Panel Discussion

Five Decades of Software Crisis: Does the Quality of Software Reflect the Societal Significance of Software?

Hans-Joachim Hof
hof@insi.science
http://insi.science

INSicherheit – Ingolstadt Research Group Applied IT Security,
CARISSMA – Center of Automotive Research
Technical University of Ingolstadt

Prof. Dr.-Ing. Hans-Joachim Hof

IN Sicherheit
Ingolstädter Forschungsgruppe
Angewandte IT-Sicherheit

CARISSMA
Automotive Safety Research

IN Sicherheit
Ingolstädter Forschungsgruppe
Angewandte IT-Sicherheit

MuSe
Munich IT Security Research Group

Technische Hochschule Ingolstadt

HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

acm
German Chapter

GI

bizcon
future information architecture

Five Decades of Software Crisis: 1968

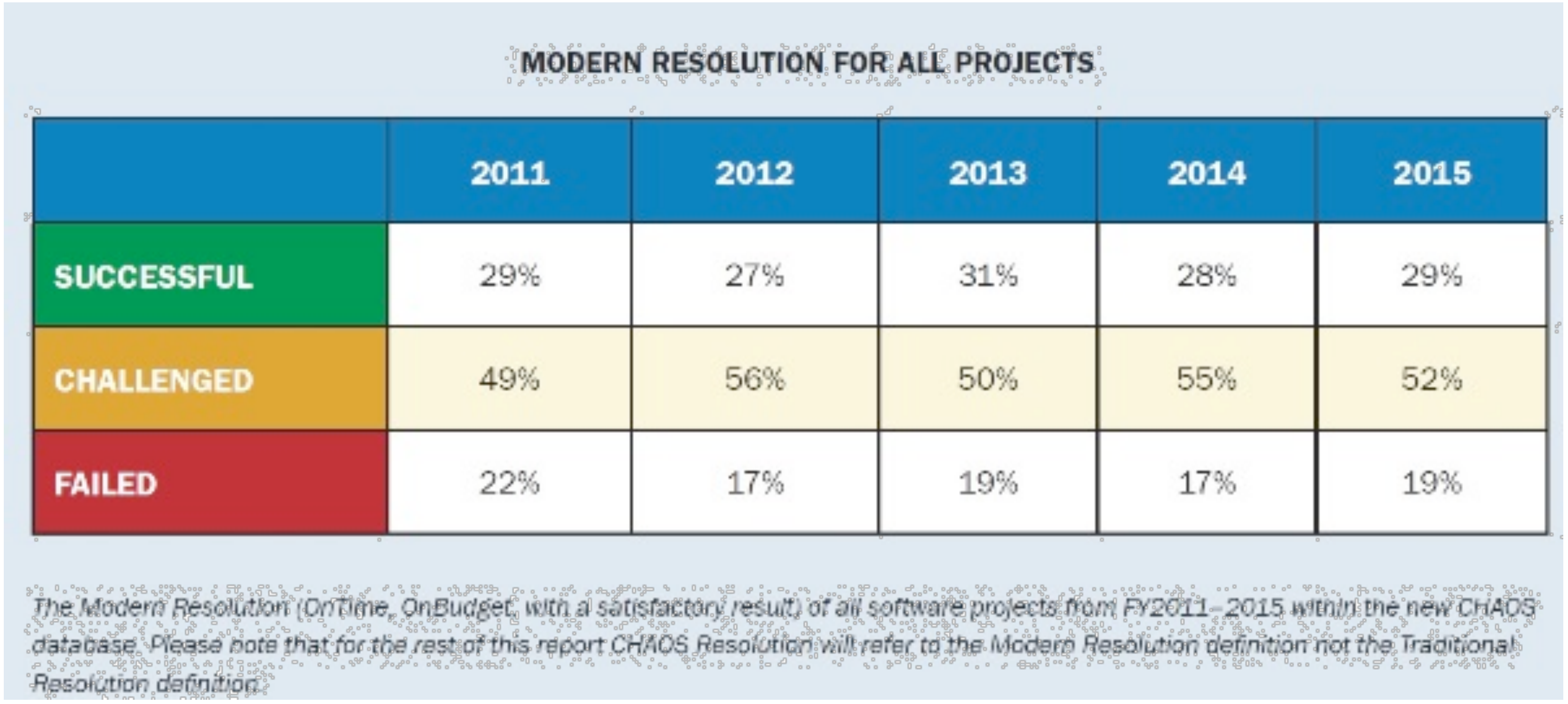
- Term „software crisis“ was coined in 1968 at the first NATO Software Engineering Conference in Bavaria, Germany

- Software crisis (1968)
    - Software with low quality
    - Software projects over budget
    - Software projects late

- Friedrich Ludwig Bauer: „“There is so much tinkering with software ... what we need is software engineering”

Five Decades of Software Crisis: Today

- Standish Group CHAOS Report 2015 (time/costs/completeness):



MODERN RESOLUTION FOR ALL PROJECTS

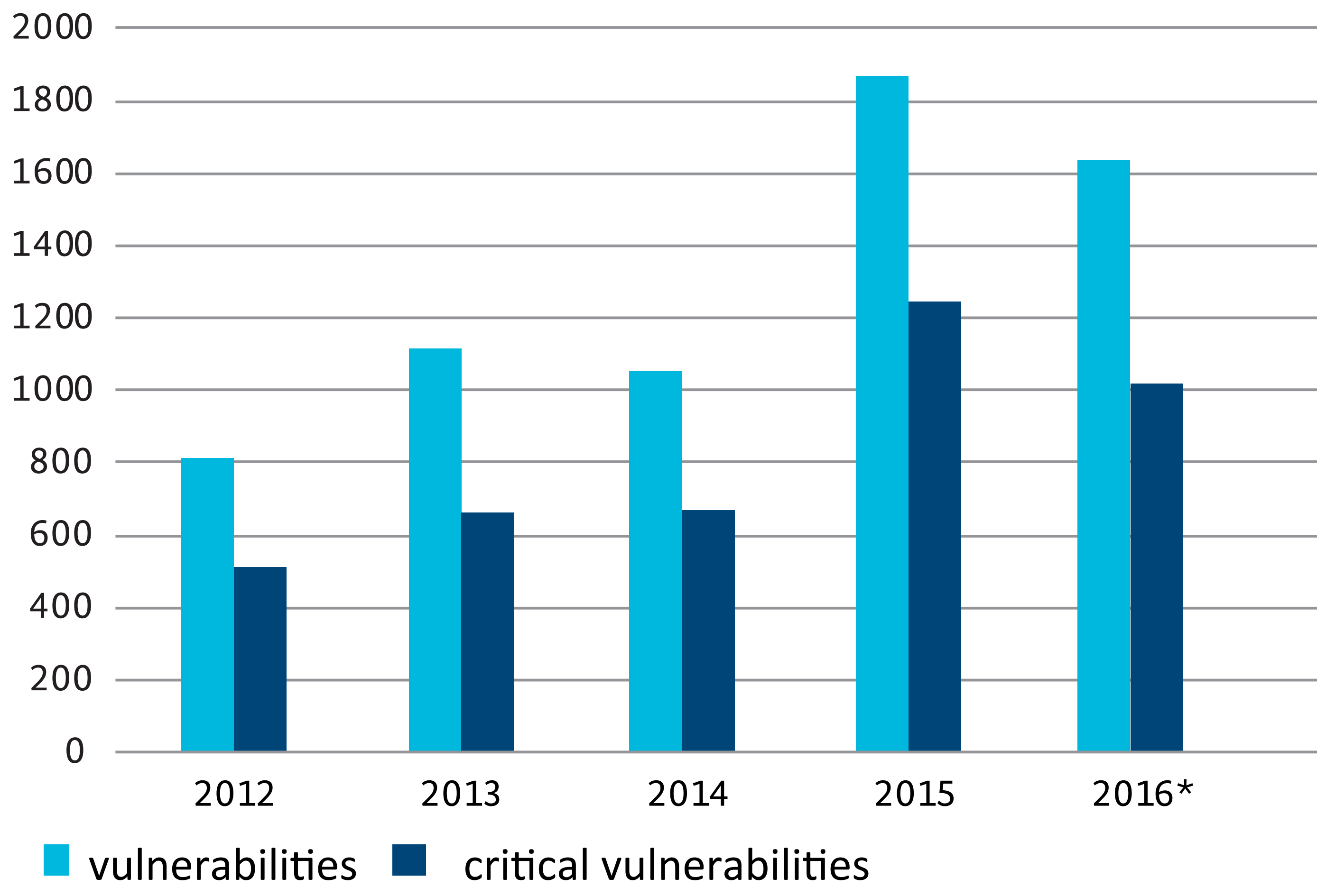
	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

- Successful = in time, in budget, all features
- Challenged = over time and/or over budget and/or reduced number of features
- Failed = Project stopped

Five Decades of Software Crisis: Today

- Quality? Quality aspect security as an (important) example
- Number of vulnerabilities in Top 10 of software products (most used):



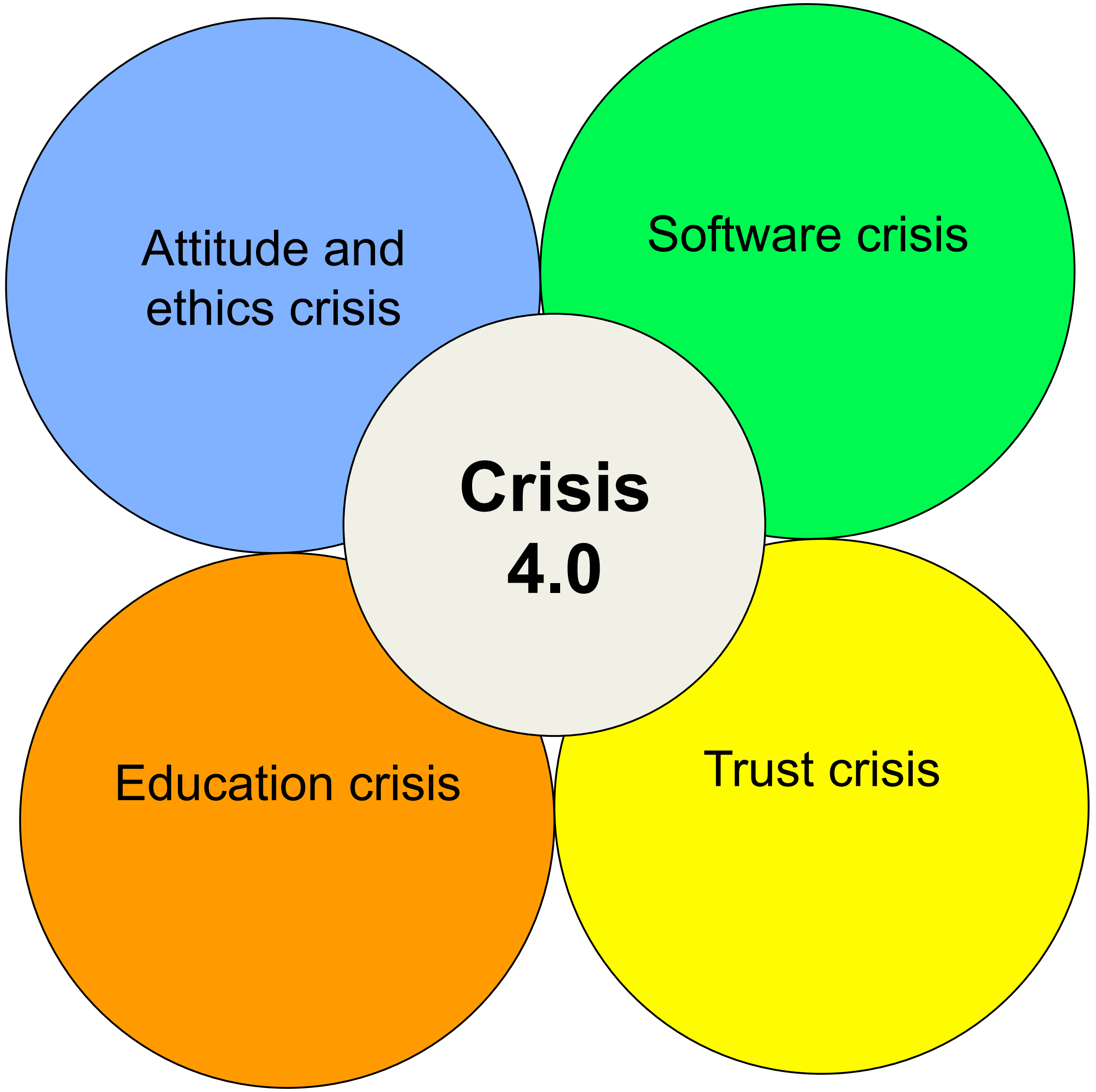
Software & Society

- Computers will do many jobs that humans do at the moment
- Computers will have more autonomy to act
- Computers will be used in any life aspect
  - E.g., autonomous driving
  - E.g., credit approval
  - E.g., crime prediction by parole boards
- Low software quality could have tremendous negative effects in all areas of life
- Somehow, people got used to low quality software...

Software & Society: Crisis 4.0

Ingolstädter Forschungsgruppe Angewandte IT-Sicherheit

INSicherheit



Panelists

Christopher Ireland
The Open University, UK
Standing on the shoulders of giants?

Bruce Sams
OPTIMAbit GmbH, Germany
Penetration Test: Tip of the Iceberg

Aspen Olmsted
College of Charleston, USA
Escalating non-functional requirements in the SDLC to increase software quality

Mudasser Wyne,
National University, USA
Are we preparing graduates for Industry; Ready to hire?

Thomas Schaberreiter
University of Vienna, Austria
Building blocks for a better software security culture are there, the challenge will be to better integrate them

Aspen Olmsted

ASSISTANT PROFESSOR AND GRADUATE PROGRAM DIRECTOR

COLLEGE OF CHARLESTON

Does the quality of software reflect the societal significance of software?

No – We need to escalate non-functional requirements in the SDLC to increase software quality

How do we model Non-Functional Requirement Implementation?

- Example Non-Functional Requirements
  - Must continue to record data for up to 24 hours of network partition
  - Must be able to support 10,000 concurrent devices
  - Must guarantee that clients are who they say they are

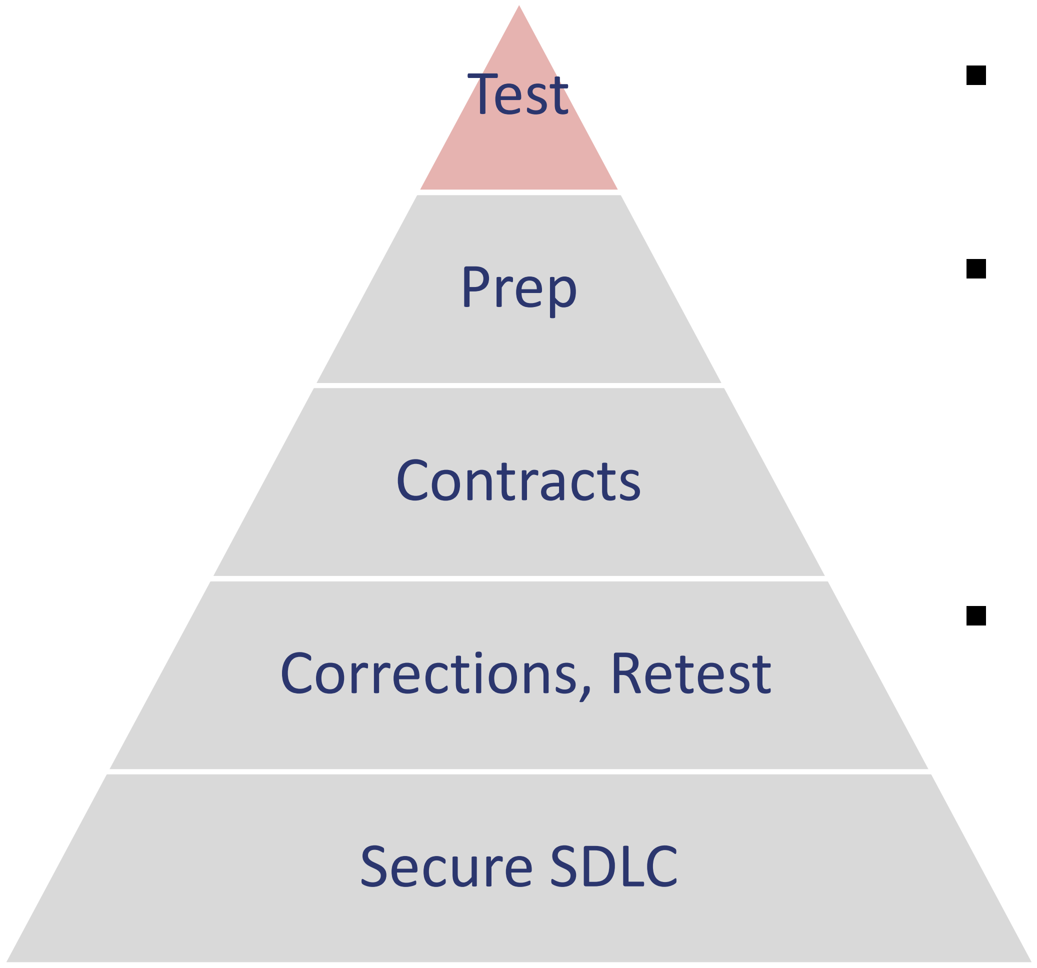
Are model models organized for non-functional requirements

- Structural - Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.
  - Class diagrams vs. constraints
- Behavioral - Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.
  - Use cases vs. misuse cases

About OPTIMAbit GmbH

Focus	Customers	Credo
IT-Security for Applications & Infrastructures	Corporations Governments Defense	Vendor neutral consulting of the highest quality

Pentests: Just the Tip of the Iceberg



- The Pentest itself is only a small part of the process.
- Do not underestimate the organization necessary for a successful test and followup.
- Focus on the bottom of the pyramid!

OPTiMA..bit
business information technology gmbh

Thanks for listening!

OPTIMAbit GmbH

Kaflerstr. 4

82141 München

Tel.: +49 89 8895 1819-0

bruce.sams@optimabit.com

www.optimabit.com

FASSI / AFIN Panel - Rome 2017

Five Decades of Software Crisis: Standing on the shoulders of giants?

Chris Ireland
The Open University, UK

Software Crisis (1968)…

The causes of the software crisis were linked to the overall **complexity** of hardware and the software development **process**.

The crisis manifested itself in several ways:

- Projects running **over-budget**
- Projects running **over-time**
- Software was very **inefficient**
- Software was of **low quality**
- Software often **did not meet requirements**
- Projects were **unmanageable**
- Code was **difficult to maintain**
- Software was **never delivered**

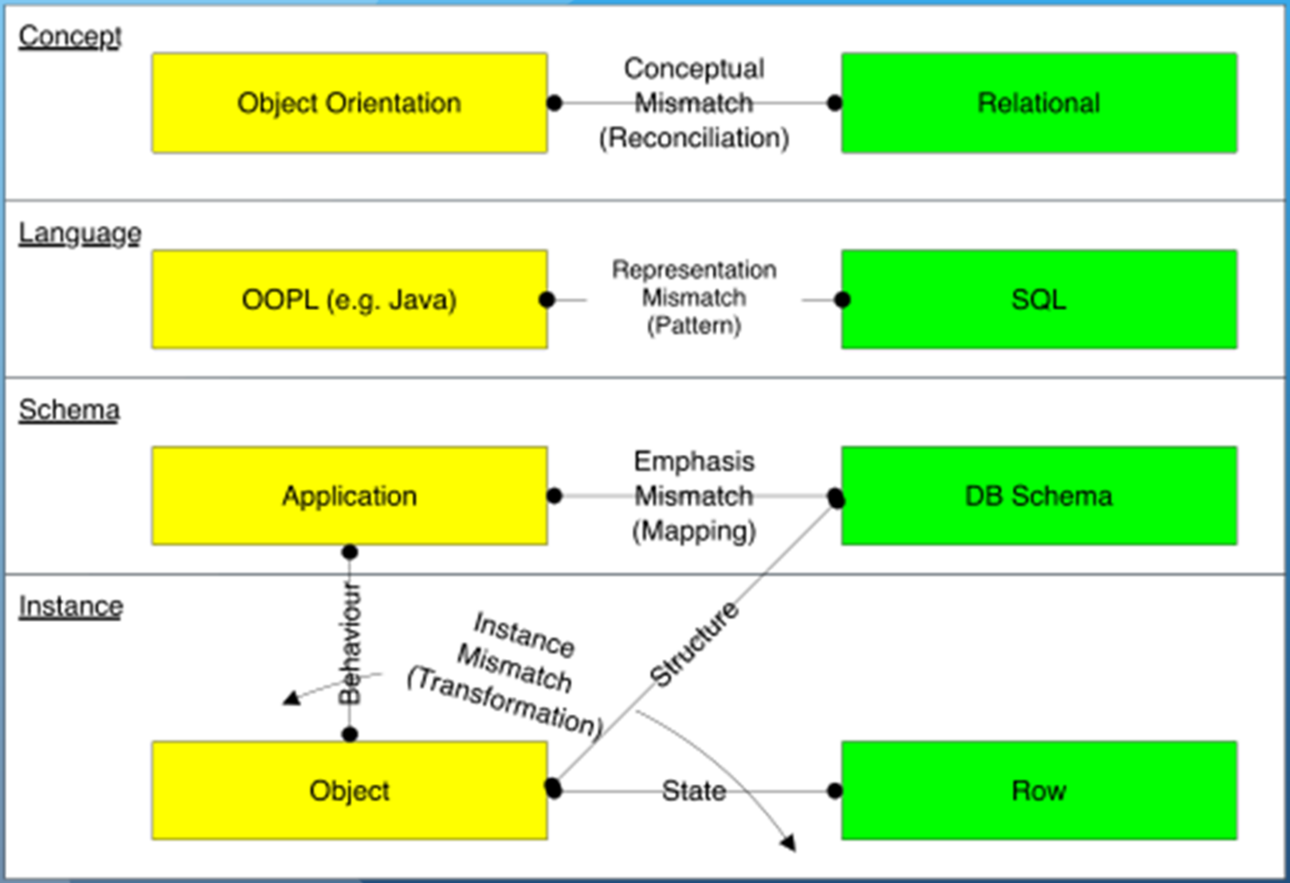
[Wikipedia]

Software Integration

Two Perspectives on the crisis:

- Macro
    - An Industry-level perspective
    - An Organisation-level perspective
- Micro
    - A system-level perspective
    - A program-level perspective

A Micro Perspective (e.g. ORIM)


Concept
Object Orientation
Conceptual Mismatch (Reconciliation)
Relational
Language
OOPL (e.g. Java)
Representation Mismatch (Pattern)
SQL
Schema
Application
Emphasis Mismatch (Mapping)
DB Schema
Instance
Behaviour
Instance Mismatch (Transformation)
Structure
Object
State
Row

Ireland, C., Bowers, D., Newton, M., Waugh, K.:
A Classification of Object-Relational Impedance Mismatch.
Proc. DBKDA 2009, Vol. 1. IEEE Computer Society, Cancun, Mexico (2009)
p36-43

Software Crisis (2017)…

- Software remains intangible & there are many **choices** (efficiency)
- Ever more **complex models** & SDLC (budget & time, delivery)
- Ever more **interconnected** (manageable?)
- There is an expectation that it should just work. Development projects are about **managing expectations** (delivery, functionality, efficiency)
- Software is embedded in many aspects of daily life. Difficult for one person or even a team to **comprehend all the possibilities** and use cases/scenarios (requirements)
- Need **macro** and **micro** solutions!

Challenges in Developing Secure Software

Panel Discussion, SECURWARE 2017

Rome, September 12th 2017

Stefan Schauer, AIT

PERSONAL INTRODUCTION

- **Affiliation**
  - AIT Austrian Institute of Technology
  - Center for Digital Safety & Security
  - Secure Communication Technologies Group

- **Scientific Background**
  - Master in computer science (IT security)
  - PhD in theoretical physics (quantum cryptography)

- **Current Research**
  - Risk and security management for critical infrastructures (CIs)
  - CI interdependencies and assessment of cascading effects
  - Game theoretic approaches for risk management

IMPACT OF INSECURE SOFTWARE

- Security needs to be an integral part of software development
  - IT systems (and software) influences our life in multiple different ways (communication, transport, government, personal data, …)
  - In many fields security is only a by-product or add-on to the developed IT systems
  - Several approaches towards Security by Design are present and need to be integrated from the start

- Flaws and errors in software open doors for attacks
  - Software vulnerabilities are mostly due to error-prone implementation
  - Flaws in software can be used to create unexpected and malicious behavior

IMPACT OF INSECURE SOFTWARE

- IT systems are misused by malicious parties
  - Botnets are created and DDoS attacks are using thousands of IoT devices

- Systems get hacked and encrypted
  - Crypto ransomware like WannaCry and Petya creates data loss and stops the operation of several important services

- Attackers get control of highly-relevant systems or information
  - Electrical power system gets shut down by attackers in Ukraine

The price for developing secure software might be small,
the potential impacts of error-prone systems can be severe!

LET‘S START THE DISCUSSION

Dr. Stefan Schauer
Center for Digital Safety & Security
Austrian Institute of Technology
Klagenfurt, Austria
stefan.schauer@ait.ac.at

PANEL FASSI/AFIN

The Ninth International Conference on Advances in Future Internet (AFIN 2017)
September 12, 2017
Rome, Italy

Are We Preparing Graduates for Industry: Ready to Hire?

Mudasser F. Wyne, Ph.D

Professor of Computer Science,
Chair: Department of Computer Science and Information Systems,
Program Director: MS in Computer Science,
School of Engineering and Computing,
National University,
San Diego, USA

GOALS

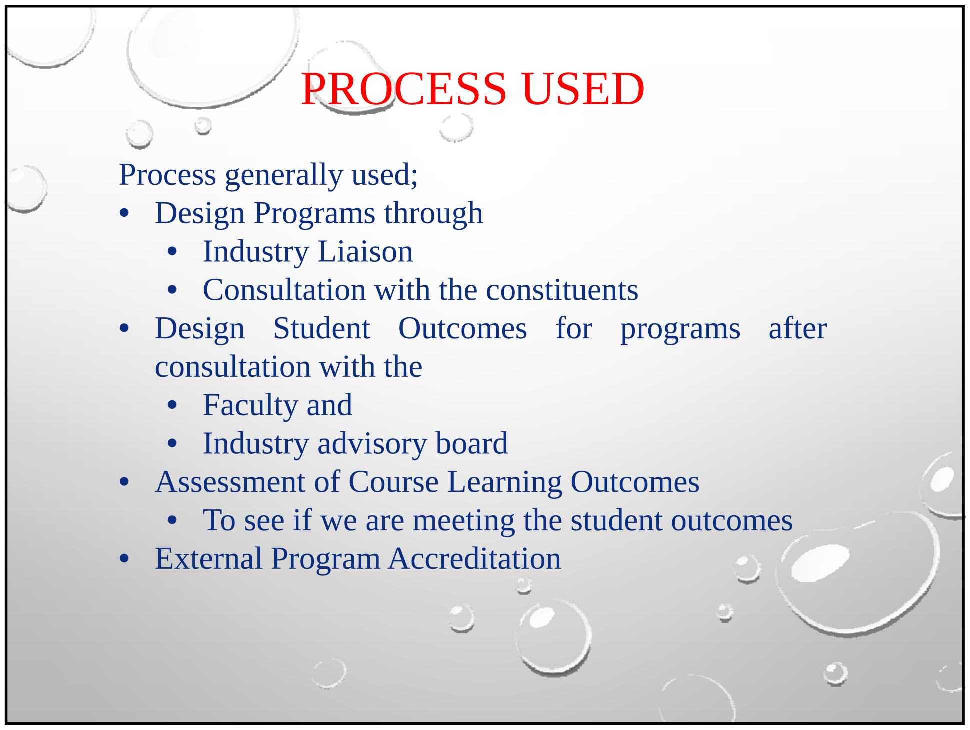
GOALS

To ensure graduates from the programs are employable.

- University and College programs should be aligned with
  - The latest technological trends as well as with the market needs.

- Study programs also need to focus on raising the level of professionalism of the students
  - By exposing them to best practices used by top organizations and companies in the relevant market.

PROCESS

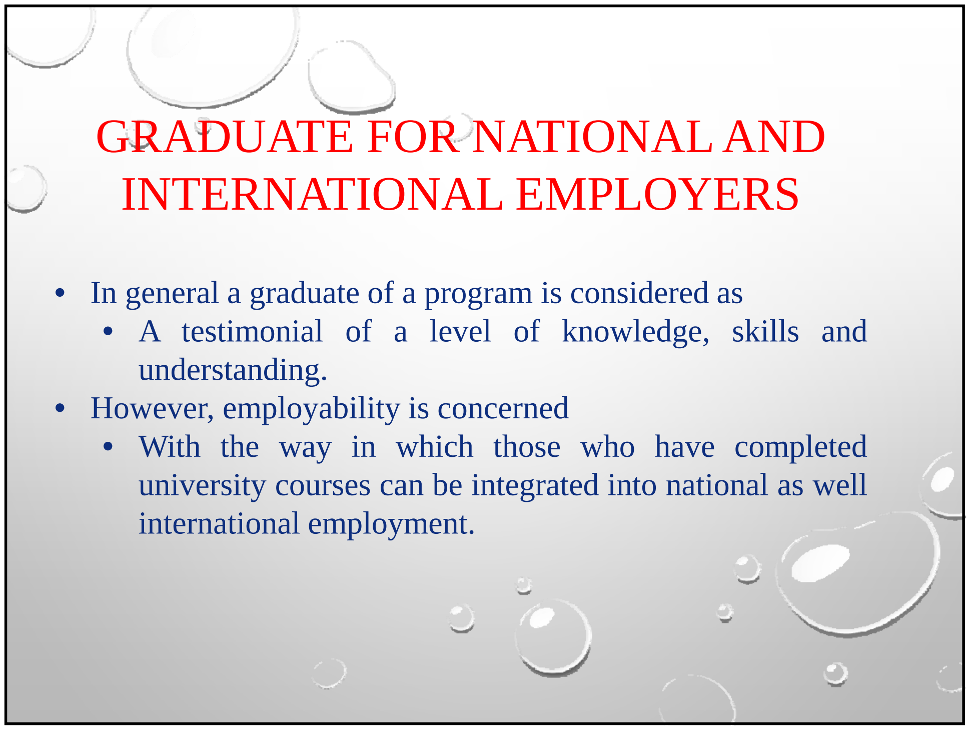


PROCESS USED

Process generally used;

- Design Programs through
  - Industry Liaison
  - Consultation with the constituents
- Design Student Outcomes for programs after consultation with the
  - Faculty and
  - Industry advisory board
- Assessment of Course Learning Outcomes
  - To see if we are meeting the student outcomes
- External Program Accreditation

CHALLENGES



GRADUATE FOR NATIONAL AND INTERNATIONAL EMPLOYERS

- In general a graduate of a program is considered as
  - A testimonial of a level of knowledge, skills and understanding.
- However, employability is concerned
  - With the way in which those who have completed university courses can be integrated into national as well international employment.

WEAKNESSES

Following are some of the weaknesses that have been reported by various researchers;

- Weak communication and collaboration with the market and the employers.
- Unsatisfactory practical and applied skills for the students.
- Students have shallow knowledge about the role of innovation, creativity and entrepreneurship in the computing discipline.
- Lack of strong personal skills for the college graduates
- Students are not up to date with the latest advances and developments in the field of computing technologies

OPEN DISCUSSION

10

FASSI/AFIN Panel

Software quality: The security perspective

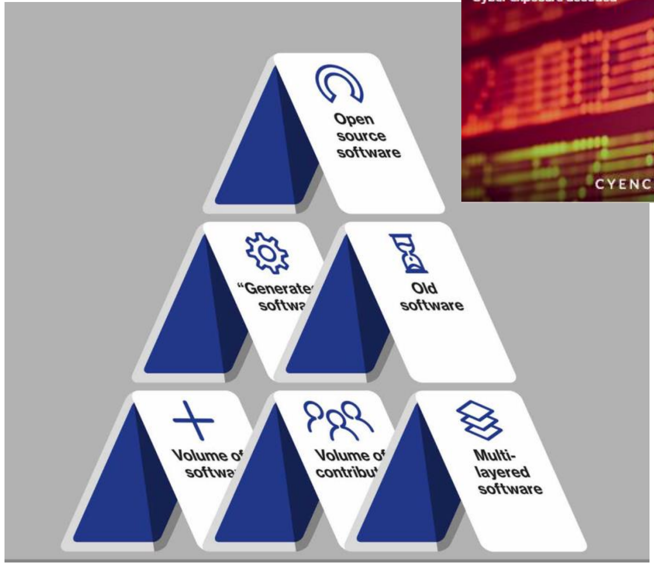
Thomas Schaberreiter

Rome, 12.9.2017

Introduction

Does the Quality of Software Reflect the Societal Significance of Software?

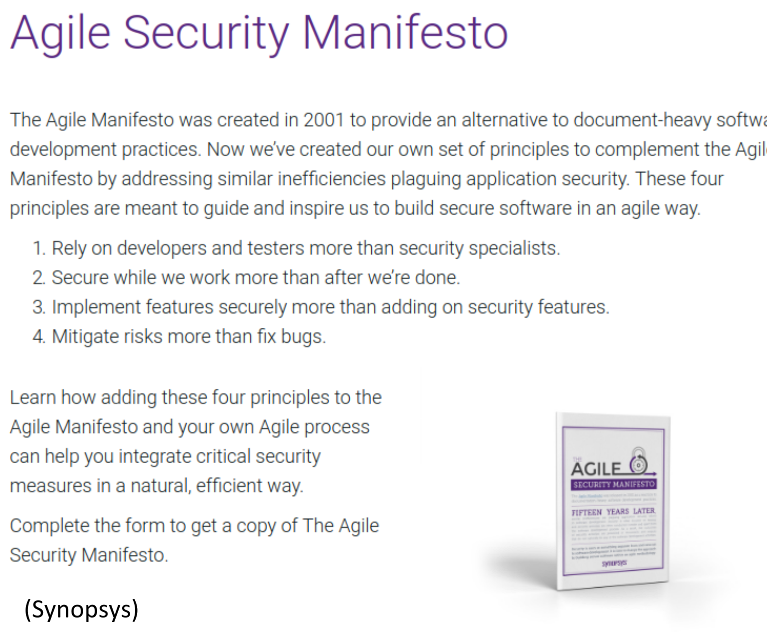
- Clear and simple answer: No.
- From the security perspective
  - Quality of software and software vulnerability are closely related
  - Mass software vulnerability losses per attack are estimated between 9.7 billion USD and 28.7 billion USD
- Is all hope lost?
  - Perceived significance of software quality has changed
  - Technological, Organizational, Governance, Economic



Software quality strategies

Technological

– Secure software development
  - Secure agile development
– Automated testing
  - Fuzz testing



Agile Security Manifesto

The Agile Manifesto was created in 2001 to provide an alternative to document-heavy software development practices. Now we've created our own set of principles to complement the Agile Manifesto by addressing similar inefficiencies plaguing application security. These four principles are meant to guide and inspire us to build secure software in an agile way.

1. Rely on developers and testers more than security specialists.
2. Secure while we work more than after we're done.
3. Implement features securely more than adding on security features.
4. Mitigate risks more than fix bugs.

Learn how adding these four principles to the Agile Manifesto and your own Agile process can help you integrate critical security measures in a natural, efficient way.

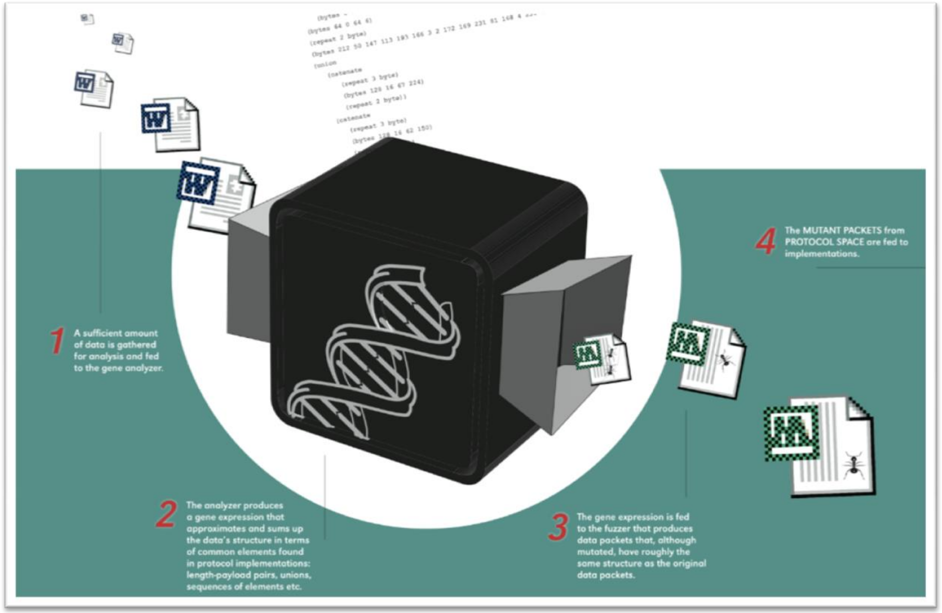
Complete the form to get a copy of The Agile Security Manifesto.

(Synopsys)

Software quality strategies

Technological

- Secure software development
  - Secure agile development
- Automated testing
  - Fuzz testing

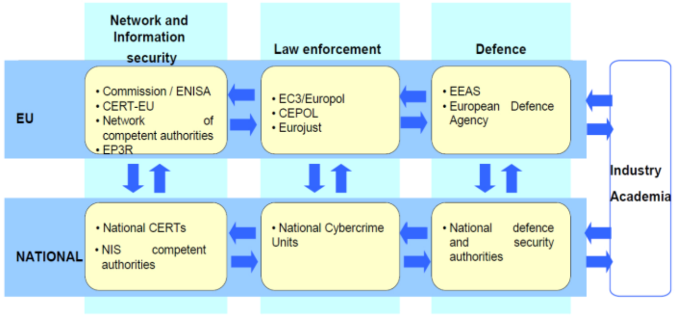

1 A sufficient amount of data is gathered for analysis and fed to the gene analyzer.
2 The analyzer produces a gene expression that approximates and sums up the data's structure in terms of common elements found in protocol implementations: length-payload pairs, unions, sequences of elements etc.
3 The gene expression is fed to the fuzzer that produces data packets that, although mutated, have roughly the same structure as the original data packets.
4 The MUTANT PACKETS from PROTOCOL SPACE are fed to implementations.

(OUSPG)

Software quality strategies

Governance

- Network and information security (NIS) directive
  - Focus on collaboration and coordination
  - Incident reporting obligations
- General data protection regulation (GPDR)
  - Article 25: Data protection by design and by default
  - Article 32: Security of Processing (e.g. monitoring and evaluation of security measures)
- ISO/IEC 27000: Information security management
  - ISO/IEC 27034: Application security (e.g. implementation of security controls)



Network and Information security
Law enforcement
Defence
EU
• Commission / ENISA • CERT-EU • Network of competent authorities • EP3R
• EC3/Europol • CEPOL • Eurojust
• EEAS • European Defence Agency
NATIONAL
• National CERTs • NIS competent authorities
• National Cybercrime Units
• National defence and security authorities
Industry
Academia

(EU Cybersecurity Strategy)

Software quality strategies

Organizational

– How to manage software and software vulnerabilities on a day to day basis

– Implement security policies that respect dynamic nature of software

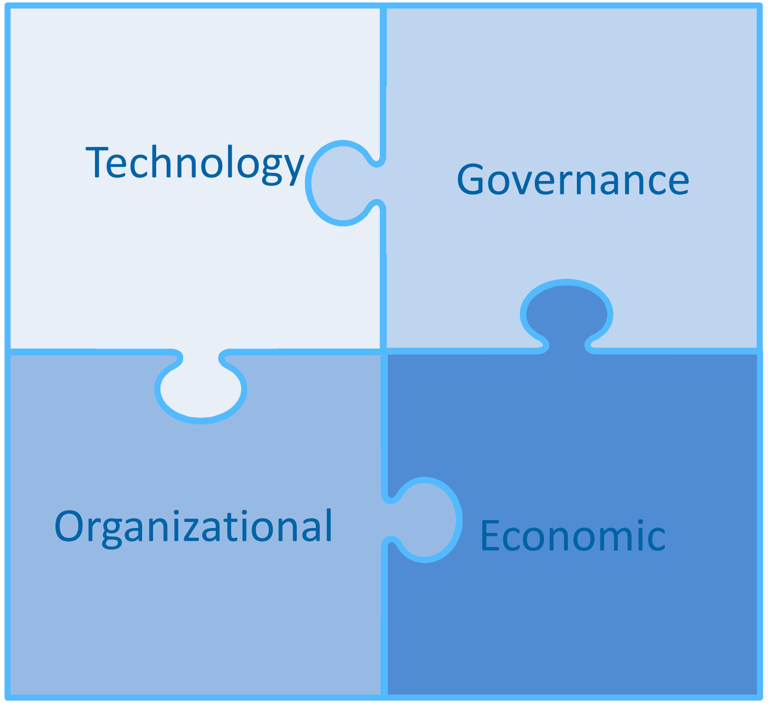
- Software is never finished, it is constantly repaired and improved
- Update and upgrade procedures
- Phase out legacy software in time
- Create awareness and provide information

Software quality strategies

Economic

- How to you get companies to pay more for higher quality software?
- “Market of lemons” situation
  - Information asymmetry between buyer and seller – the buyer does not know about the quality of software, only the seller
  - Higher quality software will disappear from the market
- Create incentives for investment in high quality software
  - Increase customer demand through awareness
  - Network economics: If enough players do it, the others need to follow
  - Compliance to regulations and standards
  - Reward/Penalty system

The silver lining



DAHINTER
SPIELT
DIE
ZUKUNFT.
Offen für Neues.
Seit 1365.
universität wien
galekovic.at