SOFTWARE QUALITY

© Matthias Heyde / Fraunhofer FOKUS

Prof. Dr.-Ing. Ina Schieferdecker
July 22, 2016

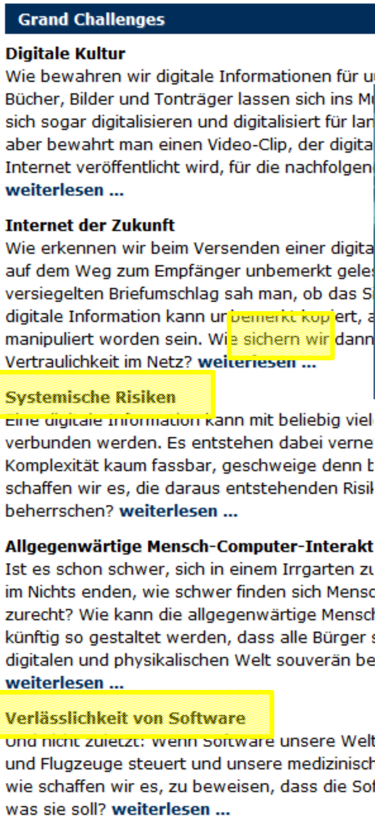
Engineering a Connected World

Fraunhofer
FOKUS

SOFTWARE OUTAGES ARE COSTLY AND DANGEROUS

1. 1996: European Ariane 5 rocket; over $370 million loss
2. 1985-1987: Therac-25 medical radiation therapy; patients received up to 100 times the intended dose, and at least three of them died
3. 2010: Virgin Blue's Reservation Desk outage; $20 million loss
4. 2012: Wrong orders for Knight Capital Group's funds; $440 million loss
5. …
6. Jan. 2016: IHS study; **costs to North American companies of $700 billion a year for ICT outages**. This includes lost employee productivity (78%), lost revenue (17%), and actual costs to fix the downtime issues (5%).

OTHER'S VIEWS



Grand Challenges

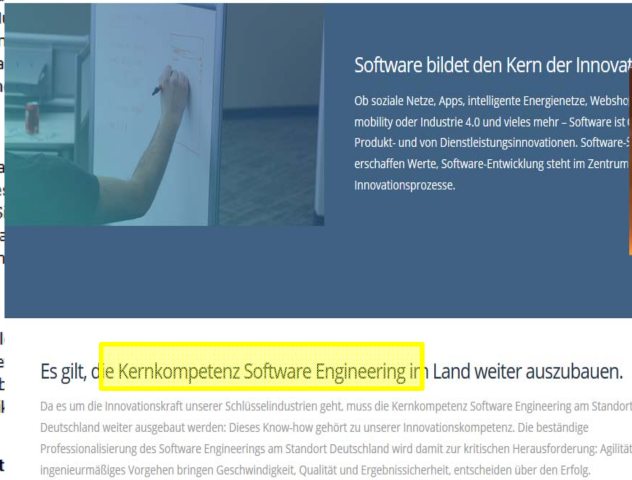
Digitale Kultur
Wie bewahren wir digitale Informationen für unsere Nachwelt auf?
Bücher, Bilder und Tonträger lassen sich ins Mu
sich sogar digitalisieren und digitalisiert für lan
aber bewahrt man einen Video-Clip, der digita
Internet veröffentlicht wird, für die nachfolgen
weiterlesen ...

Internet der Zukunft
Wie erkennen wir beim Versenden einer digita
auf dem Weg zum Empfänger unbemerkt geles
versiegelten Briefumschlag sah man, ob das Si
digitale Information kann unbemerkt kopiert, a
manipuliert worden sein. Wie sichern wir dann
Vertraulichkeit im Netz? **weiterlesen ...**

Systemische Risiken
Eine digitale Information kann mit beliebig viel
verbunden werden. Es entstehen dabei verne
Komplexität kaum fassbar, geschweige denn b
schaffen wir es, die daraus entstehenden Risik
beherrschen? **weiterlesen ...**

Allgegenwärtige Mensch-Computer-Interakt
Ist es schon schwer, sich in einem Irrgarten zu
im Nichts enden, wie schwer finden sich Menschen im virtuellen Raum
zurecht? Wie kann die allgegenwärtige Mensch-Computer-Interaktion
künftig so gestaltet werden, dass alle Bürger sich in der gemischt
digitalen und physikalischen Welt souverän bewegen können?
weiterlesen ...

Verlässlichkeit von Software
Und nicht zuletzt: Wenn Software unsere Welt regiert, unsere Autos
und Flugzeuge steuert und unsere medizinischen Instrumente dirigiert,
wie schaffen wir es, zu beweisen, dass die Software genau das tut,
was sie soll? **weiterlesen ...**



Software bildet den Kern der Innovationen

Ob soziale Netze, Apps, intelligente Energienetze, Websho
mobility oder Industrie 4.0 und vieles mehr – Software ist
Produkt- und von Dienstleistungsinnovationen. Software-S
erschaffen Werte, Software-Entwicklung steht im Zentrum
Innovationsprozesse.

Es gilt, die Kernkompetenz Software Engineering im Land weiter auszubauen.

Da es um die Innovationskraft unserer Schlüsselindustrien geht, muss die Kernkompetenz Software Engineering am Standort Deutschland weiter ausgebaut werden: Dieses Know-how gehört zu unserer Innovationskompetenz. Die beständige Professionalisierung des Software Engineerings am Standort Deutschland wird damit zur kritischen Herausforderung: Agilität ingenieurmäßiges Vorgehen bringen Geschwindigkeit, Qualität und Ergebnissicherheit, entscheiden über den Erfolg.

Management-Summary der GTB Softwaretestumfrage 2015/2016

Frank Simon, Manuel Fischer, Karin Vosseberg, Andreas Spillner, Kai Lepler, Mario Winter

April 2016

Neue IT-Trends wie Industrie 4.0/Internet of Things (48%), BigData (50%) oder auch Mobile (71%) sind aus Sicht des Managements zukunftsrelevant und werden in den Unternehmen bereits angemessen berücksichtigt. Von den Testenden wird die Vorbereitung auf die Trends kritischer gesehen: Die positive Einschätzung fällt hier durchschnittlich um 10%-20% geringer aus. Diesem kritischen Blick wird auch von der Forschungsseite zugestimmt: 74% sehen hier noch einen deutlichen Forschungsbedarf für IoT, 60% für Big Data und 50% für Mobile.

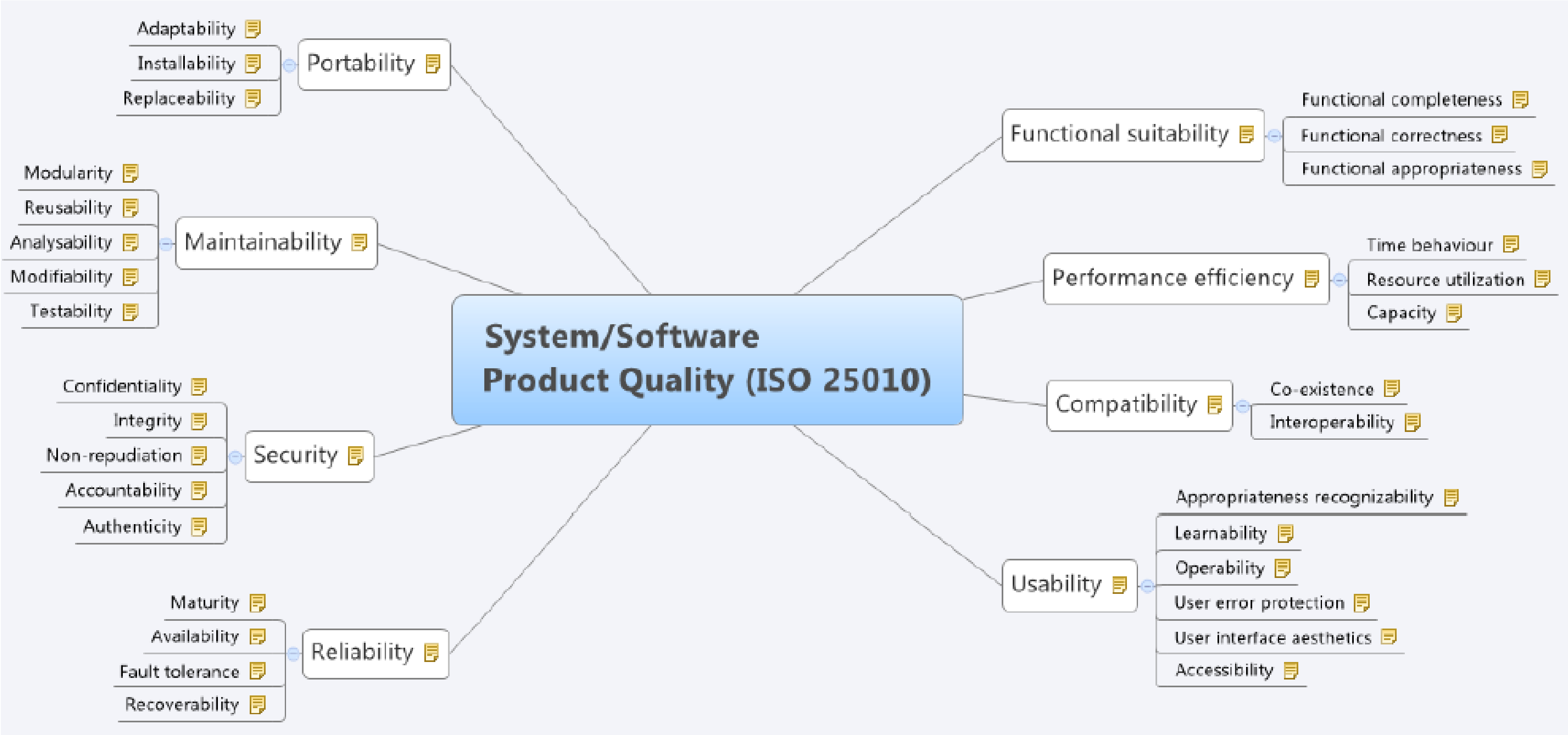
Herausforderungen des IoT-Testings

Als Top-Anforderungen werden hierbei Sicherheit, Interoperabilität und Konnektivität gesehen, wobei unter den Antwortenden, die bereits eigene IoT-Lösungen am Markt haben, die Leistungsfähigkeit der IoT-Lösungen an die dritte Stelle rückt.

OUTLINE

1. Status software quality
2. Some history
3. Some future perspectives

WHAT IS SOFTWARE QUALITY



Fraunhofer FOKUS

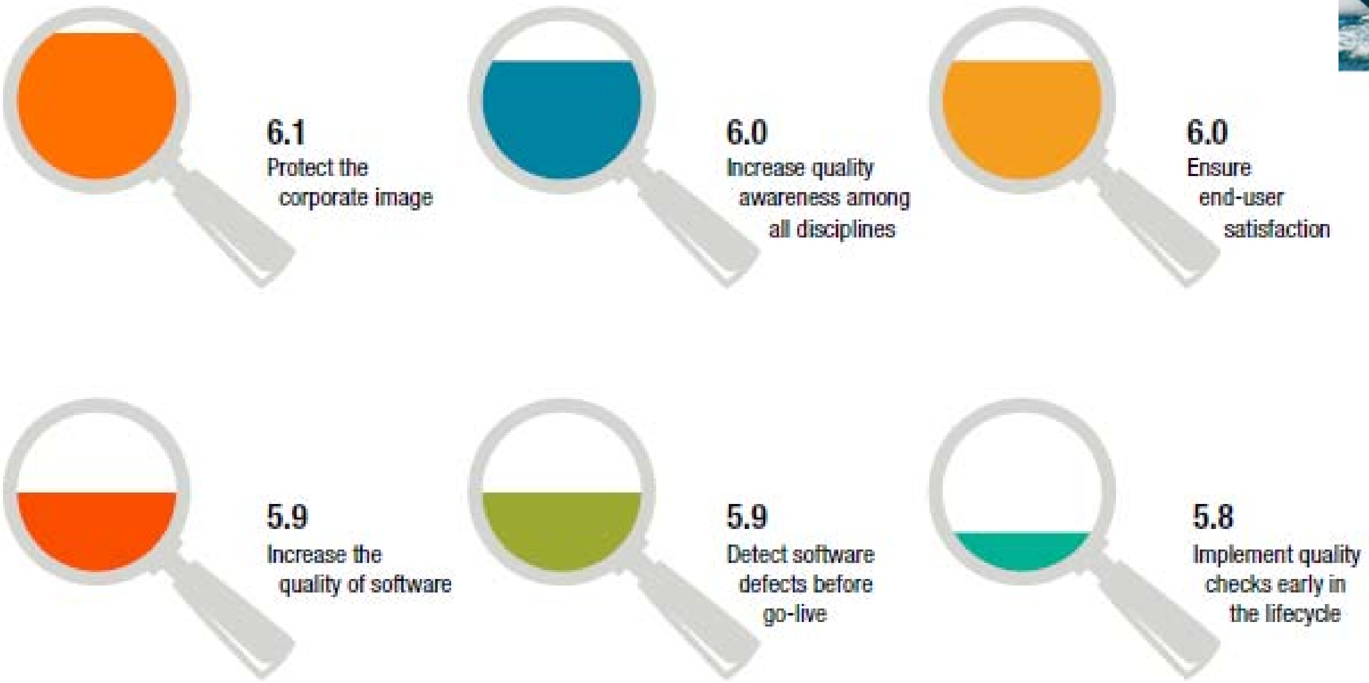
WHY IS IT IMPORTANT

1560 CIOs and IT and testing leaders

From 32 countries across the globe

Scale of 1-7 with 7 highest

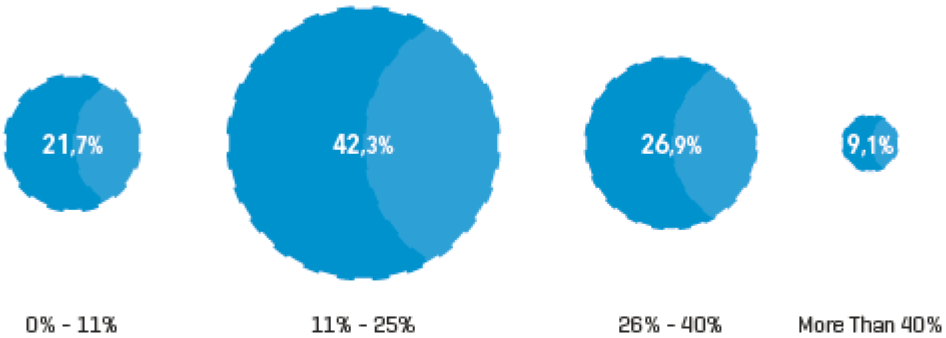
Management objectives with QA & Testing

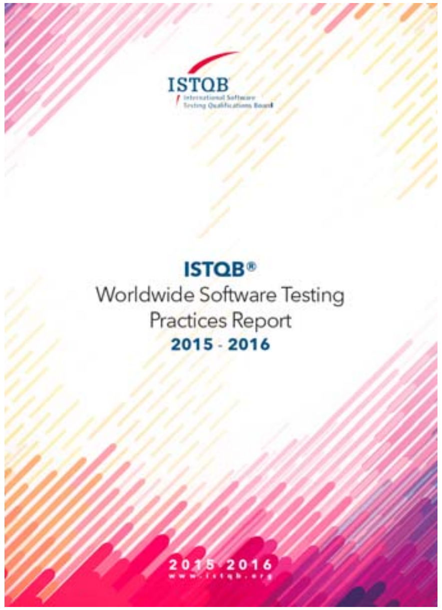


WHY IS IT IMPORTANT

About 3,200 respondents from 89 countries

What percent of a typical IT/ R&D project budget is allocated to software testing?





ISTQB
International Software Testing Qualifications Board

ISTQB®
Worldwide Software Testing Practices Report
2015 - 2016

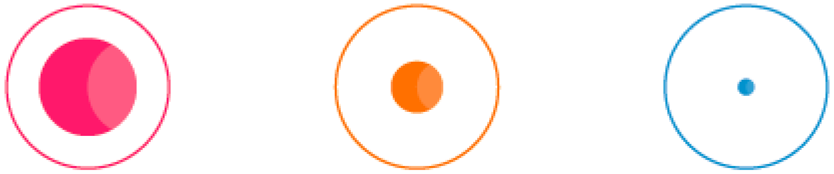
2015 2016
www.istqb.org

The large majority of respondents indicate budgets between 11% and 40%.
This is in line with World Quality Report 2015-16 that indicates an average expenditure of 26% for 2014 and 35% for 2015.

WHY IS IT IMPORTANT

About 3,200 respondents from 89 countries

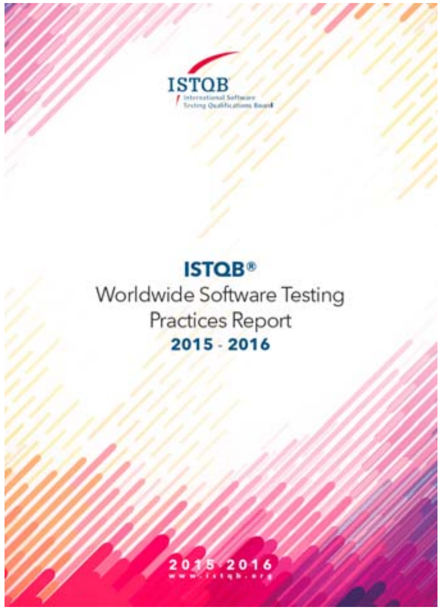
What is your expectation for your organization’s software testing budget in the next 12 months?



59,2%	31%	9,8%
Growth	Stable	Decrease

16,6%	1%-10% Growth
14,7%	11%-20% Growth
11,9%	21%-30% Growth
6,2%	31%-40% Growth
4,1%	41%-50% Growth
5,7%	More Than 50% Growth

6,5%	Slight Decrease
3,3%	Signification Decrease



ISTQB
International Software Testing Qualifications Board

ISTQB®
Worldwide Software Testing
Practices Report
2015 - 2016

2015 2016
www.istqb.org

About 60% of the respondents expect an increase of the budgets allocated to testing; this confirms the growing trend exhibited in the World Quality Report 2015-16, which forecasts that by 2018 the IT budget allocated to QA & testing will rise to 40%.

Average expected growth is 14% which is in line with the forecasted CAGR of the Global Testing Market in 2015-2019 in the Technavio Report (www.technavio.com)

OUTLINE

1. Status software quality

2. Some history

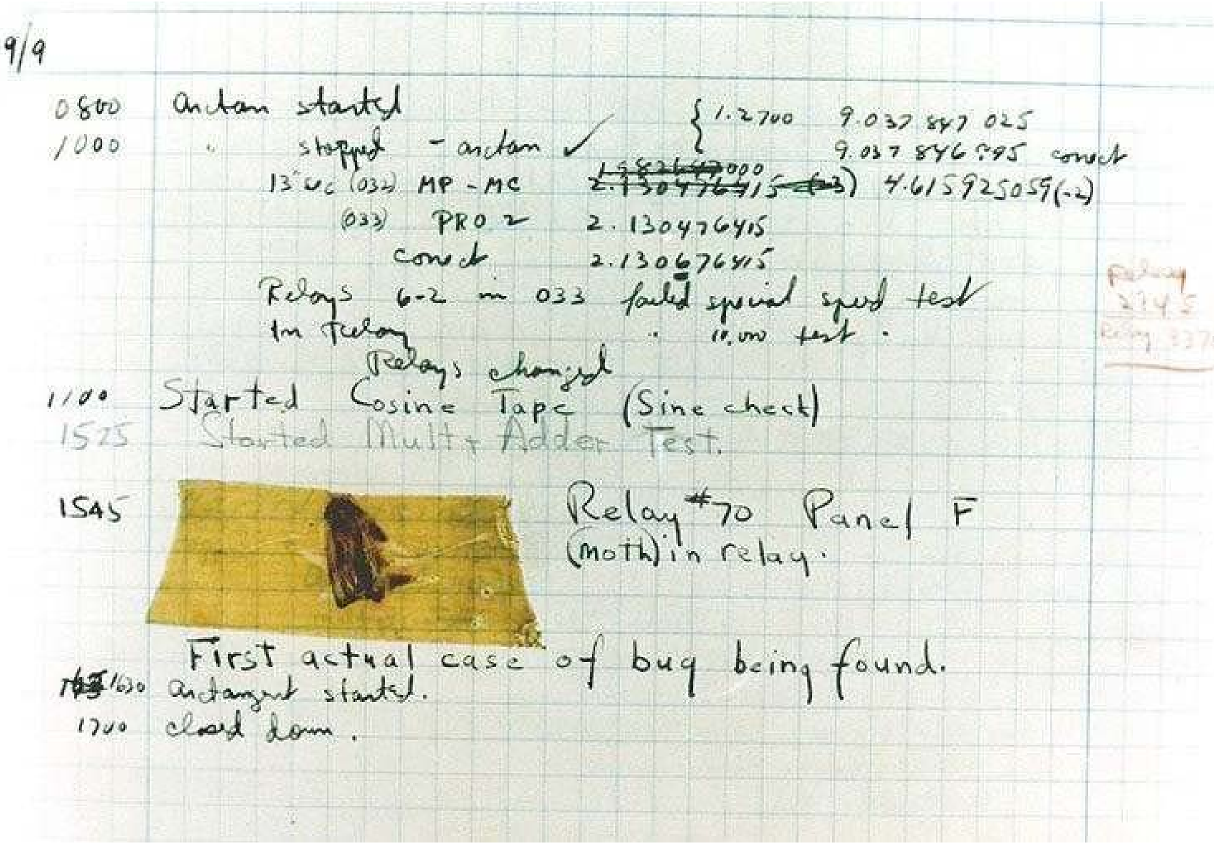
3. Some future perspectives

9.9.1945 15:45

The first „software bug“

- A moth in the computer Mark II causes a defect in Relay No. 70, Panel F.
- Mrs. Grace Murray Hopper removes the defect and records it in the log book.

  »First actual case
  of bug being found.«



9/9

0800 Antan started

1000 " stopped - antan ✓ {1.2700 9.037 847 025
9.037 846 995 conct

13" uc (033) MP - MC ~~1.982147000~~ ~~2.130476415 (-3)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

conct 2.130676415

Relays 6-2 in 033 failed special speed test
In relay " 11.000 test .

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

~~143~~ 1630 antangent started.

1700 closed down.

Relay 2145
Relay 337

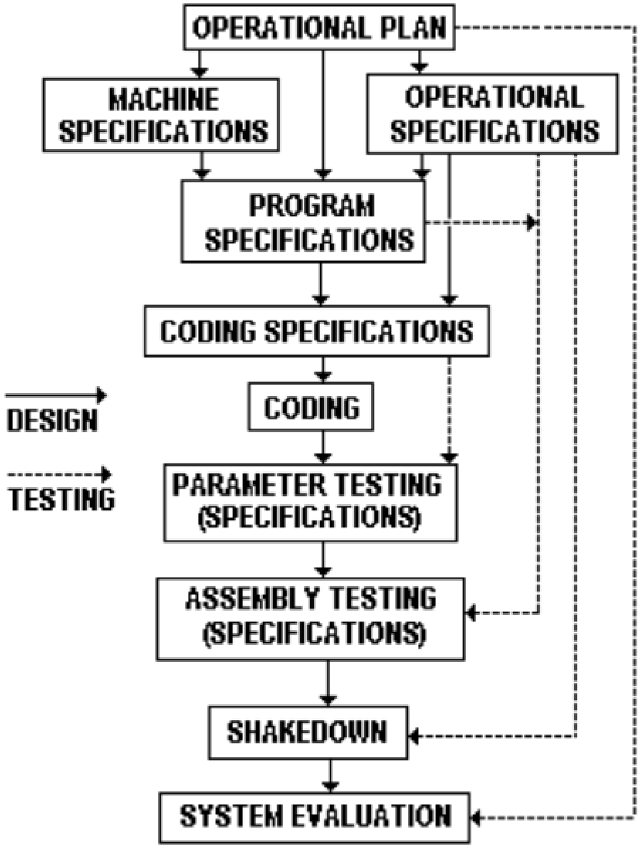
Source:
http://www.history.navy.mil/photos/pers-us/uspers-h/g-hoppr.htm

1956

H. D. Benington: Production of Large Computer Programs. Proceedings of Symposium On Advanced Computer Programs for Digital Computers, June 1956

»The paper is adapted from a presentation at a symposium on advanced programming methods for digital computers sponsored by the Navy Mathematical Computing Advisory Panel and the Office of Naval Research in June 1956. The author describes the techniques used to produce the programs for the Semi-Automatic Ground Environment (SAGE) system.«

»We find that large programs can now be produced; unfortunately, they are difficult to test and document.«



TESTING PRINCIPLES – INTEGRATION AND SYSTEM TEST

Stepwise integration test

»As parameter testing of component subprograms is completed, the system program is gradually assembled and tested using first simulated inputs and then live data.«

System tests in target environment

»When the completed program has been assembled, it is tested in its operational environment during shakedown.«

Acceptance test as final step

»At the completion of this phase, the program is ready for operation and evaluation.«

TESTING PRINCIPLES – CRITICALITY AND EFFORTS

Software as system part

»When the program is delivered for operation, its performance must be highly reliable because the control system is a critical part of a much larger environment of men and machines.«

Test efforts ~38% (without system test!)

No “complete tests” – testing is sampling

»It is debatable whether a program of 100,000 instructions can ever be thoroughly tested - that is, whether the program can be shown to satisfy its specifications under all operating conditions. Considering the size and complexity of a system program, it is certain that the program will never be subjected to all possible input conditions during its lifetime.«

»For this reason, one must accept the fact that testing will be sampling only.«

Testing to be improved

»On the other hand, many sad experiences have shown that the program-testing effort is seldom adequate.«

SELECTED HISTORICAL RESULTS BY JORIS MEERTS

1949	**On Checking a Large Routine (Turing)**	In the conference paper On Checking a Large Routine Alan M. Turing proposes an answer to the question how one can check a routine in the sense of making sure that it is right.
1951	**Total Quality Control (Feigenbaum)**	In his famous book 'Total Quality Control' Armand Vallin Feigenbaum defines quality as a customer determination. Quality depends on the perspective of the customer. The product should satisfy the customer in both actual and expected needs. There is a company-wide responsibility for quality.
1957	**Program testing vs debugging (Baker)**	Charles L. Baker (RAND Corporation) distinguishes program testing from debugging in his review of the book Digital Computer Programming by Dan McCracken. The review is published in the journal Mathematical Tables and Other Aids to Computation.
1958	**First software test team (Weinberg)**	The first test team is formed by Gerald M. Weinberg, working as manager of Operating Systems Development for the Project Mercury. Project Mercury is the first human spaceflight program of the United States.
1967	**Evaluation of the Functional Testing of Control Programs**	In the IBM white paper Evaluation of the Functional Testing of Control Programs William Elmendorf calls for a disciplined approach to software testing.
1968	**NATO report mentions Software Quality Assurance**	During the Software Engineering conference sponsored by the NATO Science Committee (7th to 11th October 1968) among other things quality assurance for software production is one of the topics. The report of the conference includes the working paper Checklist for planning software system production by Robert W. Bemer. This paper contains a chapter on quality assurance. One of the questions in the checklist is 'Is the product tested to ensure that it is the most useful for the customer in addition to matching functional specifications?'.

http://www.testingreferences.com/testinghistory.php

SELECTED HISTORICAL RESULTS BY JORIS MEERTS

1969	**Testing shows the presence, not the absence of bugs**	Edsger Dijkstra's famous quote was reportedly first spoken on a conference by the NATO Science Committee, Rome, Italy, 27–31 October 1969.
1971	**Mutation testing (Lipton)**	In a class term paper titled Fault Diagnosis of Computer Programs Richard Lipton proposed the initial concepts of mutation. Mutation testing is a methodology for unit testing in which small parts of the code are changed. This is done, for example, in order to test the quality of the unit tests.
1973	**Program Test Methods (Hetzel)**	The Chapel Hill Symposium, organized by the University of North Carolina and held on June 21-23 1972, leads to publication of the book Program Test Methods edited by William Hetzel. The book contains the edited papers of the symposium as well as a large annotated bibliography. The book focuses on the problems in testing and validation.
1975	**Toward a Theory of Test Data Selection (Goodenough, Gerhart)**	The paper by John B. Goodenough and Susan L. Gerhart discusses formal proof methods and the limitations of structure-based testing. It also outlines the use of decision tables.
1976	**Cyclomatic Complexity (McCabe)**	Thomas J. McCabe introduces cyclomatic complexity as a software metric for the complexity of a program in his IEEE paper A Complexity Measure. McCabe also introduces basic path testing as a white box test technique.
1976	**Software Reliability: Principles and Practices (Myers)**	In his book Software Reliability: Principles and Practices Glenford Myers discusses software testing among other things. He mentions, for example, that 'The goal of the testers is to make the program fail'.

SELECTED HISTORICAL RESULTS BY JORIS MEERTS

1976	**Cost-of-change curve (Boehm)**	In his paper Software Engineering, published in the December 1976 issue of IEEE Transactions , Barry Boehm publishes his cost-of-change curve. The curve essentially shows that the cost of changing the software (fixing a software defect) rises exponentially in time. Boehm uses data from his work at TRW and other sources such as GTE, IBM and Bell Laboratories.
1982	**SQS founded in Germany**	The German company Software Quality Systems (SQS) is founded Heinz Bons and Rudolf van Megen. It is one of the leading software testing organisations in Europe.
1983	**IEEE 829 published**	The first version of the IEEE 829 Standard for Software Test Documentation is published in 1983. The standard specifies the form of a set of documents for use in eight defined stages of software testing.
1984	**SEI founded**	The Carnegie Mellon Software Engineering Institute (SEI) is established by the U.S. Department of Defense. In its own words "the SEI advances software engineering and related disciplines to ensure the development and operation of systems with predictable and improved cost, schedule, and quality."
1986	**V-model published (Rook)**	In the article Controlling Software Projects, published in the IEEE Software Engineering Journal, Paul E. Rook introduces the V-model. Rook works for GEC Software Ltd. in London at that time. The model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
1987	**Test, then code**	Motto on the lapel pin of SQE as worn during the Fourth International Conference on Software Testing, Washington DC.

SELECTED HISTORICAL RESULTS BY JORIS MEERTS

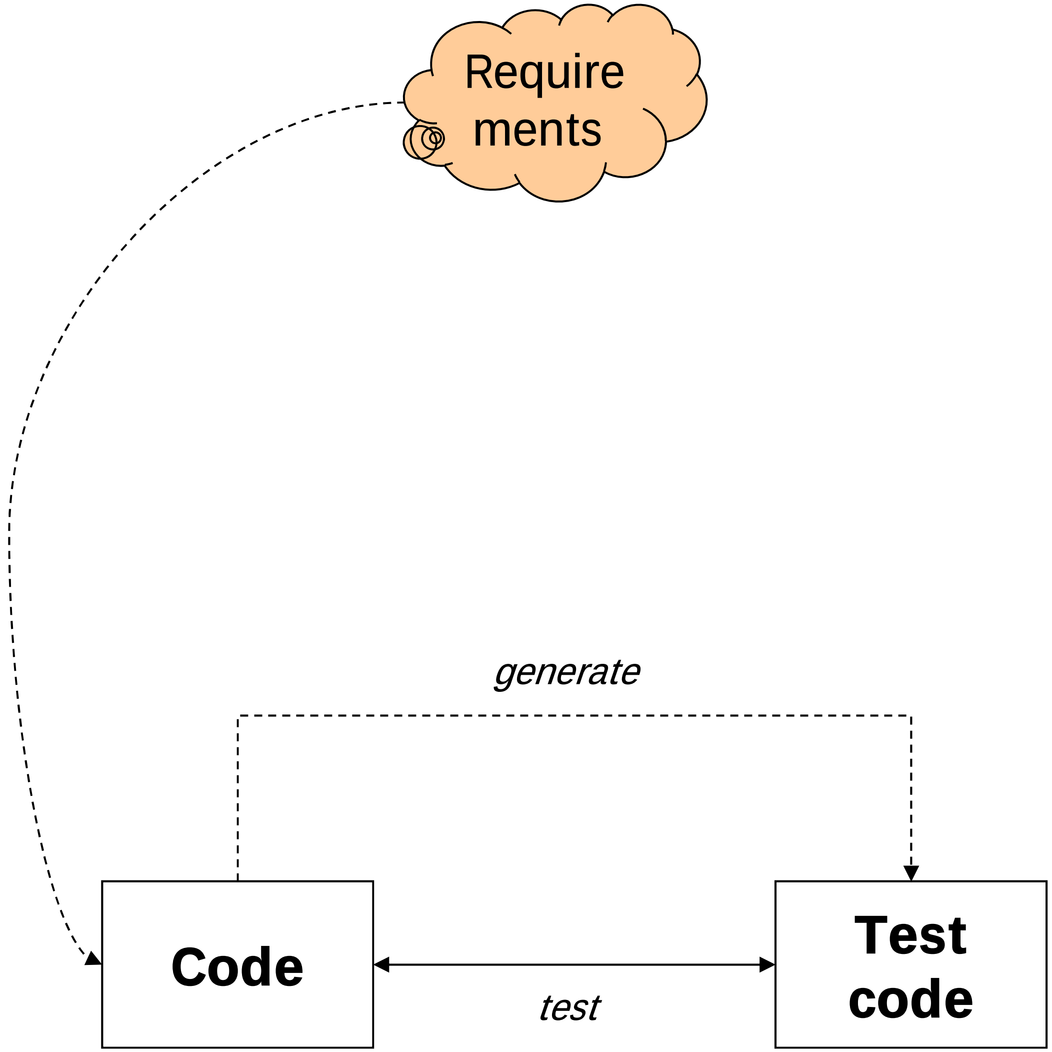
1987	**Software reliability (Musa)**	The seminal work Software Reliability: Measurement, Prediction, Application is published by John D. Musa, Anthony Iannino, and Kazuhira Okumoto. Software reliability has become a key part in software quality.
1989	**SIGIST founded**	The British Specialist Interest Group in Software Testing (SIGIST) is founded in 1989 by Geoff Quentin. Its first meeting is held at Imperial College in London. The meeting, during which four presentations (on risks, standards and reliability) are given, is attended by 29 people. The aim of the group is to share problems, successes and failures in testing, share techniques and share ideas of tools to support testing.
1991	**ISO 9126 published**	ISO/IEC 9126 Software engineering — Product quality is an international standard for the evaluation of software quality. Its quality model splits up quality into six characteristics.
1992	**First version of TTCN**	The first version of the Testing and Test Control Notation (TTCN) - originally meaning Tree and Tabular Combined Notation - is published by the ETSI Centre for Testing and Interoperability. The language is launched as a specification of abstract test suites for conformance testing of International Telecommunications Union protocols. It is now promoted as a general purpose test language for distributed communicating systems.
1993	**W-model introduced (Herzlich)**	In his presentation The Politics of Testing Paul Herzlich introduces the W-model. The model attempts to address shortcomings in the V-Model. Herzlich holds the presentation during the first EuroSTAR conference in London.

SELECTED HISTORICAL RESULTS BY JORIS MEERTS

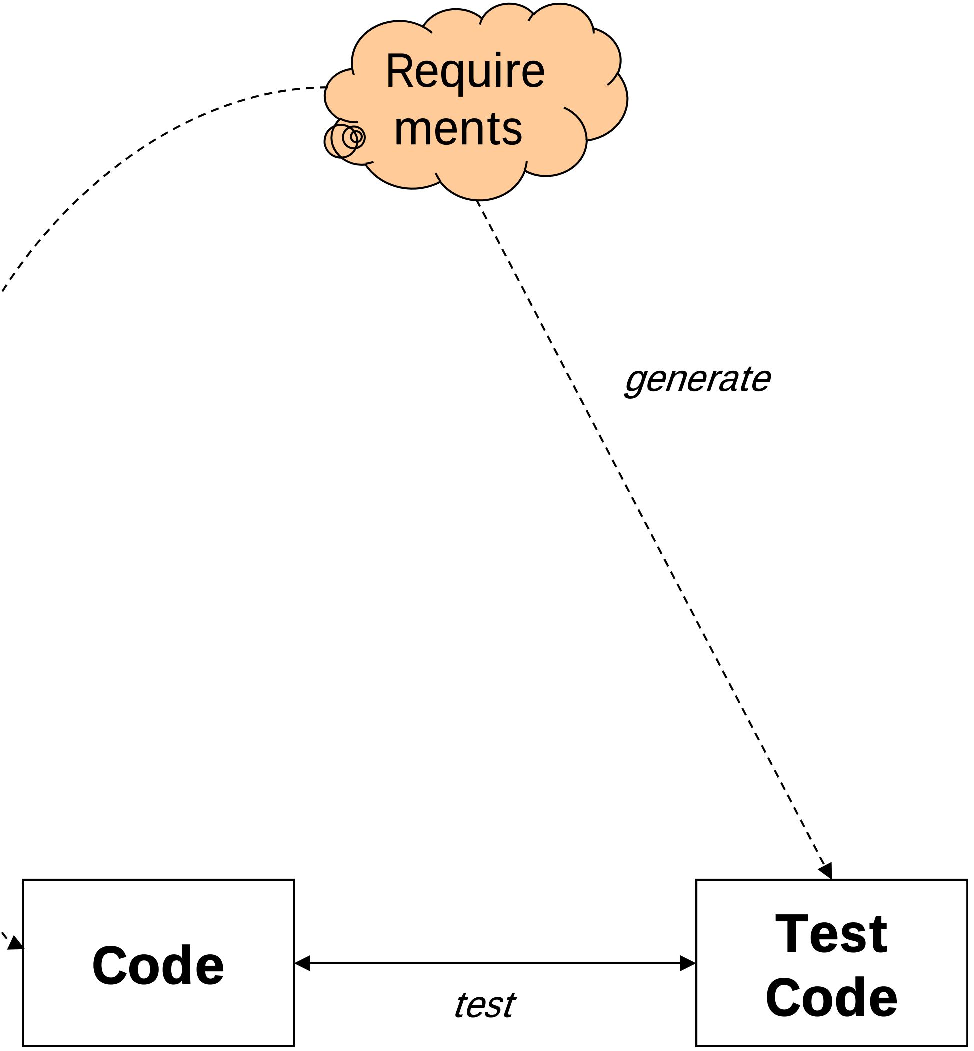
1994	**First Chaos report (Standish Group)**	The Standish Group starts the Chaos report, a continuing study to identify the scope of software project successes and failures, the major factors that cause software projects to fail, and the key ingredients that can reduce software project failures.
1996	**TMM developed**	The Testing Maturity Model is developed at the Illinois Institute of Technology.
2000	***TTCN-3 developed***	*The Testing and Test Control Notation is published by ETSI*
2004	***UTP developed***	*The UML Testing Profile is published by OMG*
2010	***TTCN-3 embedded developed***	*Real-time and performance extensions for TTCN-3 published by ETSI*
2013	***TTCN-3 fuzzing developed***	*Fuzzing extensions for TTCN-3 published by ETSI*

Fraunhofer
FOKUS

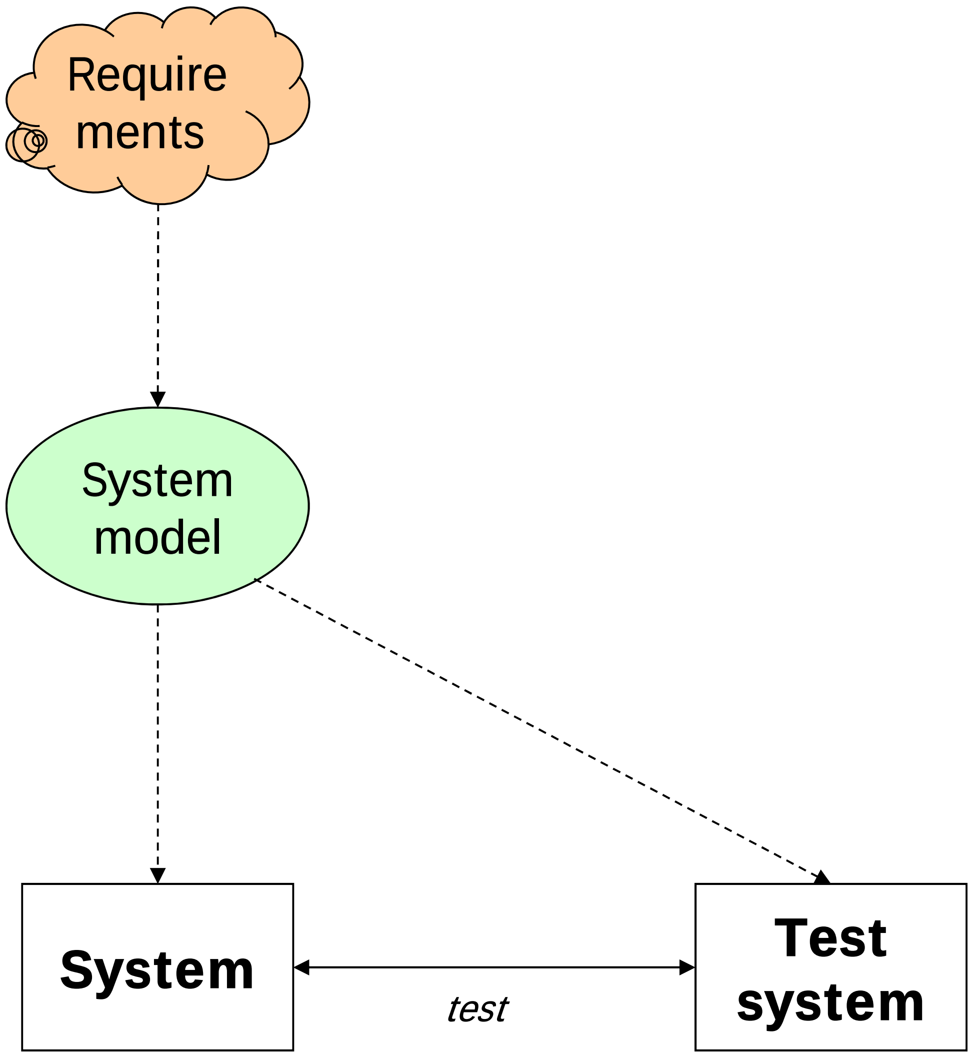
TESTING TECHNIQUES – STRUCTURAL TESTING



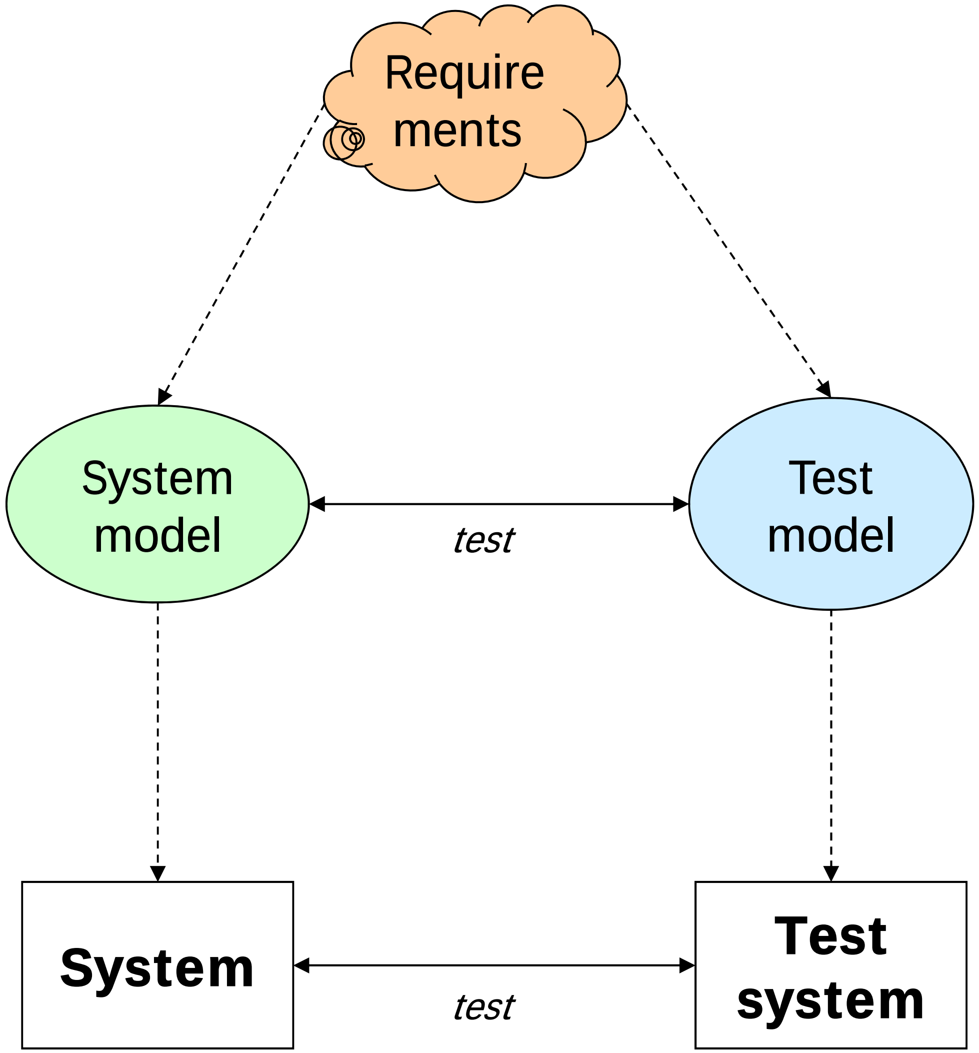
TESTING TECHNIQUES – FUNCTIONAL TESTING



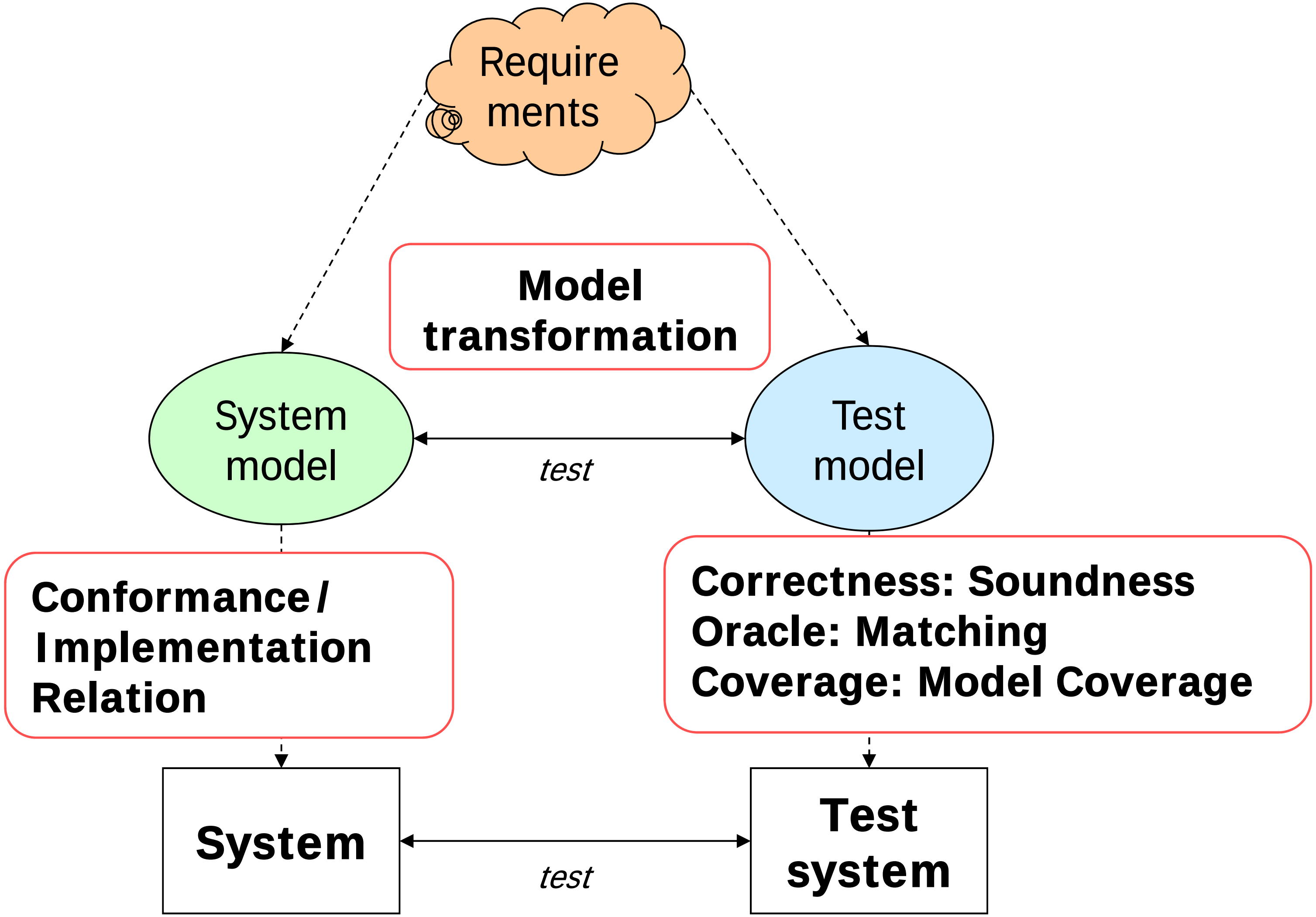
TESTING TECHNIQUES – MODEL-BASED TESTING 1.0



TESTING TECHNIQUES – MODEL-BASED TESTING 2.0



TESTING TECHNIQUES – MODEL-BASED TESTING 2.0



Requirements

Model transformation

System model

Test model

test

Conformance / Implementation Relation

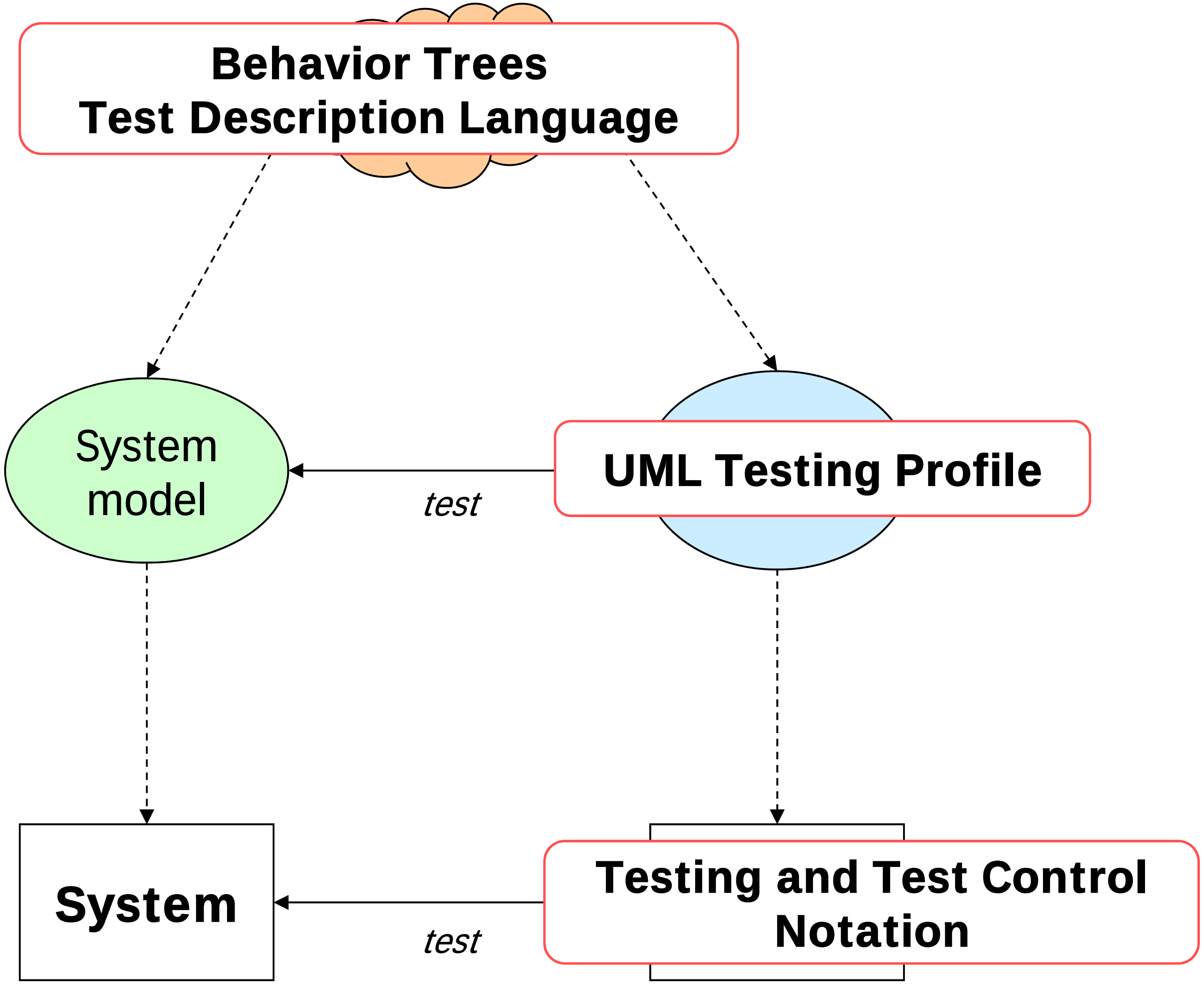
Correctness: Soundness
Oracle: Matching
Coverage: Model Coverage

System

Test system

test

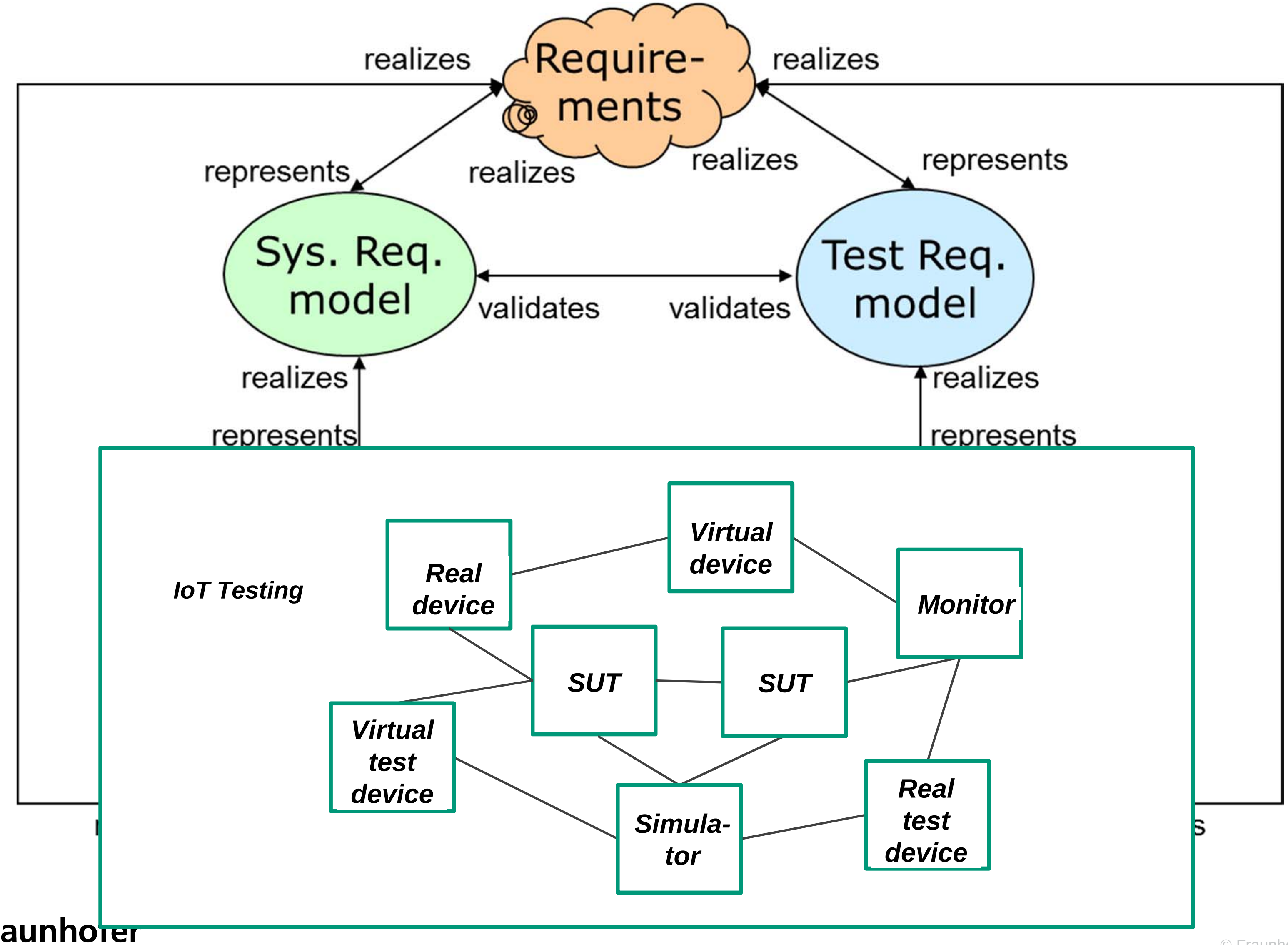
TESTING TECHNIQUES – TEST TECHNOLOGIES



OUTLINE

1. Status software quality
2. Some history
3. Some future perspectives

TESTING INTERNET OF THINGS SOLUTIONS



PROFESSIONAL TESTING

	Main modules	Supplementary modules
CTEL	EL-ITP EL-TM	
CTAL	AL-TM AL-TA AL-TTA	Security Test Autom **Industrial IoT ?!** **Consumer IoT ?!** **Public IoT ?!**
CTFL	Software Foundation **Embedded Systems Foundation ?!**	Agile Auto motive MBT Usability Mobile IoT ?!

ISTQB

VOR 10 JAHREN: AUSGEWÄHLTE PROGNOSEN – 2010

- »Vielleicht (hoffentlich!) ist erstmals ein Mensch ins Gefängnis gekommen, weil er schlampige Software gemacht oder zugelassen hat. Aber wahrscheinlich geht auch das nicht so schnell.«

Prof. Dr. Jochen Ludewig, Uni Stuttgart

- »Wir werden uns an Software-Tote gewöhnen wie wir uns an die Verkehrstoten gewöhnt haben. Einzelne, besonders verantwortungslose Entwickler werden zwar bestraft werden (so wie im Straßenverkehr ein Raser, der Menschen auf dem Gewissen hat), aber allgemeine, durchgreifende (und wirksame!) Maßnahmen werden bei der Software ebenso wenig ergriffen werden, wie im Straßenverkehr, da zu aufwendig und zu unbequem.«

Prof. Dr. Martin Glinz, Uni Zürich, Schweiz

Andreas Spillner: Von der Motte zur Roboter-Revolte Geschichte, Gegenwart und Zukunft des Prüfens und Testens von Softwaresysteme, 11. Juli 2006, GI-Regionalgruppe Bremen/Oldenburg

VOR 10 JAHREN: AUSGEWÄHLTE PROGNOSEN – 2020

- »In 15 Jahren wird die Zahl der Leute, die eine eigentliche Informatik-Ausbildung haben, gewachsen sein, von ca. 20 % vielleicht auf 40 %, höchstens 50 % derer, die von ihrer Tätigkeit her eigentlich eine solche Ausbildung brauchen. Durch eine veränderte Rechtssprechung wird es eine schärfere Haftung geben, die eine bessere QS unvermeidlich macht. Darum wird es in 15 Jahren weit mehr als heute definierte Prüfprozesse geben, teilweise rein bürokratisch, teilweise auch mit praktischen Folgen, also mit sinnvollen Reviews und systematischen Tests. Und natürlich wird der Markt mehr Werkzeuge anbieten, die das unterstützen.«

  Prof. Dr. Jochen Ludewig, Uni Stuttgart

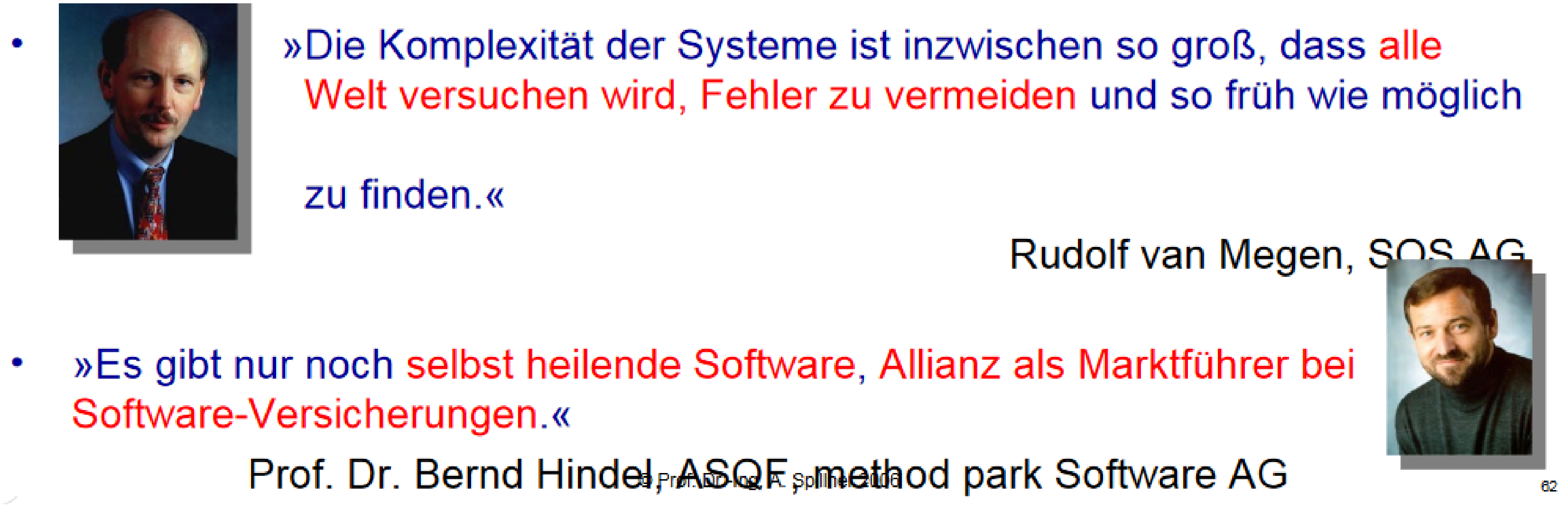
- »Es gibt ein weltweit anerkanntes Curriculum in Sachen QS, und viele der ehemals analytischen Verfahren sind nun bereits konstruktiv berücksichtigt und somit aus dem QS-Kanon "verschwunden". QS betrachtet zunehmend die "politisch-sozialen" Auswirkungen und Möglichkeiten der Systeme.«

  Prof. Dr. Mario Winter, FH Köln

- »Im Gefolge fortgeschrittener Akzeptanz von deklarativen Sprachen - insbesondere ausführbaren Spezifikationssprachen - weite Verbreitung von High-Level Testing.«

  Prof. Dr. Klaus Peter Löhr, FU Berlin

VOR 10 JAHREN: AUSGEWÄHLTE PROGNOSEN – 2035



- »Die Komplexität der Systeme ist inzwischen so groß, dass alle Welt versuchen wird, Fehler zu vermeiden und so früh wie möglich zu finden.«

  Rudolf van Megen, SQS AG

- »Es gibt nur noch selbst heilende Software, Allianz als Marktführer bei Software-Versicherungen.«

  Prof. Dr. Bernd Hindel, ASQF, method park Software AG

© Prof. Dr.-Ing. A. Spillner 2006

62

- »Wir haben endlich den SSTV - den Software und System Test Verein, der unabhängig, systematisch und automatisiert Software und Software-basierte Systeme welcher Art auch immer, prüft, Gütesiegel vergibt und darüber Qualitätsstandards durchsetzt.«

  Prof. Dr. Ina Schieferdecker, Fraunhofer, Fokus

MY CONJECTURES

- Testing – both static and dynamic – continues to be the most important instrument in assuring quality of software-based critical systems as approaches like correctness by construction, security by design, etc. have their limits

- Testing needs to be extended into the production environment with models at runtime, online testing methods, and alike

- Along digitized networking, Industry 4.0, Smart Cities, etc., the SSTV (Software und System Test Verein) is more and more needed and hopefully established before 2035

- In future, software and test software may be merged into self-testing software, which can sanity check itself and check its environment

DISCUSSION