe von Musterlösungen
- Bearbeitung und Abgabe in Gruppen bis Größe 4

Übungen (Abgabe)

- Abgabe: immer **Donnerstags 12h** in der Woche nach Ausgabe
 - Besonderheit: Erstes Blatt (siehe nächste Folie) !!!!
 - Termin steht auf dem Blatt
 - Briefkasten EG OH14 / 1.OG OH12
- Bei Unklarheiten: Nachfragen!

Übungen (Termine)

- Termine (14-tägig)
- Ausgabe 1. Übungsblatt:
15.10.2016 nach der Vorlesung
- Ungerade Gruppen: KW44,
KW46, KW48, KW50, KW2,
KW4
- Gerade Gruppen: KW45,
KW47, KW49, KW51, KW3,
KW5

Bei Bedarf max. 4 weitere Gruppen
(2 weitere Termine)

Nummer	Zeit	Raum
1	Dienstag 10:15 – 11:45	OH12 / 1.056
2	Dienstag 10:15 – 11:45	OH12 / 1.056
3	Dienstag 10:15 – 11:45	OH12 / 3.031
4	Dienstag 10:15 – 11:45	OH12 / 3.031
5	Dienstag 16:15 – 17:45	OH12 / 2.063
6	Dienstag 16:15 – 17:45	OH12 / 2.063
7	Dienstag 16:15 – 17:45	OH16 / 205
8	Dienstag 16:15 – 17:45	OH16 / 205
9	Mittwoch 14:15 – 15:45	OH12 / 2.063
10	Mittwoch 14:15 – 15:45	OH12 / 2.063
11	Mittwoch 14:15 – 15:45	OH12 / 3.031
12	Mittwoch 14:15 – 15:45	OH12 / 3.031
13	Mittwoch 16:15 – 17:45	OH12 / 2.063
14	Mittwoch 16:15 – 17:45	OH12 / 2.063
15	Mittwoch 16:15 – 17:45	OH12 / 3.031
16	Mittwoch 16:15 – 17:45	OH12 / 3.031

Übungen (Anmeldung)

- Anmeldung:
 - Via AsSESS
 - <http://ess.cs.uni-dortmund.de/ASSESS/index.php?do=exerciselist&lectureid=316>
- Platzvergabe:
 - Anmeldung möglich ab 14:00 Uhr.
 - Anmeldung bis 15.10.2018, 14:00 Uhr.
 - Verteilung mittels Algorithmus (Solver).
(nicht priorisiert nach zeitlichem Eingang der Anmeldung)
 - Bekanntgabe der Verteilung am 17.10.2018.
- Spätere Anmeldungen: *first come first served*

Moodle-Forum

- Diskussion der Studierenden untereinander
- Zusätzliche Kommunikation mit Veranstaltern für Fragen zum Inhalt, aber:
 - Keine garantierten Antwortzeiten
 - Für Dringendes oder Vertrauliches: Sprechstunde oder Email
- Organisatorische + inhaltliche FAQ
 - Für Fragen von Studierenden, die auch für andere interessant sein könnten
 - Moderation durch Veranstalter

Zusammenfassung: Kontakt

- Webseiten (**immer erste Inofrmationsquelle!!!**)
 - <https://ls14-www.cs.tu-dortmund.de/cms/de/Lehre/Lehrveranstaltungen/2018WS/SWK/index.html>
 - <https://moodle.tu-dortmund.de/course/view.php?id=12923>
- Ansprechpartner (verlinkt von o.g. Webseite)
- Vorlesung:
 - Falk Howar
- Übung:
 - Simon Dierl
 - Frederik Gossen
- Übungsanmeldung:
- <http://ess.cs.uni-dortmund.de/ASSESS/index.php?do=exerciselist&lectureid=316>

„Bildungsvertrag“

Wir bieten:

- Fachliche Einführung in das Thema Softwarekonstruktion
- Engagierte Betreuung
- Interessante Vorlesung
- Regelmäßige Sprechstunden
- Betreute Übungen
- Korrigierte Hausübungen
- Transparente Anforderungen
- Möglichkeiten zum direkten Feedback
- Möglichkeit zum Erwerb des Scheins

Wir erwarten:

- Aktive Teilnahme an der Vorlesung
- Vor- und Nachbereitung der Vorlesung
- Aktive Teilnahme an den Übungen
- Bearbeitung der Hausübungen

Feedback

Ich bitte um vorlesungsbegleitendes Feedback, um Verbesserungen semesterbegleitend durchführen zu können.

- Übliche Kontaktmöglichkeiten:
 - Nach der Vorlesung
 - Sprechstunde: Mo 14-15 Uhr (bitte vorher per Email anmelden)
 - Email
 - Tel.: 0231 755-7945

Umfrage im Moodle!

Softwarekonstruktion

Anknüpfungspunkte

- Was Sie aus „Softwaretechnik“ und ggf. dem „Software-Praktikum“ wissen sollten:
 - Qualitätsmerkmale von Software
 - V-Modell
 - Testverfahren auf den verschiedenen Ebenen
 - Programmtest, Klassentest, Komponententest, ...
 - UML-Diagramme
 - Klassendiagramme
 - Aktivitätsdiagramme

Lernziele

Erlangbare Kompetenzen innerhalb der Vorlesung:

- Überblick Software Engineering
- Basiswissen Anforderungen
- Modellbasiertes Software-Engineering
- Software Architektur
- Überblick über das Spektrum gängiger Konzepte zur Spezifikation und zum Testen
- Einbettung Spezifikationskonzepte in die Qualitätssicherung
- Fortgeschrittene Spezifikations- und Verifikationstechniken (Statische Analyse, Symbolische Analyse)
- Basiswissen Software Management (Qualität, Projekt Management, Konfiguration)

Agenda

- Organisatorisches
- Vorstellung des Lehrstuhls
- Kurze Einführung: Was ist Software Engineering
- Vorlesungsinhalte

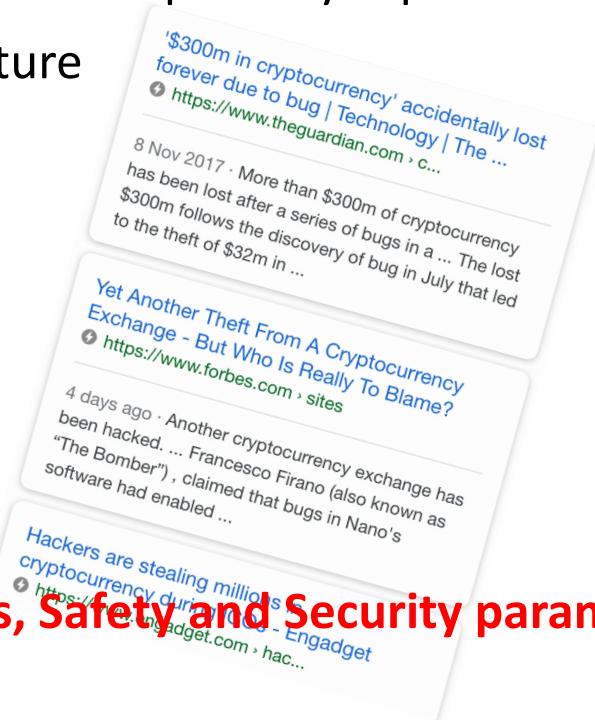
Lehrstuhl für Software Engineering

- Vier Gruppen

- Software Engineering by Algorithms and Logic (Prof. Dr. Jakob Rehof)
- Virtual Machining (Prof. Dr.-Ing. Petra Wiederkehr)
- RSE für autonome Systeme (Prof. Dr. Falk Howar)
- Datenanalyse in Anwendungsbereichen Biologie und Materialwissenschaften (Dr. Lars Hildebrand)

Digital Infrastructure of the 21st century

Autonomous and intelligent systems are envisioned to disrupt many aspects of our lives in the near future



'\$300m in cryptocurrency' accidentally lost forever due to bug | Technology | The ...
<https://www.theguardian.com> › c...

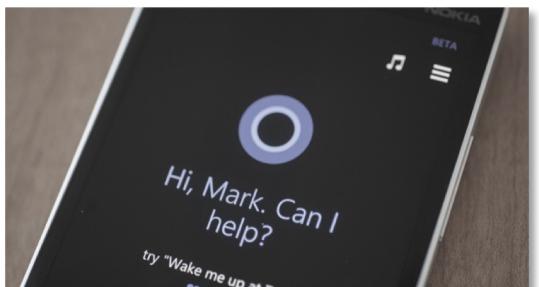
8 Nov 2017 · More than \$300m of cryptocurrency has been lost after a series of bugs in a ... The lost \$300m follows the discovery of bug in July that led to the theft of \$32m in ...

Yet Another Theft From A Cryptocurrency Exchange - But Who Is Really To Blame?
<https://www.forbes.com> › sites

4 days ago · Another cryptocurrency exchange has been hacked. ... Francesco Firano (also known as "The Bomber"), claimed that bugs in Nano's software had enabled ...

Hackers are stealing millions of cryptocurrencies during ICOs - Engadget
<https://www.engadget.com> › hac...

Correctness, Safety and Security paramount!



New Notion of Safety

State of the Art



Applications with limited scope/risk
(delivery robots in hospitals, warehouses)

Or constant human oversight

Safety: Controllability

Next Generation



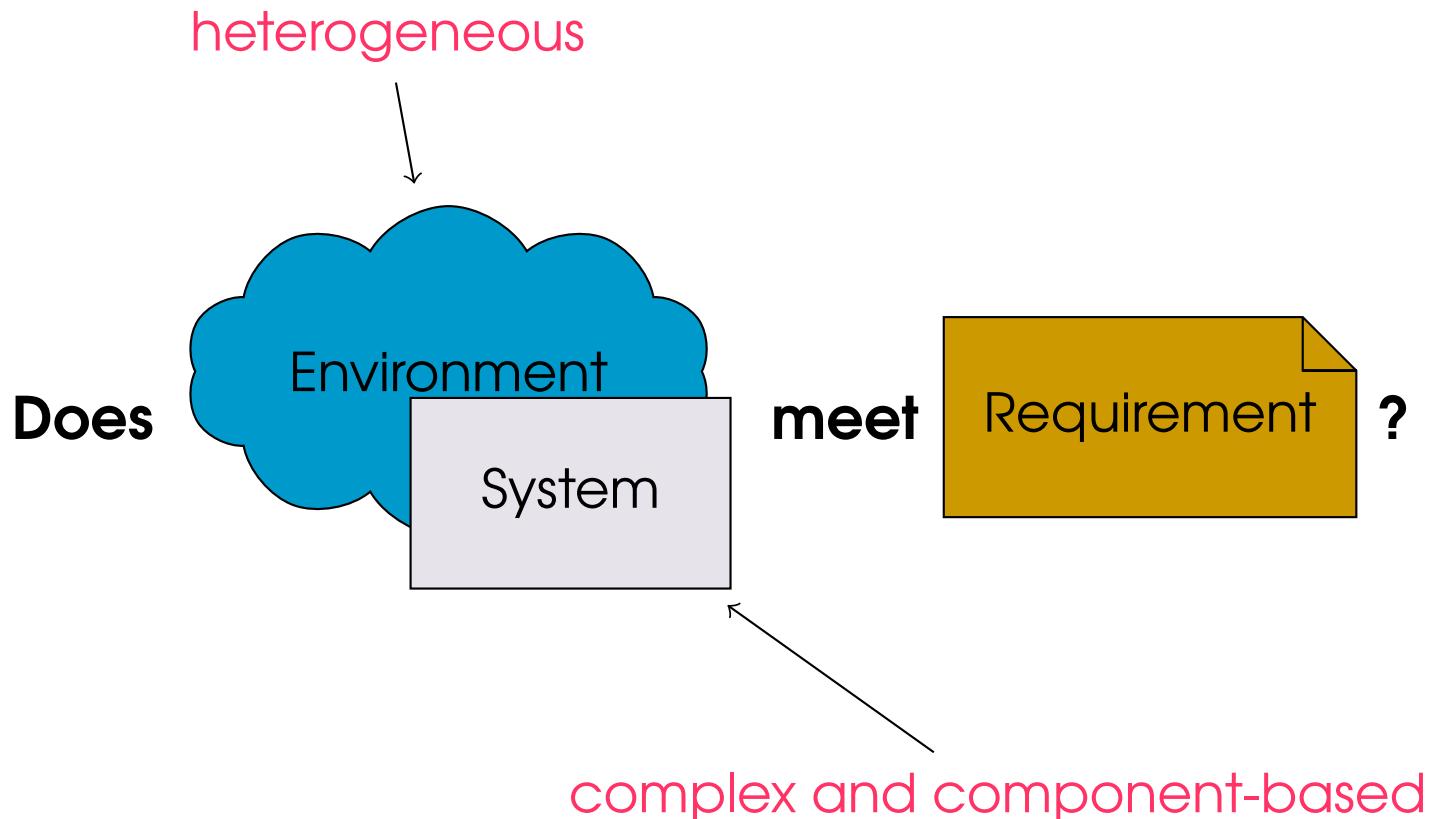
- Operation in less controlled environments
- Tasks with greater autonomy / risks

Safety: Safe behavior in all scenarios

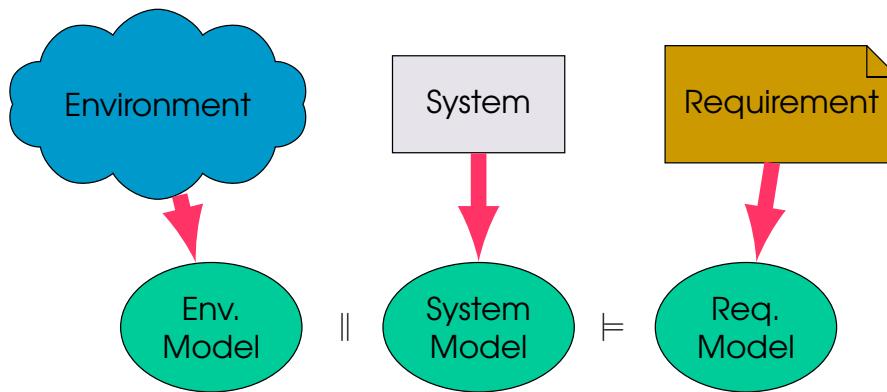


Fundamentally
Different!

Konkrete Domänen / Fragestellungen



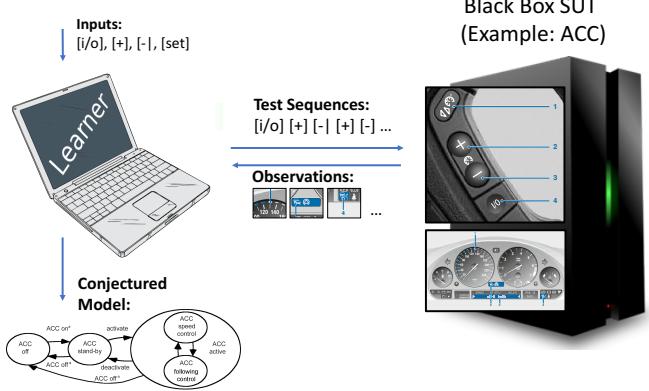
Approach



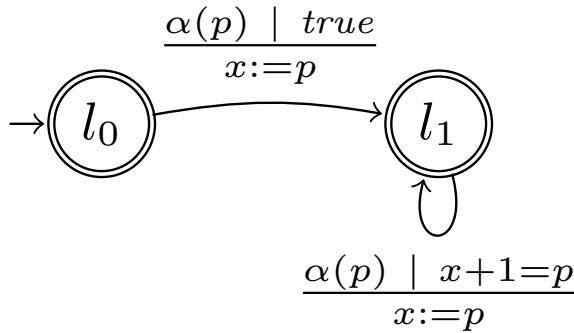
- (1) How can we generate models of system, environment, and properties that are adequate for formal verification?
- (2) How can we account for the imprecision of generated models when checking safety?

Example 1: Learning-based Testing of TCP

Active Automata Learning



Register Automata



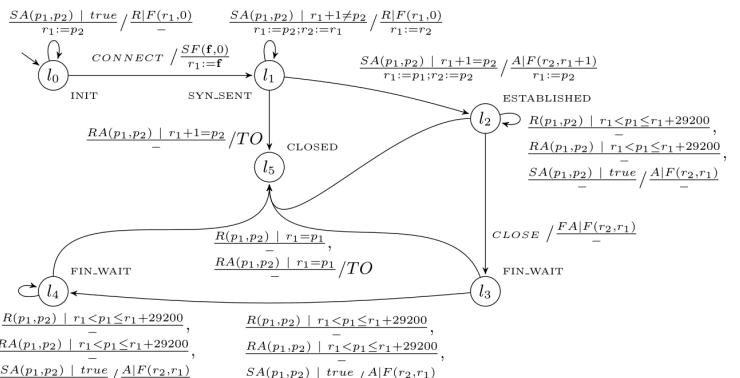
Protocol Conformance

- protocols: SSH, TLS, SMTP, FTP, TCP, UDP...
- many implementations per protocol
 - implementations MUST/SHOULD/MAY **adhere** to the specifications...

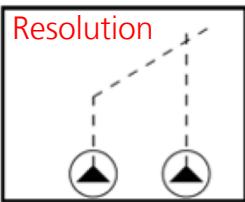
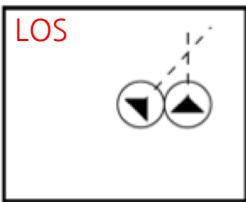
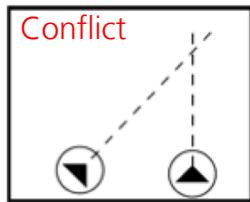
} conformance testing



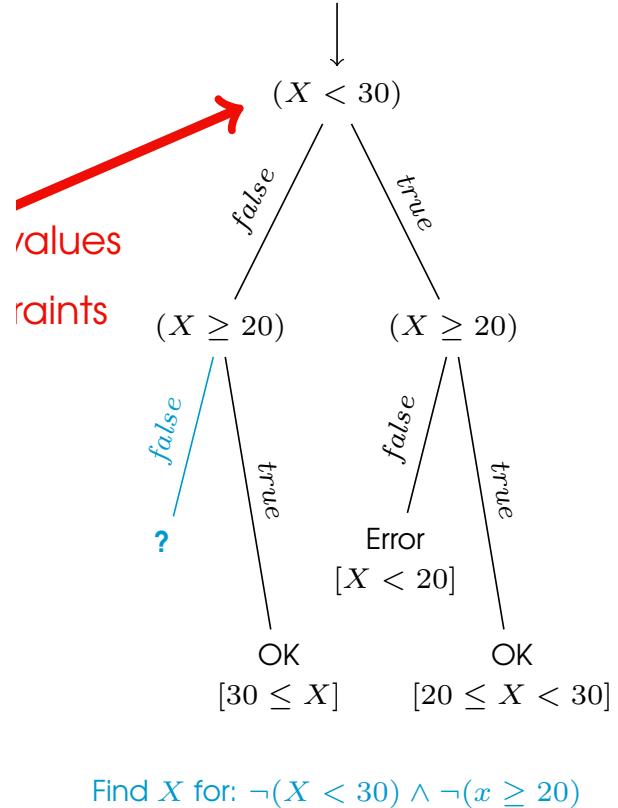
Testing TCP



Example 2: Symbolic Testing of AutoResolver



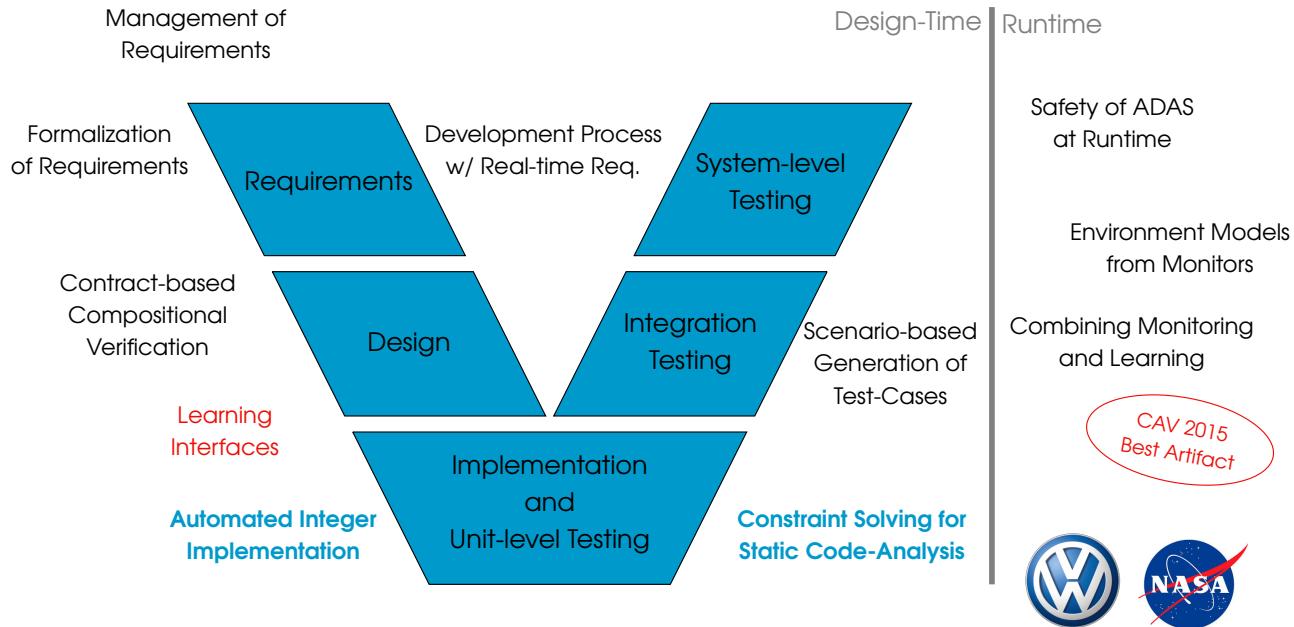
- AutoResolver prototype developed by NASA
- My project developed a **scenario-based** approach for **testing AutoResolver in simulations**
- Simulation environment provided by NASA



Symbolic Execution: J.C. King, 1976

<https://github.com/psycopaths/jdart>

More Past Projects



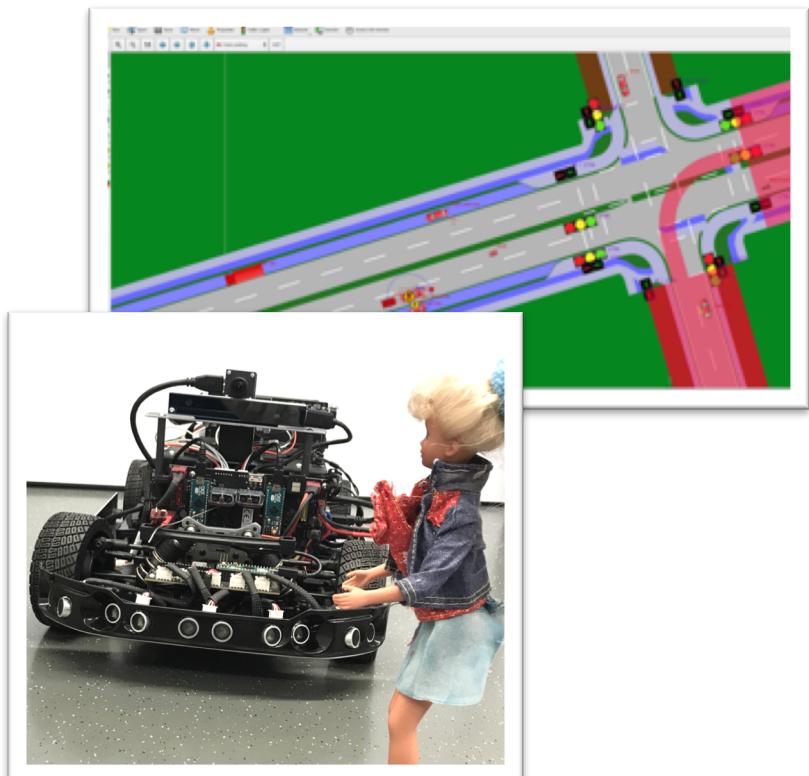
Research Interests: Methods for analyzing and assuring behavior of software systems

Approach:

1. Apply **learning** and **formal methods** research in industrial collaborations
2. Ground research in questions derived from industrial collaborations

Ausstattung

Labor I: Sichere Autonome Fahrzeuge



Labor II: Lernende Verfahren und Formale Methoden im Software Engineering

Rechnerausstattung zum Erproben
von Automatenlernen und Machine
Learning

- jonas
- shaw
- andrews (2x GTX 1080)

Abschlussarbeiten

- Themen in den Bereichen:
 - Symbolische Ausführung
 - Automatenlernen
 - Absicherung sicherheitskritischer Systeme
- Bei Interesse bitte bei mir melden
- Vorlaufzeit: Aktuell 3-4 Monate

Hiwi Tätigkeit

- Wir suchen immer nach engagierten Studentischen Hilfskräften zur:
 - Unterstützung der Forschung im Lehrstuhl 14 – Software Engineering
 - Graduiertenkolleg 2193
 - Nationales Leistungszentrum: „Logistik und IT“
 - Unterstützung von Lehrveranstaltungen am LS14
 - Projektmitarbeit im Fraunhofer Institut für Software- und Systemtechnik (ISST)
- Bei Interesse melden Sie sich bitte bei mir!

Agenda

- Organisatorisches
- Vorstellung des Lehrstuhls
- Kurze Einführung: Was ist Software Engineering
- Vorlesungsinhalte

Software Engineering

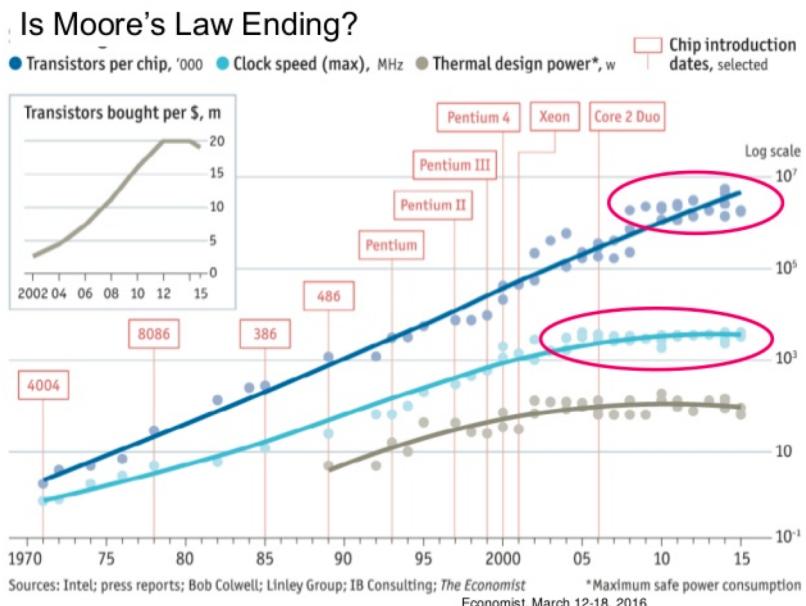
Software Engineering is...

... the establishment and use of **sound engineering principles** in order to obtain **economically** software that is **reliable** and **works efficiently** on real machines. [NR68]

?

... the **systematic approach** to the development, operation, maintenance, and **retirement** of software.“ [IEEE83]

Geschichte (I)



- Erste programmierbare Computer Ende 1940er Jahren
- Erste Programmiersprachen in den 1950er Jahren
- Ab den 1960ern wird Hardware immer günstiger und leistungsfähiger
- Software Projekte werden größer

Software Krise (I)

Ende der 1960er Jahre

- Software Projekte laufen unbefriedigend ab
- Projekte sprengen Budgets
- Deadlines werden nicht eingehalten
- Software hat Fehler und ist langsam
- Projekte werden abgebrochen

Software Krise (II)

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

*Edsger Dijkstra, The Humble Programmer (EWD340),
Communications of the ACM*

Geschichte (II)

- Die Idee einer Disziplin 'Software Engineering' entsteht bei einer NATO Konferenz 1968 / 1969 in einer Diskussion der Software Krise

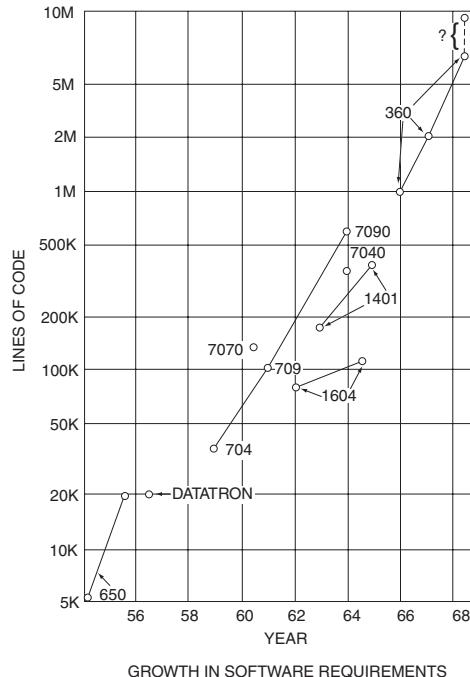
»Production of large software has become a scare item for management. By reputation it is often an unprofitable morass, costly and unending.

...

No less a person than T. J. Watson said that OS/360 cost IBM over 50 million dollars a year during its preparation, and at least 5000 man-years' investment.

...

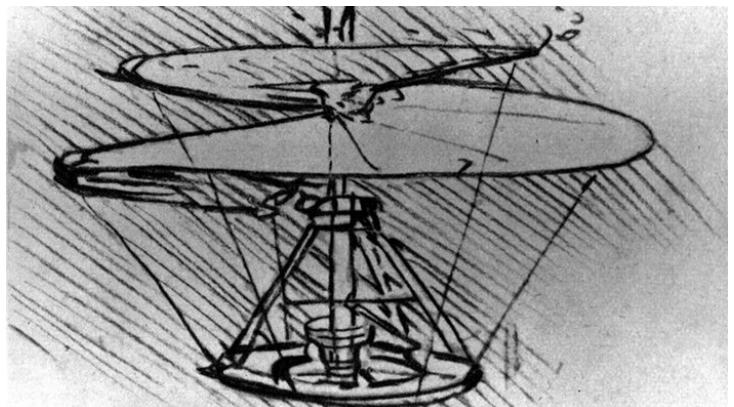
The commitment to many software projects has been withdrawn. This is indeed a frightening picture.«



<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>

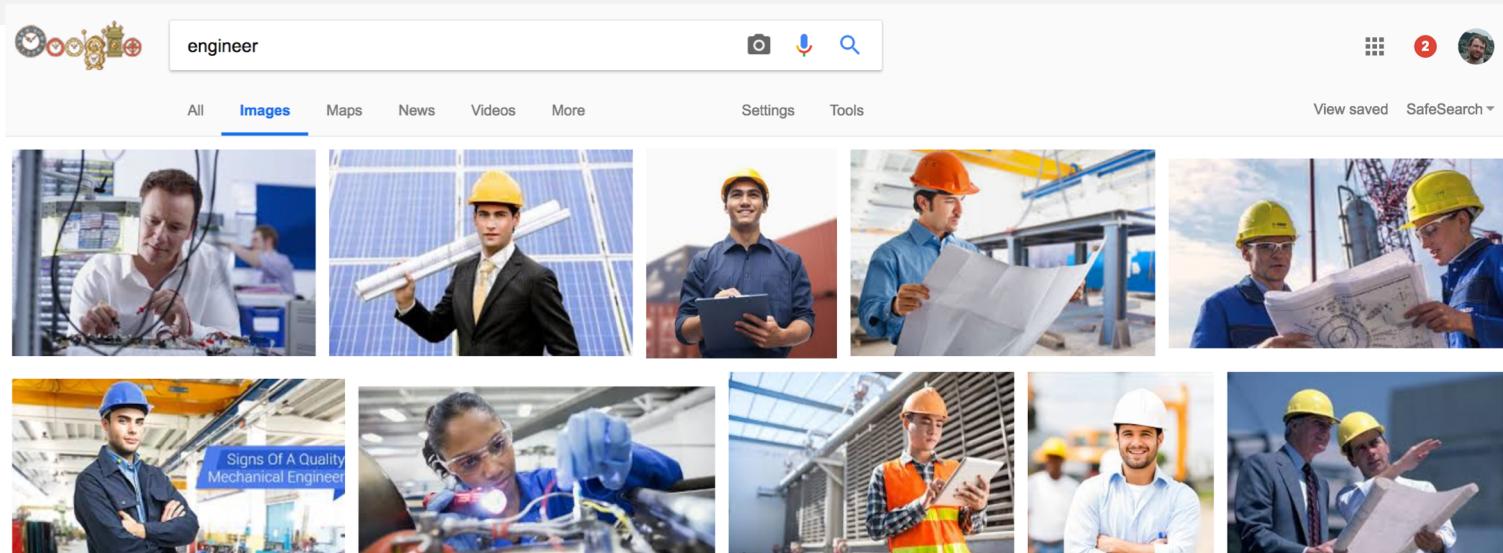
Ingenieurwesen

- **Fachleute auf dem Gebiet der Technik**
- Technik: von Menschen gemachten Gegenstände und deren Herstellung und Verwendung
- Das Wort Ingenieur kommt vom lateinischen "ingenium"
 - Bedeutung ungefähr: Erfindung, Scharfsinn
- Der Begriff ist im 16ten Jahrhundert entstanden und hat ursprünglich den Beruf des künstlerischen Erfinders beschrieben (z.B. Leonardo da Vinci)



Quelle teilw.: http://www.iaeng.org/about_IAENG.html

Ingenieure (I)



“Engineers apply the knowledge of the mathematical and natural sciences (biological and physical), with judgment and creativity to develop ways to utilize the materials and forces of nature for the benefit of mankind.

The subjects are diverse and include names like bioengineering, computer engineering, electrical and electronics engineering, financial engineering, industrial engineering, internet engineering and systems engineering, etc.”

Quelle: http://www.iaeng.org/about_IAENG.html

Ingenieure (II)

- Ingenieure konstruieren technische Systeme
- Oft interdisziplinär (Physik, Chemie, Biologie)
- Auch für neue Produkte: traditionell vorhandene Methoden und Mittel zur Herstellung
- Ingenieur beherrscht Herstellungsmethoden, Werkzeuge und Werkstoffe (Regeln, Standards, Erfahrungswissen)
- Oft Abwägung Kosten / Nutzen

Beispiel: Hausbau

Hausbau:

- <https://www.youtube.com/watch?v=tlnUVxqAzDU>

Software Engineering (II)

Software Engineering is...

erprobт, zuverлssig,
berechenbar, ...



... the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines. [NR68]

standardisiert, feste Garantien



... the systematic approach to the development, operation, maintenance, and retirement of software.“ [IEEE83]

SE Teilgebiete

SE Sub-Disziplinen

- Planung
- Analyse
- Entwurf
- Implementierung
- Verifizierung und Validierung
- Einführung
- Wartung

+ Unterstützungsprozesse

SE Methoden V&V

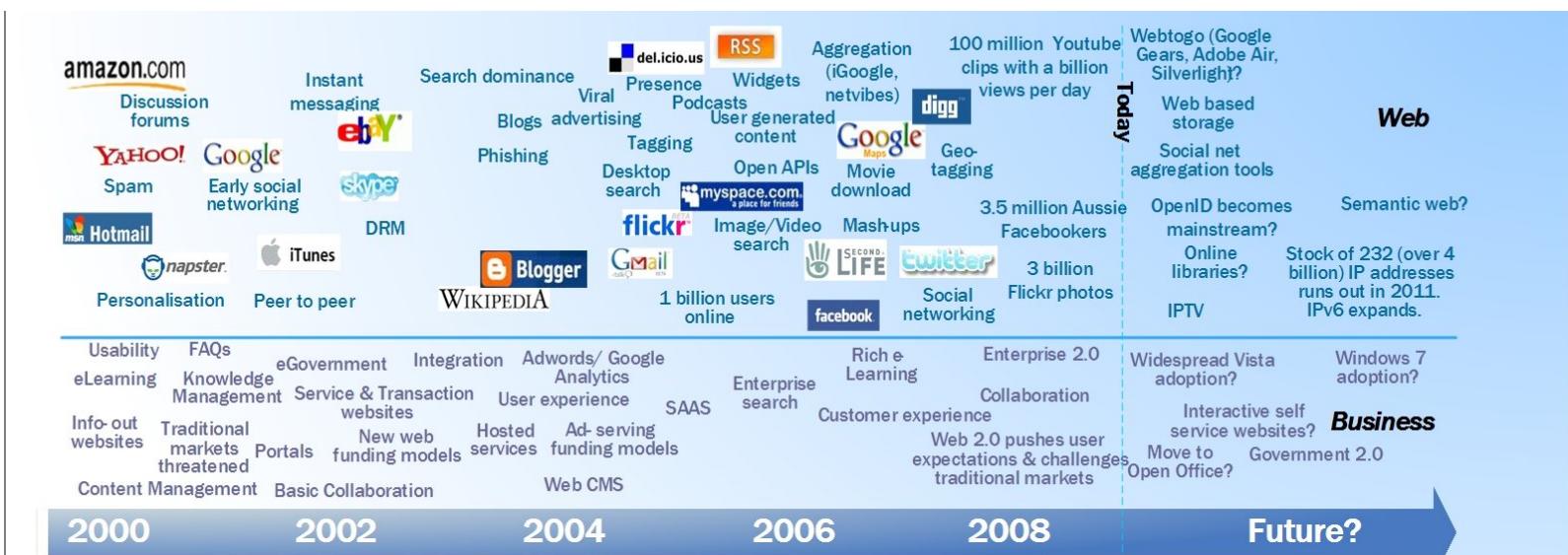
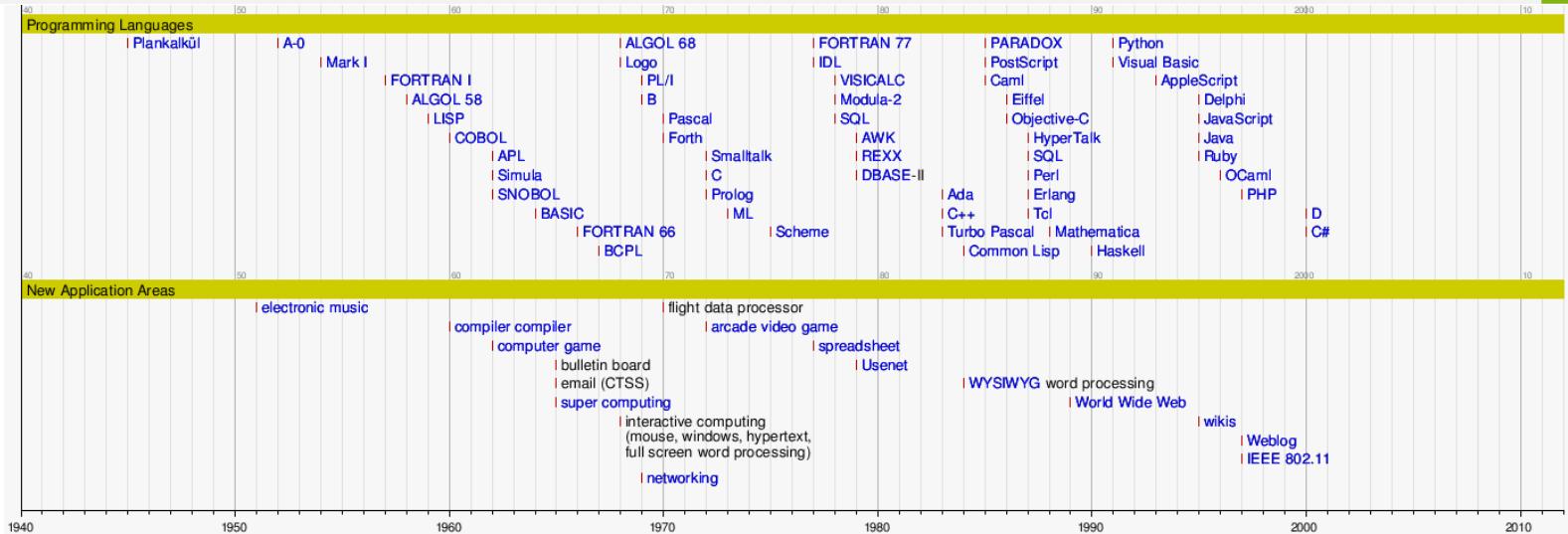
- Modultests
- Integrationstests
- Systemtests
- ...

SE Geschichte

Auszug SE Methoden Analyse Entwurf / Implementierung:

- [1968] Dijkstra "goto" ist gefährlich
- [frühe 1970er] Strukturierte Programmierung, erste Objektorientierung
- [späte 1970er] Erste IDEs
- [späte 1980er] CASE Werkzeuge
- [späte 1980er] Weitere Verbreitung von OOP durch C++ und OOD
- [frühe 1990er] Kommerzielle Tools für Anforderungsmanagement
- [später 1990er] Java, UML, ...
- [frühe 200er] Eclipse IDE, CASE Werkzeuge verschwinden

Geschichte (IV)



Software Engineering (IV)

- Neue Technologien (z.B. Web)
- Neue Klassen von Anwendungen (z.B. Apps)

⇒ Neue Paradigmen und Methoden (z.B. SCRUM)

⇒ Domänen-spezifische Lösungen

Kampfjet vs. Web-Anwendung



ABER: Generell gültige Ansprüche an Produkt,
Methoden und Ingenieur

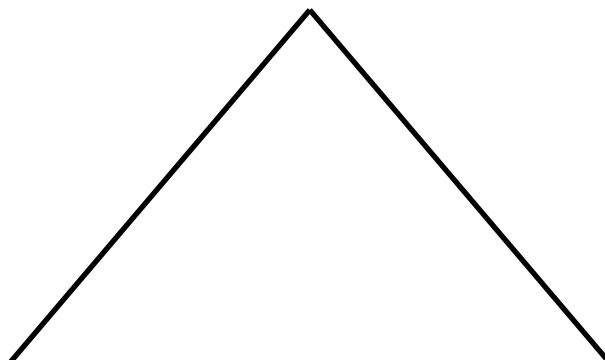
Anspruch an Software

- Korrektheit
- Zuverlässigkeit
- Sicherheit
- Effizienz
- Benutzbarkeit
- Wartbarkeit
- Lesbarkeit
- Erweiterbarkeit
- Wiederverwendbarkeit

Anspruch an Methoden

- Stellt Anforderungen an Software sicher
- Planbar
- Effizient

Qualität: korrekt
(funktionsfähig)



Termin:
rechtzeitig

Kosten:
im Budget

Anspruch an Ingenieur

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1. PUBLIC – Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT – Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT – Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION – Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES – Software engineers shall be fair to and supportive of their colleagues.
8. SELF – Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

ACM/IEEE Code of Ethics (© IEEE/ACM 1999)

Engineer vs. Software Engineer

Google search results for "engineer":

Images



Google search results for "software engineer":

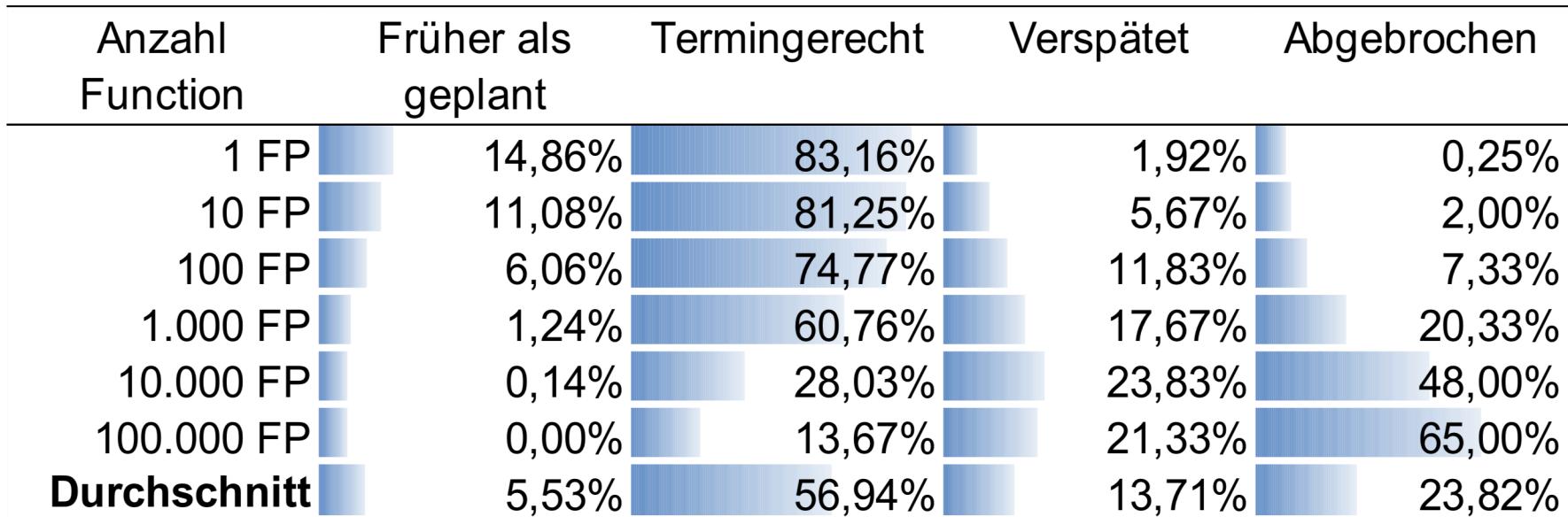
Images



Bestandsaufnahme (I)

- Entwicklungsprozesse der Software-Industrie bei weiten nicht so planbar, zuverlässig, effektiv, effizient und flexibel wie die „althergebrachter“ Industrien:
- Keine zuverlässige Herstellung von Software im industriellen Maßstab.
- Kosten- und Terminüberschreitungen.
- Bei Auslieferung: ungenügende Softwarereife.
- Keine Produktivitätskontrolle wie in anderen industriellen Fertigungsbereichen.
- Keine Qualitätskontrolle wie in anderen industriellen Fertigungsbereichen, gerade das Testen ist immer noch unterbewertet.

Bestandsaufnahme (II)

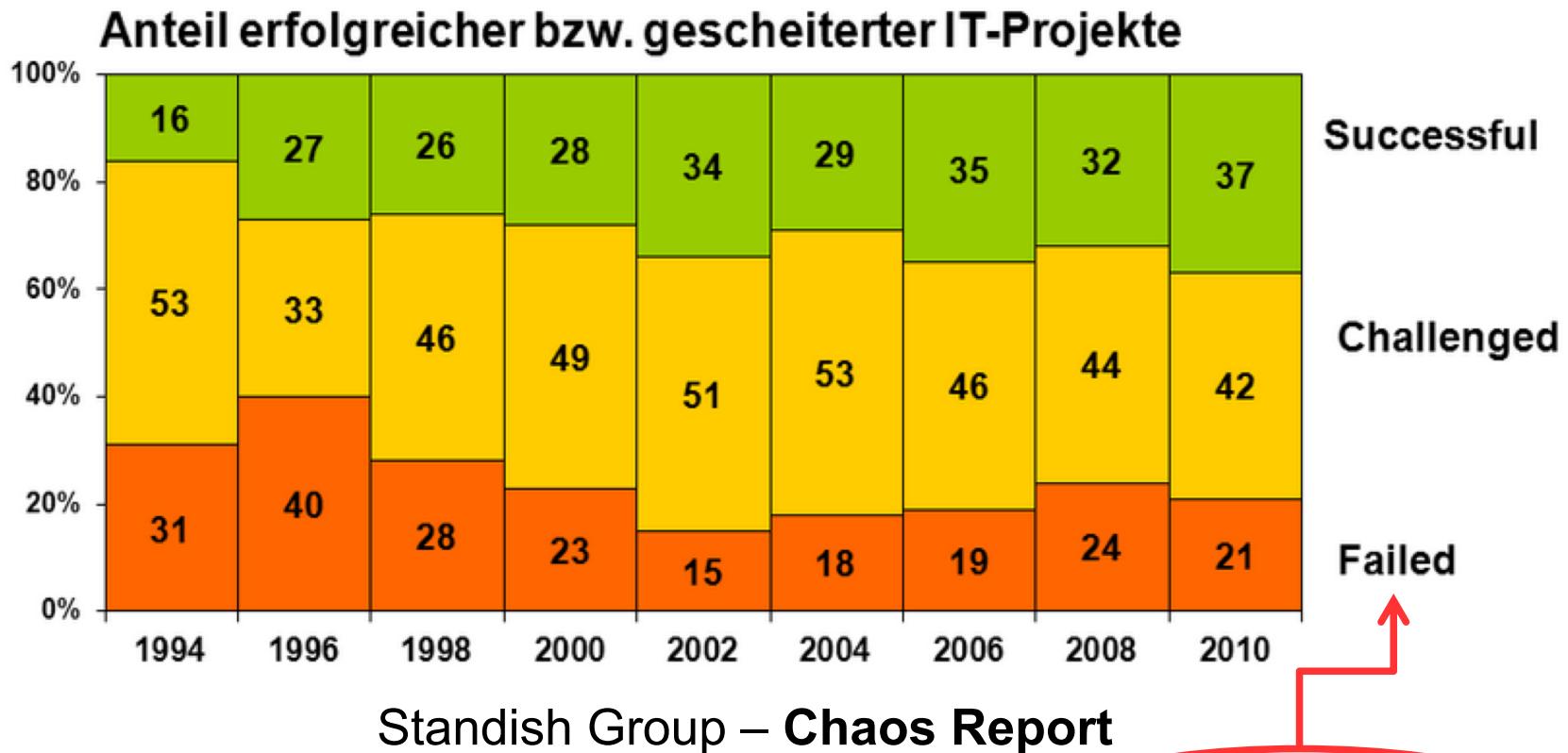


Abbruchrate großer Projekte nach [Jon96]

Andere Quelle: **Erfolgreich durchgeführte Software-Projekte:**
 • 16% im Jahre 1994; 34% im Jahre 2003

1 Function Point (FP): ca. 55 Lines of Code (LOC) in C++/Java, 20 LOC Perl, 13 LOC SQL, etc.
 Desktop-Projekt mit 1 FP dauert ca. eine Woche mit einem Entwickler
 Allgemeines Projekt mit 100.000 FP und 100 Entwicklern: ca. 17 Monate

Bestandsaufnahme (III)



Erfolgreich durchgeführte Software-Projekte:

.35% seit 15 Jahren

Gescheiterte Software-Projekte:

.20%

Abnahme gescheiterter Projekte:
Geänderte Kriterien

Quelle: <http://www.ikmt.de/public/homepage.htm#de/service/management-forum.htm#showthread.php@&tid=334&pid=#pid>

Ariane 5

- Maiden flight on June 4, 1996
- Crashed 40 seconds after takeoff
- Most expensive software bug in history:
500M USD (uninsured)



ncorrect integration of unnecessary legacy code:

- Uncaught exception in floating-point conversion
- Conversion from a 64-bit integer to a 16-bit signed integer
- Should only have been applied to a number less than 2^{15}
- Was applied to a greater number ⇒ Exception ...

<https://archive.eiffel.com/doc/manuals/technology/contract/ariane/>

Überlingen

- Two aircraft collided in mid air
- All 69 passengers and crew died



Collision Avoidance System (TCAS) was not designed to handle situation:

- Both aircraft were level at FL360
- Swiss ATC instructed TU-154 to descend (to avoid a conflict)
- TU-154 acknowledged instruction late (after second attempt)
- TCAS instructed Boeing 757 to descend (to resolve conflict)

<http://www.1001crash.com>

Mars Rover

- Abnormal Behavior on Sol 18
- Analysis and repair for 11 sols
- Returned to normal operation on Sol 33



Flash file system crashed after (short) sequence of untested events:

- Combination of conceptual flaws and configuration errors in file system abstraction and memory management
- Caused sequence of out-of-memory, reboot, out-of-memory, ...
- Resolved through backup systems

Knight Trading

- August 1, 2012: bug in software update
- 397M shares traded in 45 minutes
- Major disruption in the prices of 148 companies
- E.g., Wizzard Software Corporation went from USD 3.50 to USD 14.76.
- Knight Capital lost USD 440 million (pre-tax).



Software started unexpected legacy function after deployment:

- Re-purposed a flag
- Did not update original reader of flag on one machine
- Flag enabled code that was designed to verify the behavior of trading algorithms in controlled environments (by moving stock prices higher and lower)

https://en.wikipedia.org/wiki/Knight_Capital_Group

Heartbleed

- Anyone on the Internet can read the memory of the systems protected by vulnerable versions of OpenSSL
- Affected about 17% of secure certified servers

*"We attacked ourselves from outside, without leaving a trace.
... we were able steal from ourselves
user names and passwords, instant messages, emails and
business critical documents and communication."*



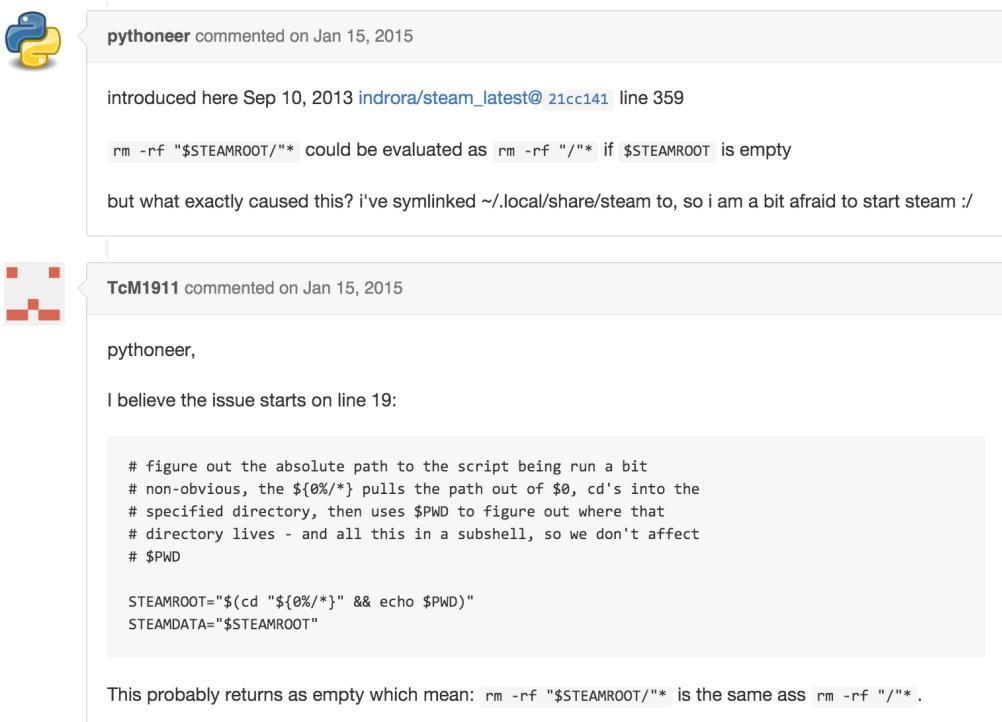
Variable bounds not checked, Assumptions on Caller Side:

- Clients can send heartbeat requests, asking for a response of length k
- OpenSSL sends response of length k
- Server copied memory chunk of length k without checking

<http://heartbleed.com/>

Steam

- Linux client for some users deletes entire home directory (and more...)
- Lost data cannot be recovered in most cases



pythoneer commented on Jan 15, 2015

introduced here Sep 10, 2013 [indrora/steam_latest@21cc141](#) line 359

`rm -rf "$STEAMROOT/*" could be evaluated as rm -rf /* if $STEAMROOT is empty`

but what exactly caused this? i've symlinked ~/.local/share/steam to, so i am a bit afraid to start steam :/

TcM1911 commented on Jan 15, 2015

pythoneer,

I believe the issue starts on line 19:

```
# figure out the absolute path to the script being run a bit
# non-obvious, the ${0%/*} pulls the path out of $0, cd's into the
# specified directory, then uses $PWD to figure out where that
# directory lives - and all this in a subshell, so we don't affect
# $PWD

STEAMROOT="$(cd "${0%/*}" && echo $PWD)"
STEAMDATA="$STEAMROOT"
```

This probably returns as empty which means: `rm -rf "$STEAMROOT/*"` is the same as `rm -rf /*`.

Unsafe Code: Variable contents not checked, delete warnings suppressed:

- Variable \$STEAMROOT can be empty
- Program can execute command `rm -Rf /`

<https://github.com/valvesoftware/steam-for-linux/issues/3671>

Dieselgate

Dienstag, 22. September 2015

Der Hausmeister war's! VW präsentiert Einzeltäter hinter Abgas-Affäre

f 37.7K t 4.1K



Investigation ongoing ...

Wolfsburg (dpo) - Im Skandal um manipulierte Abgaswerte von Dieselfahrzeugen setzt VW nach Tagen der Kritik überraschend auf volle Transparenz und legt den alleinigen Verantwortlichen offen. Demnach soll der 61-jährige Hausmeister Hans Böbner seinen Zugang zur Firmenzentrale in Wolfsburg genutzt haben, um eigenhändig Manipulationen an der Software mehrerer Millionen Fahrzeuge vorzunehmen. Inzwischen hat Böbner selbst ein erstes Geständnis abgelegt.

Auschnitt aus Der Postillon

Autonome Fahrzeuge

Toyota (Gas Pedal Stuck)



Google (AV crashes into bus)



Tesla (AV cuts under truck)



Uber (AV kills cyclist)



Facebook / Cambridge Analytica

- Cambridge Analytica „klaut“ Nutzerdaten von Facebook
- Erstellt psychologische und politische Profile der Nutzer (**Nix: zwischen 4 und 5 tausend Datenpunkte zu jedem erwachsenen US Amerikaner**)
- Profile werden Basis für Werbe Kampanien vor US Wahl 2016
- Trump wird überraschend Präsident



Fakt oder Fiktion?

Andererseits ...

BER

- Eröffnungstermine: 2007, 2011, 2012, 2013 ...
- Aktuell geplant 2019

Falcon Rakete landet auf Fuß:



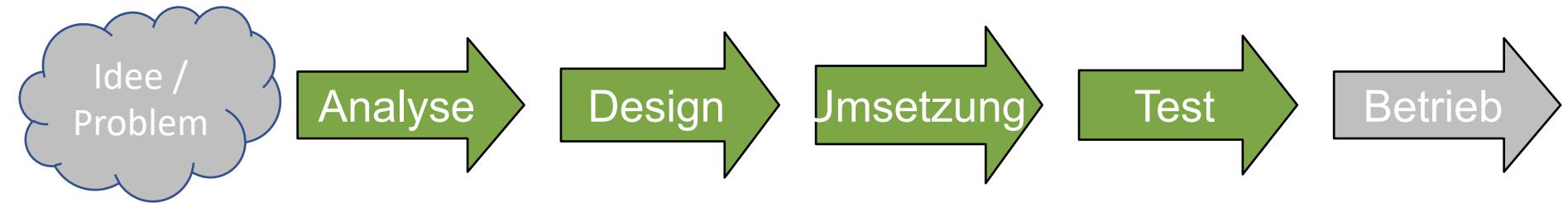
Zusammenfassung

- Software Engineering zielt ab auf die **ingenieurmäßige** Entwicklung, Wartung, Anpassung und Weiterentwicklung großer Softwaresysteme.
- Verwendung **bewährter systematischer Vorgehensweisen**, Prinzipien, Methoden und Werkzeuge.
- Methoden und Vorgehen teilweise spezifisch für Domäne.
- Berücksichtigung der Aspekte: **Kosten, Termine, Qualität**.

Agenda

- Organisatorisches
- Vorstellung des Lehrstuhls
- Kurze Einführung: Was ist Software Engineering
- Vorlesungsinhalte

Orientierung an Entwicklungsphasen



Aspekte:

- Methoden
- Qualität / Metriken
- Werkzeuge

Querschnittsthemen:

- Vorgehen
- Projekt Management

Teil 1: Analyse

... lautete die Anforderung:

'Absenkung der Mautpflichtgrenze auf 7,5 Tonnen' ohne weitere Differenzierung der Maut nach Gewichtsgrenzen bzw. Gewichtsklassen."

Analyse

Martin Rickmann, Sprecher Toll Collect GmbH

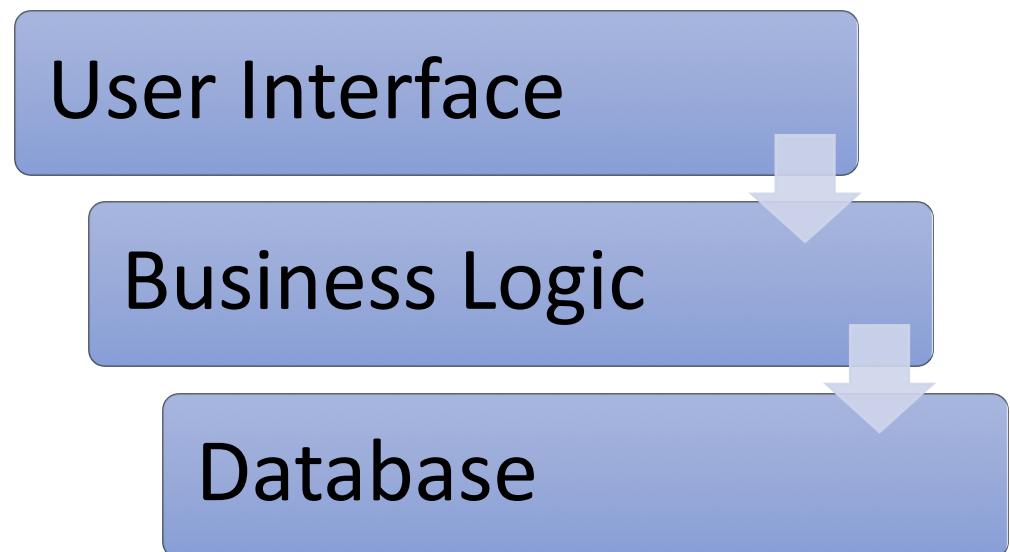
Anforderungen mit Formalisierung

- Verschiedene Formen von Anforderungen
- Wie erhebt dokumentiert man Anforderungen
- Beispielhafte Formalismen
- Umfang von Anforderungen / Metriken für Anforderungen

Teil 2: Design

Architektur / System Design

- Architektur Stile
- Zerlegung
- Architektur Metriken

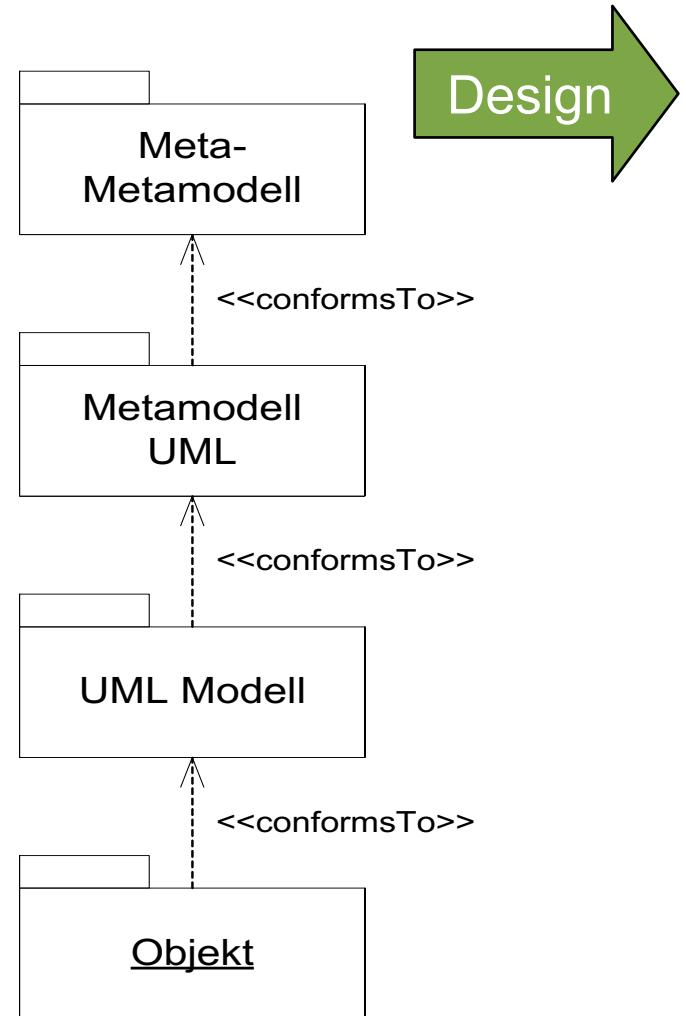


Design

Teil 2: Design

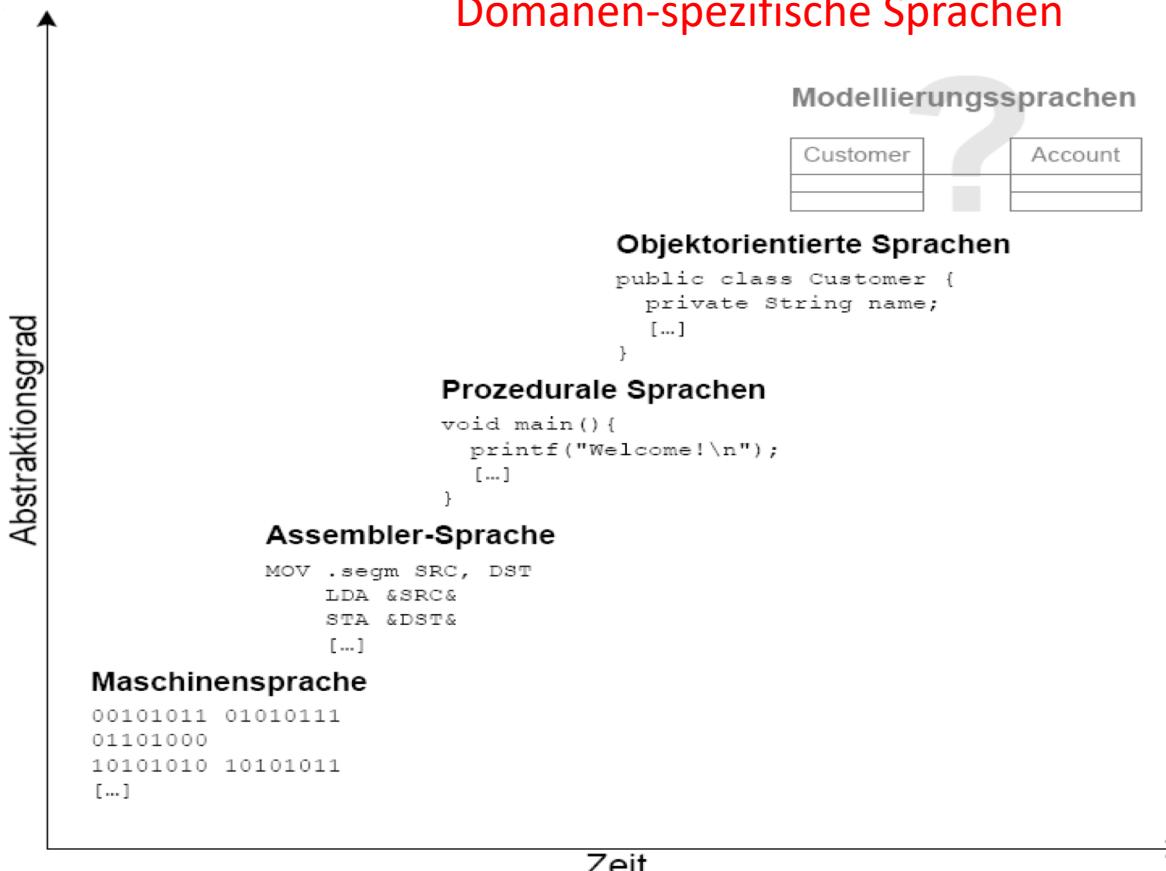
Modellgetriebene Entwicklung

- Domain Modellierung
- Metamodellierung
- Best Practices / Patterns
- UML schon bekannt aus SWT und SoPra.
- Hier neu:
 - Metamodellierung
 - Spezifikation der UML

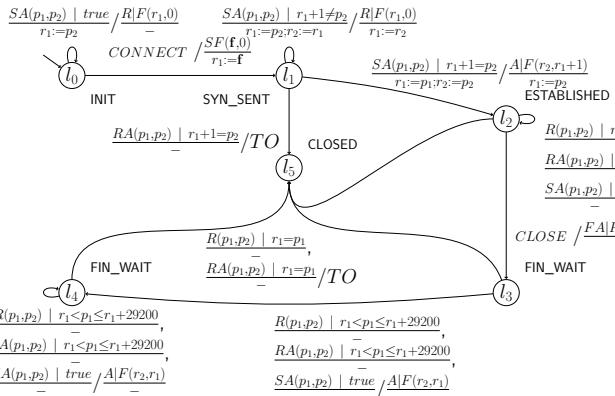
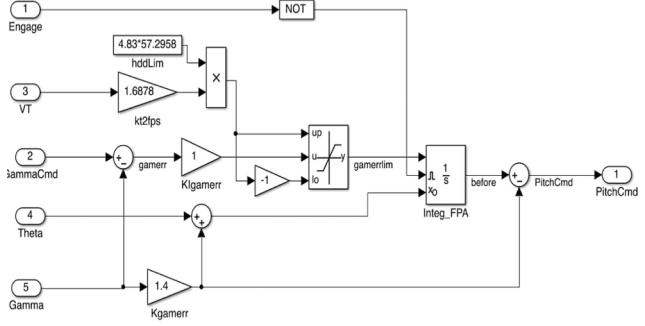


Teil 3: Umsetzung

Design



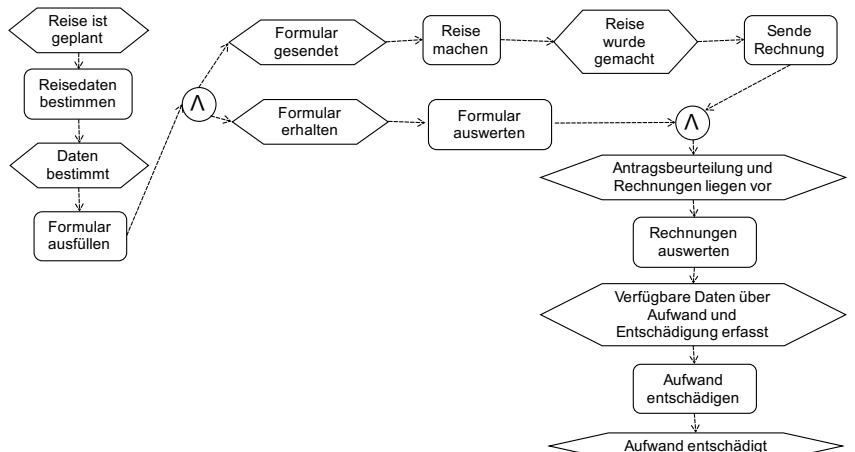
Teil 3: Umsetzung



Design

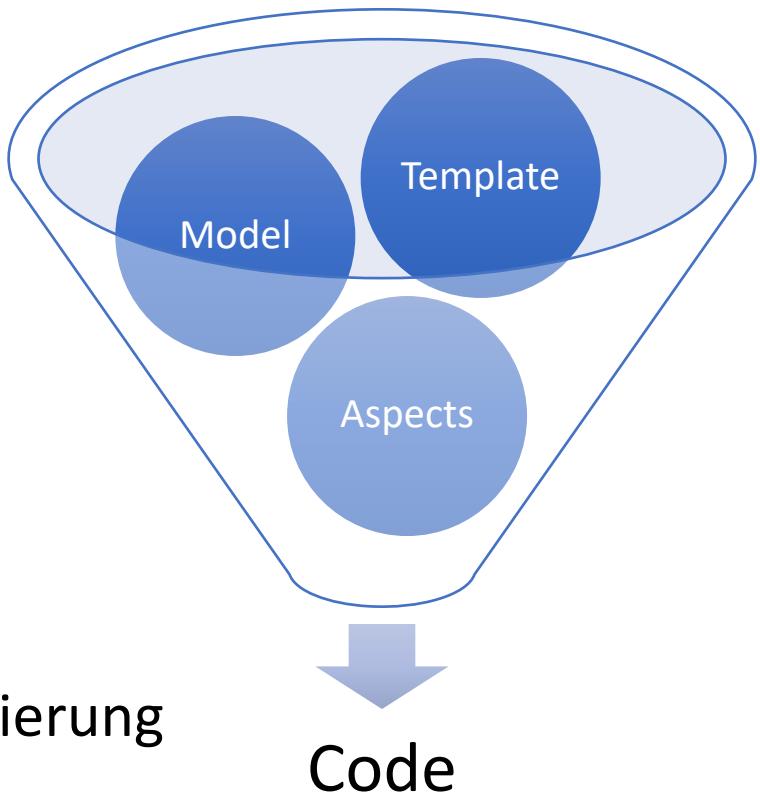
Modellierung von Verhalten

- Blockschaltdiagramme
- Zustandsautomaten



Teil 3: Umsetzung

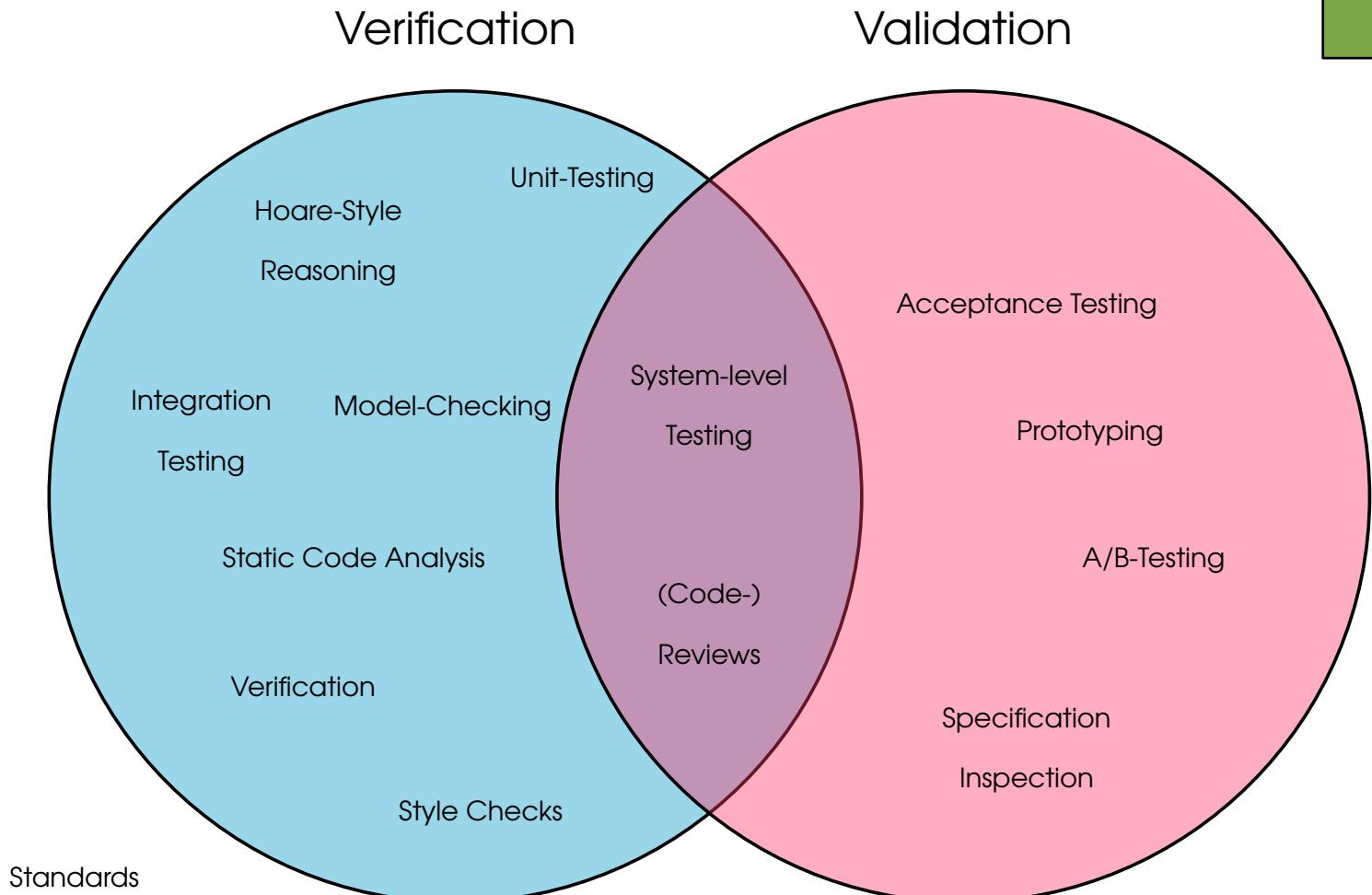
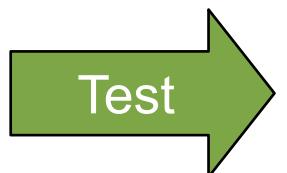
Umsetzung



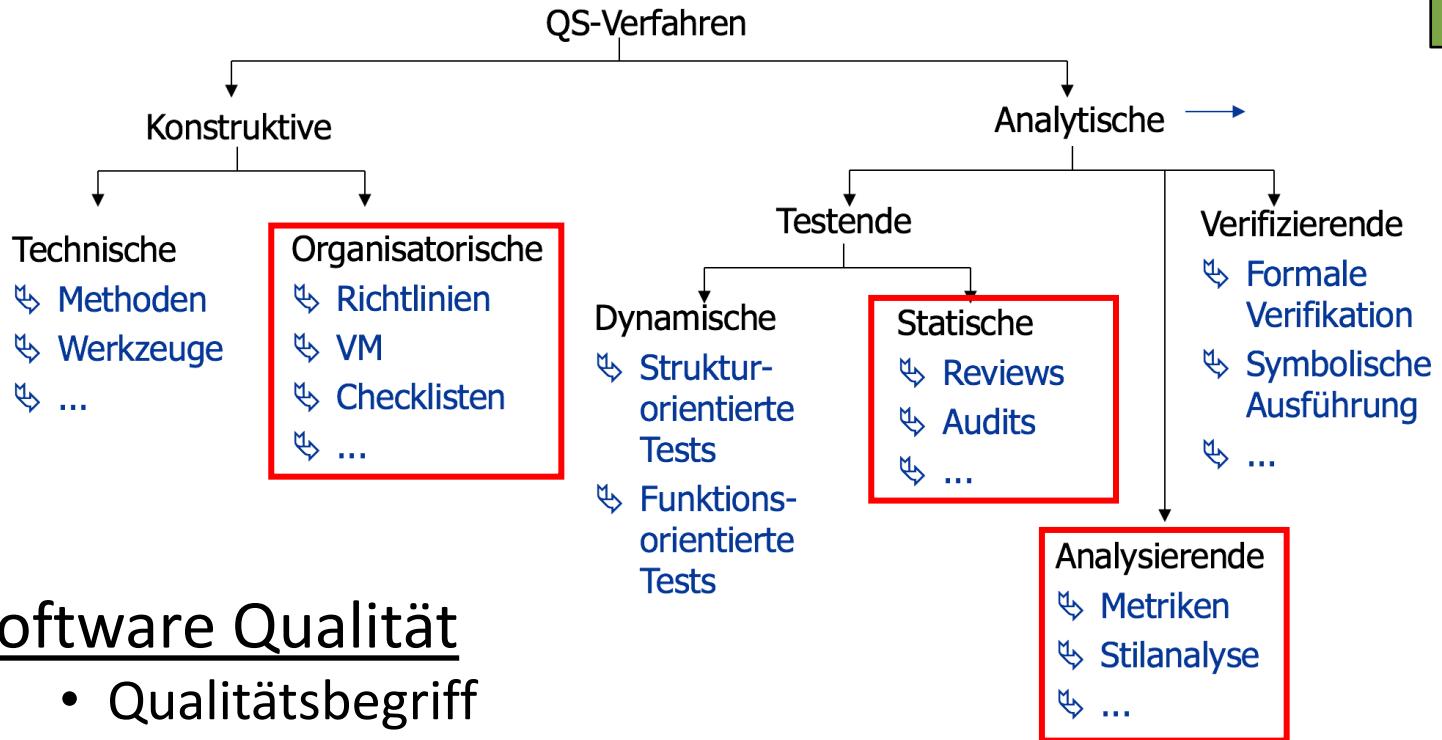
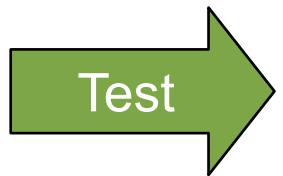
Code Generierung

- Model-Transformation
- Template-basierte Code Generierung

Teil 4: Qualitätssicherung



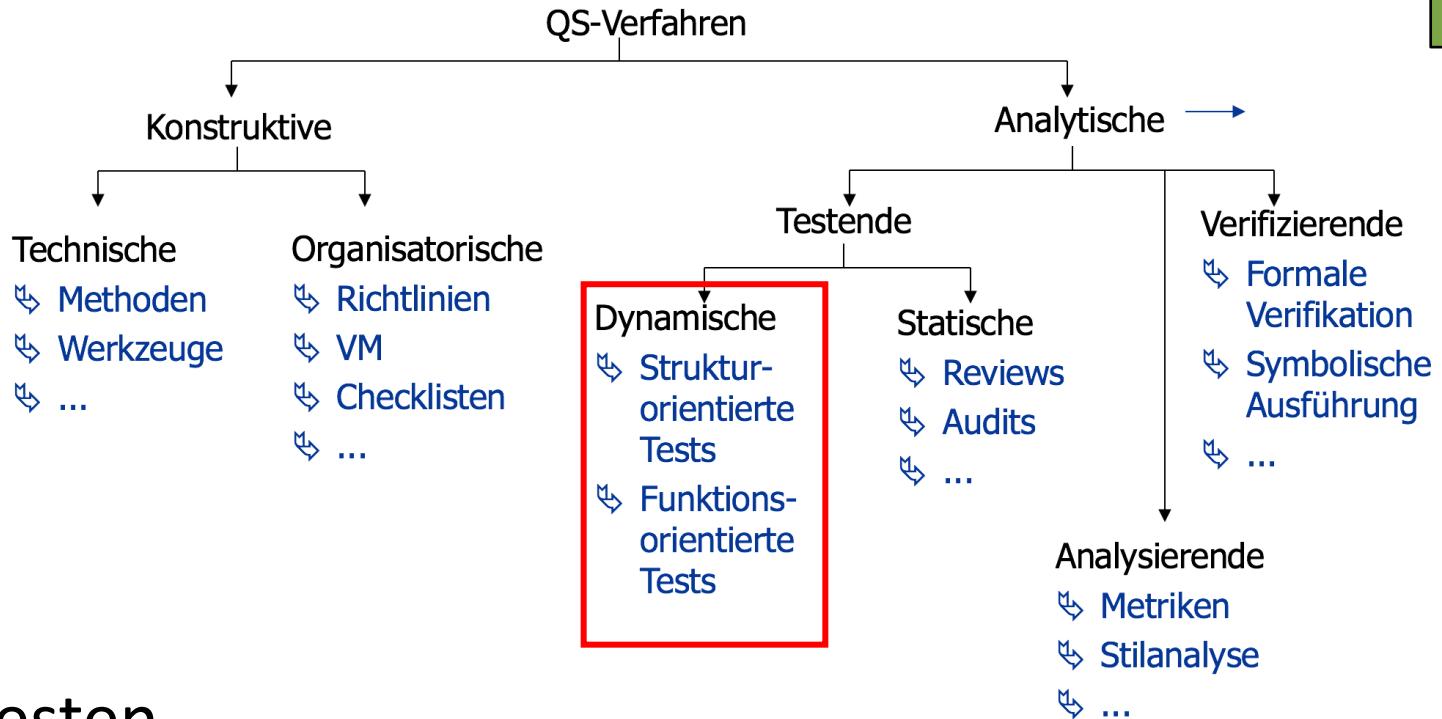
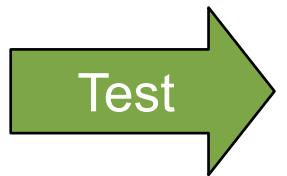
Teil 4: Qualitätssicherung



Software Qualität

- Qualitätsbegriff
- Richtlinien
- Reviews
- Metriken

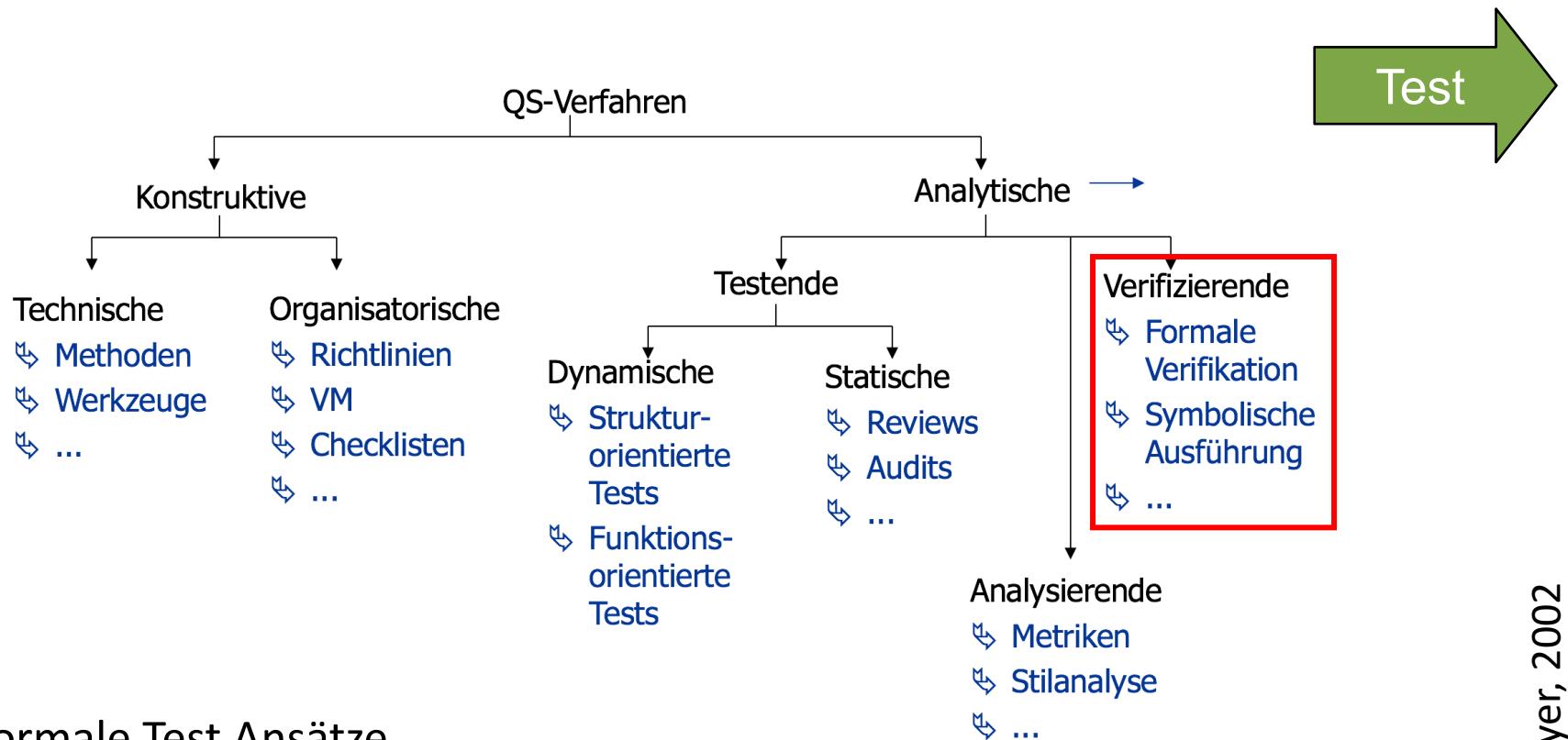
Teil 4: Qualitätssicherung



Testen

- Wdh. Basiswissen Testen
- Integrations- / Systemtest (Mocking, Test-Umgebung)

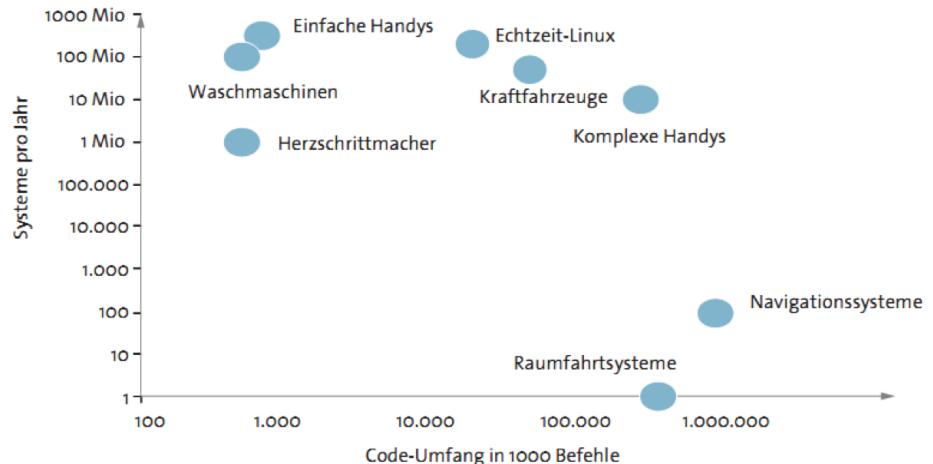
Teil 4: Qualitätssicherung



Formale Test Ansätze

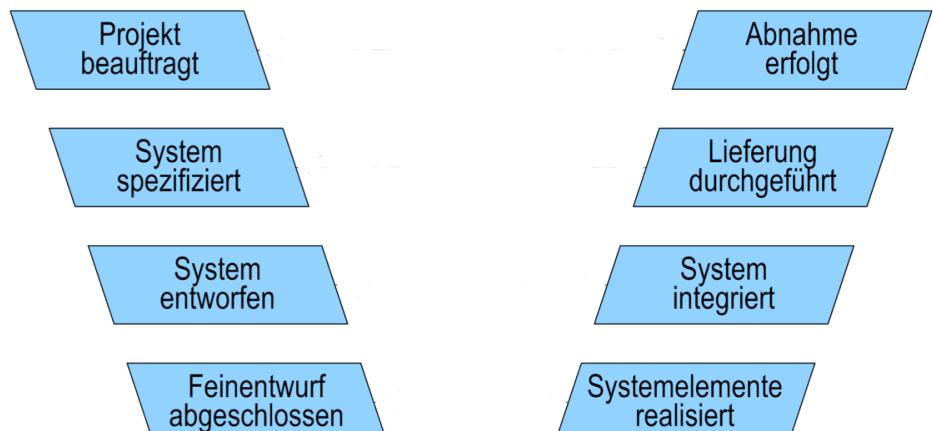
- Statische Analyse
- Formale Analyse von Verhalten

Teil 5: Querschnittsthemen



Projektmanagement

- Vorgehensmodelle
- Projektplanung
- Aufwandschätzung



Teil 5: Querschnittsthemen

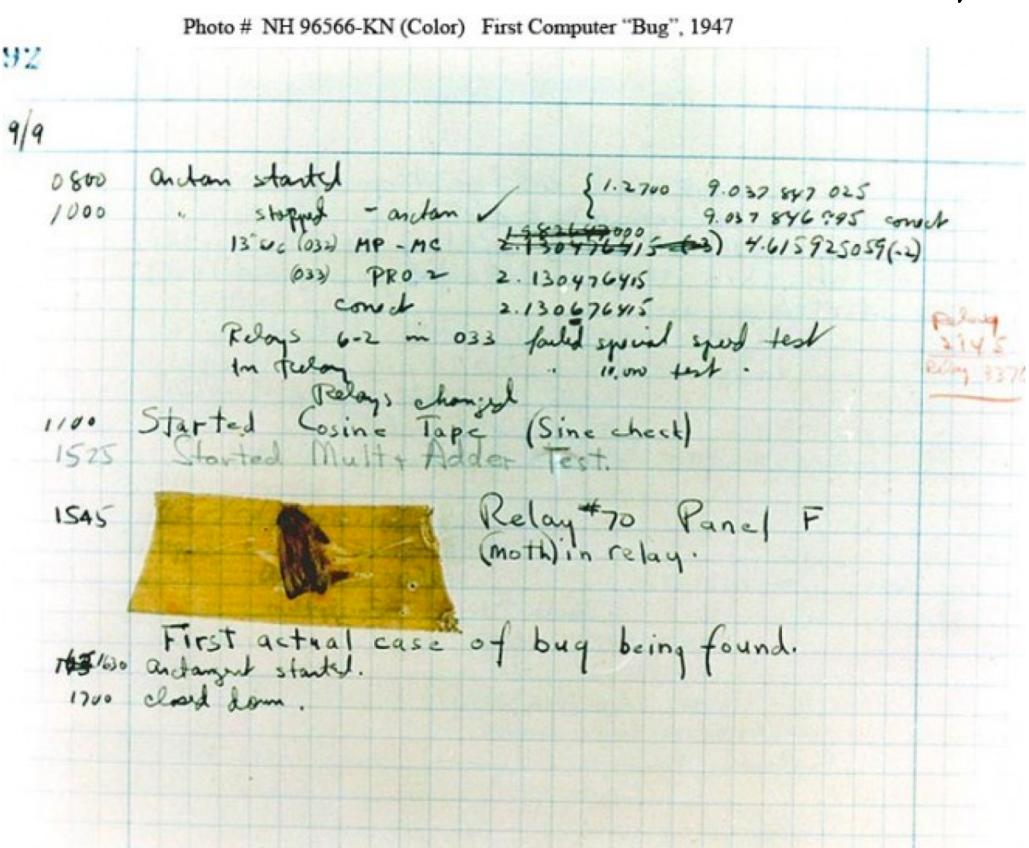
Jmsetzung

Arbeiten mit Code

- Code Organisation
- Repositories

Auslieferung

- Build Prozess
- Automatisierung
- Abhängigkeiten



Agenda

- Organisatorisches
- Vorstellung des Lehrstuhls
- Kurze Einführung: Was ist Software Engineering
- Vorlesungsinhalte

Ihre Fragen?