

Automated System Level Software Testing of Networked Embedded Systems

Mälardalen University Licentiate Thesis 275

Per Erik Strandberg



MÄLARDALEN UNIVERSITY
SWEDEN

1. Personal and Industrial Context



Per Erik Strandberg, the Test Expert

- 2017: Industrial doctoral student
- 2006 – 16: Work in “industry”
- 2011-16: competence network in testing
- Certified tester (3xISTQB)
- Certified professional for requirements engineering (REQB CPRE)
- Employed full time at Westermo



Per Erik Strandberg at ESEM'18. Photo: Päivi Raulamo-Jurvanen

Per Erik Strandberg, the Person



Selfie. Photo: Per Erik Strandberg

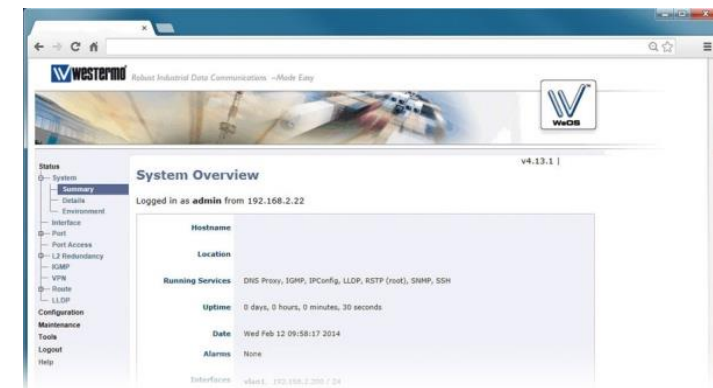
Westermo

- Founded 1975
- 200+ employees
(Most in Västerås and Stora Sundby)
- Westermo designs and manufactures data communications products for mission-critical systems in physically demanding environments.



WeOS – Westermo Operating System

- One software – Many products
 - GNU/Linux + Open Source
 - Proprietary 3rd party libraries
 - Proprietary internal code
- Developed every day
- Tested every night

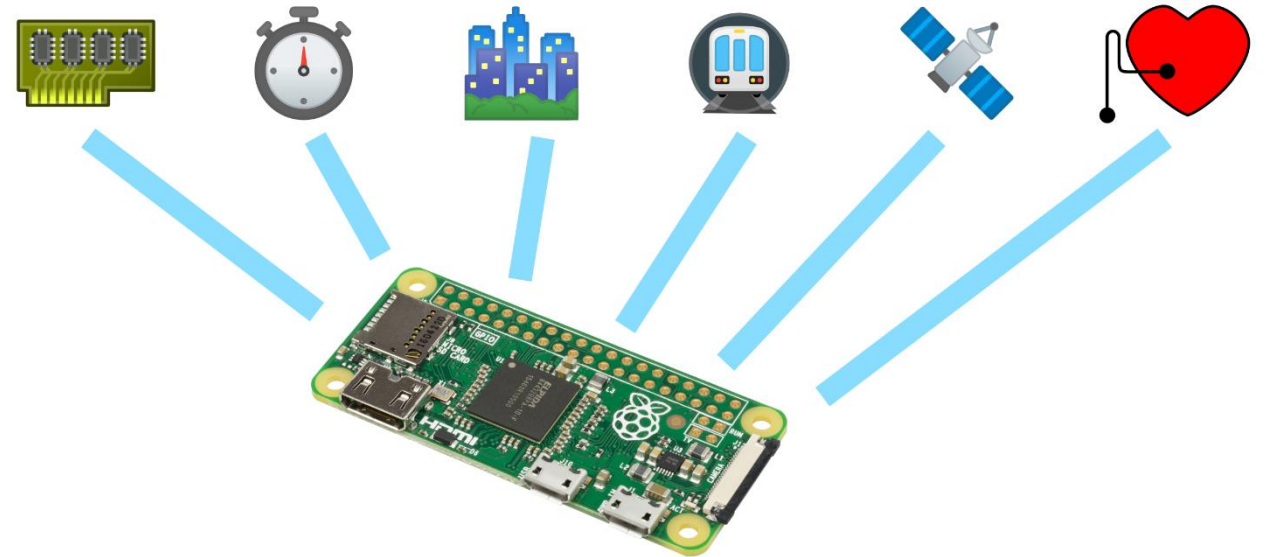




2. Networked Embedded Systems?

Embedded System

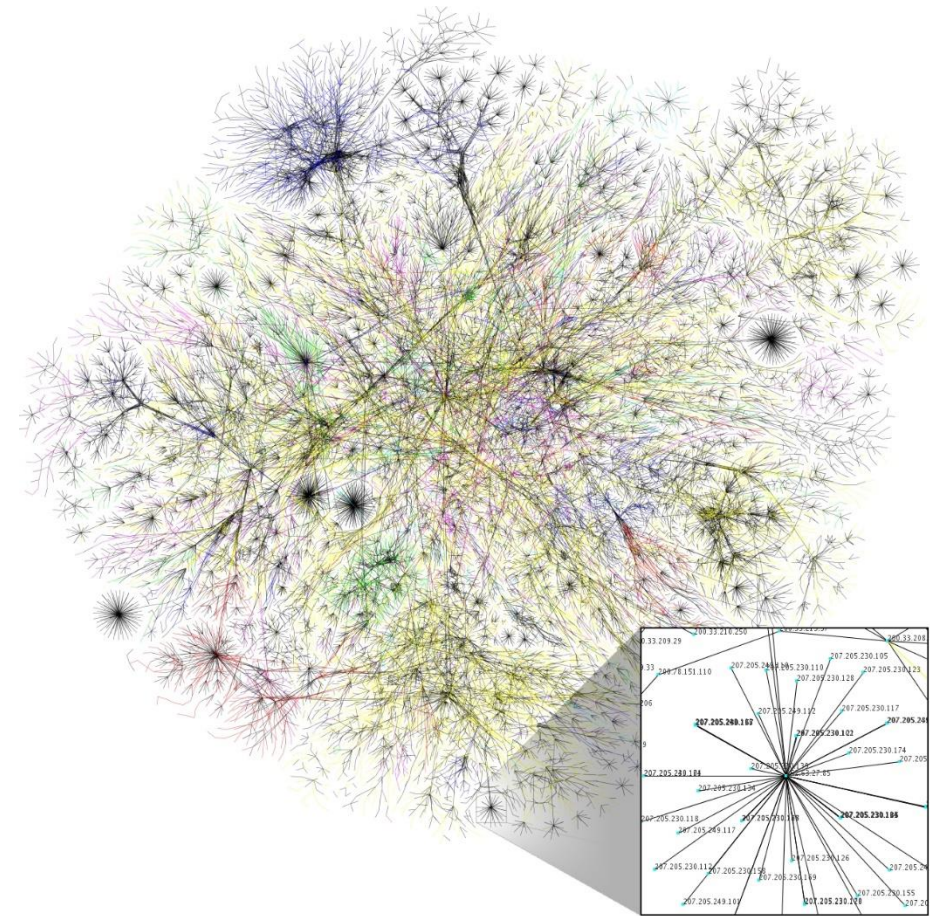
- “An embedded system is a programmed controlling and operating system with
- a dedicated function
- within a larger mechanical or electrical system” –Wikipedia
- Typically: no mouse, keyboard, monitor, ...



Raspberry Pi Zero. Photo: Evan-Amos, Public Domain

Computer Network

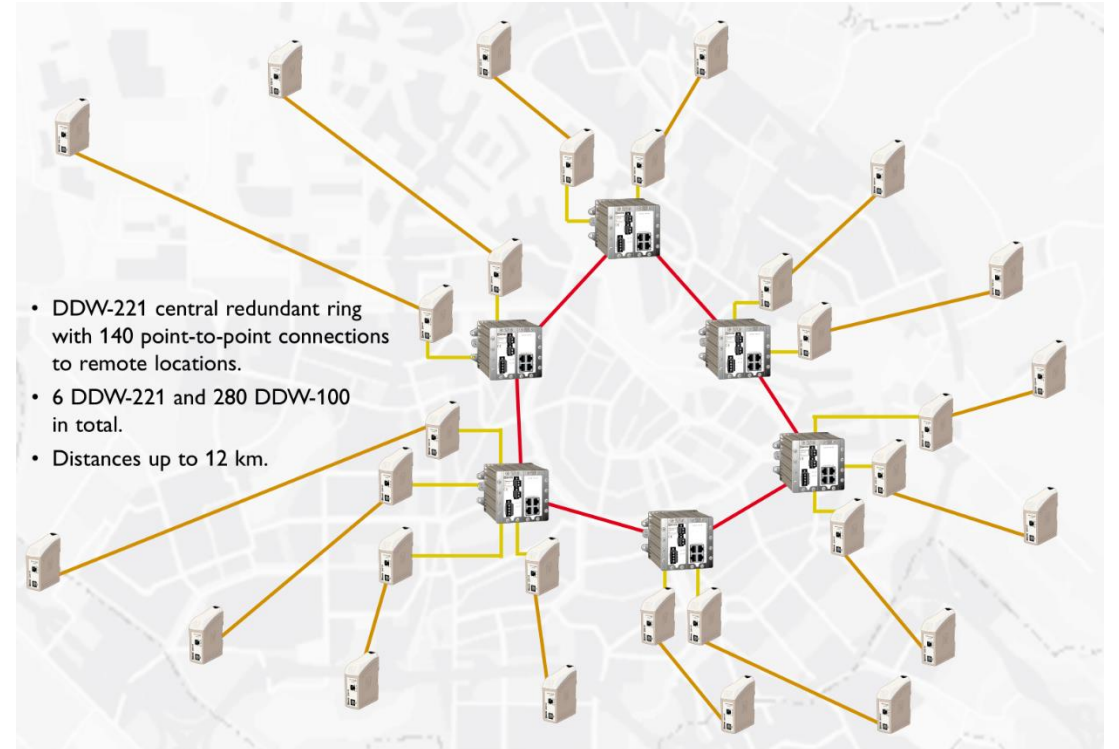
- “A computer network, or data network, is a digital telecommunications
- network which allows
- nodes to share resources” -- Wikipedia



The Internet (partial map as of 2005). Image: the Opte Project, CCA 2.5

Networked Embedded Systems

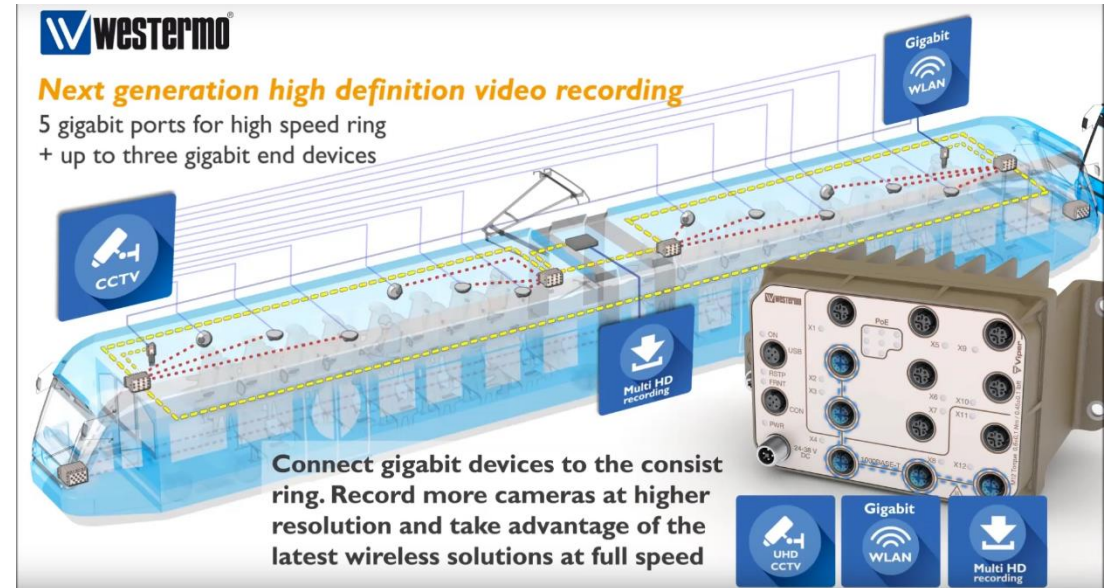
- In our cities!



Network Amsterdam gas and energy distribution. Image: Westermo.com

Networked Embedded Systems

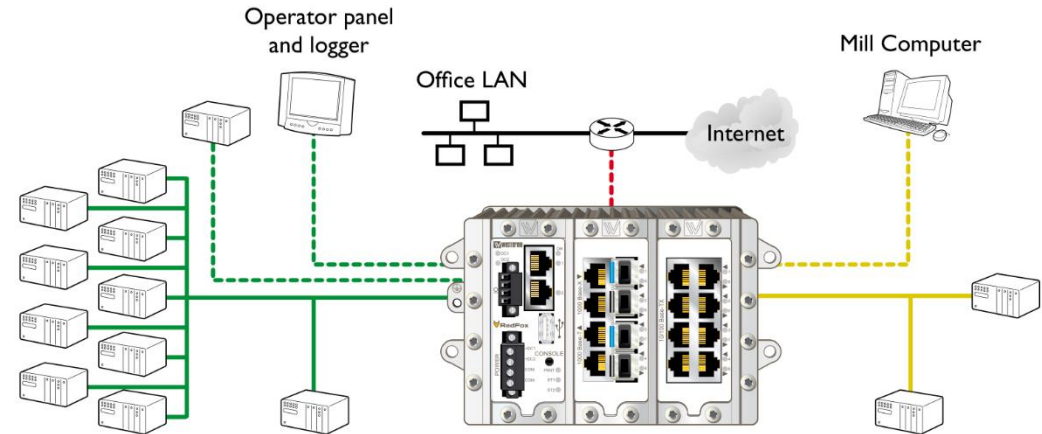
- In our cities!
- In our transportation systems!



Train Network. Image: Westermo at YouTube

Networked Embedded Systems

- In our cities!
- In our transportation systems!
- In our factories!



Industry/Factory Network. Image by Westermo.com

Networked Embedded System

- In our cities!
- In our transportation systems!
- In our factories!
- In our homes!



Remote Lock

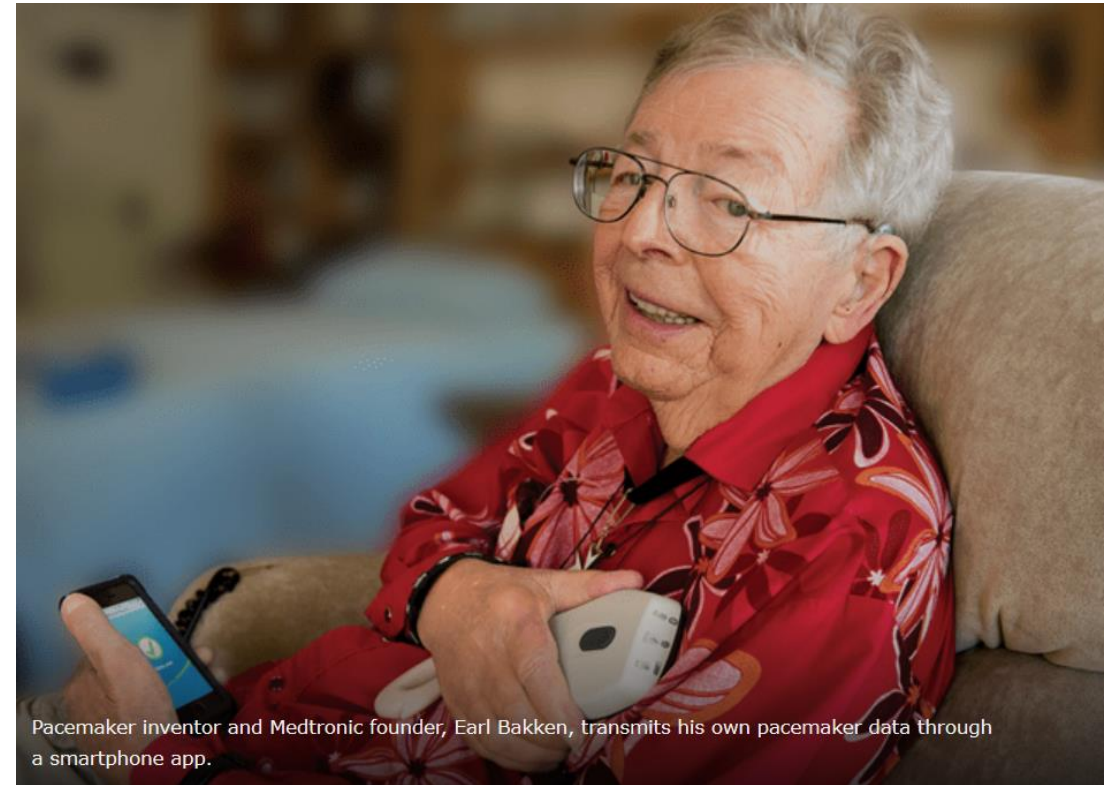


IOS and Android

Pet Pal Wifi Feeder

Networked Embedded Systems

- In our cities!
- In our transportation systems!
- In our factories!
- In our homes!
- Inside our bodies!



Pacemaker inventor and Medtronic founder, Earl Bakken, transmits his own pacemaker data through a smartphone app.

Screenshot from hospitalnews.com

Software Quality? Bugs?

- Quality shortcomings
 - Johnson & Johnson warns diabetic patients: [Insulin pump vulnerable to hacking](#), Jim Finkle, Reuters, 2016 Oct 4
- Software Quality is Key!
- Testing is the Standard Method

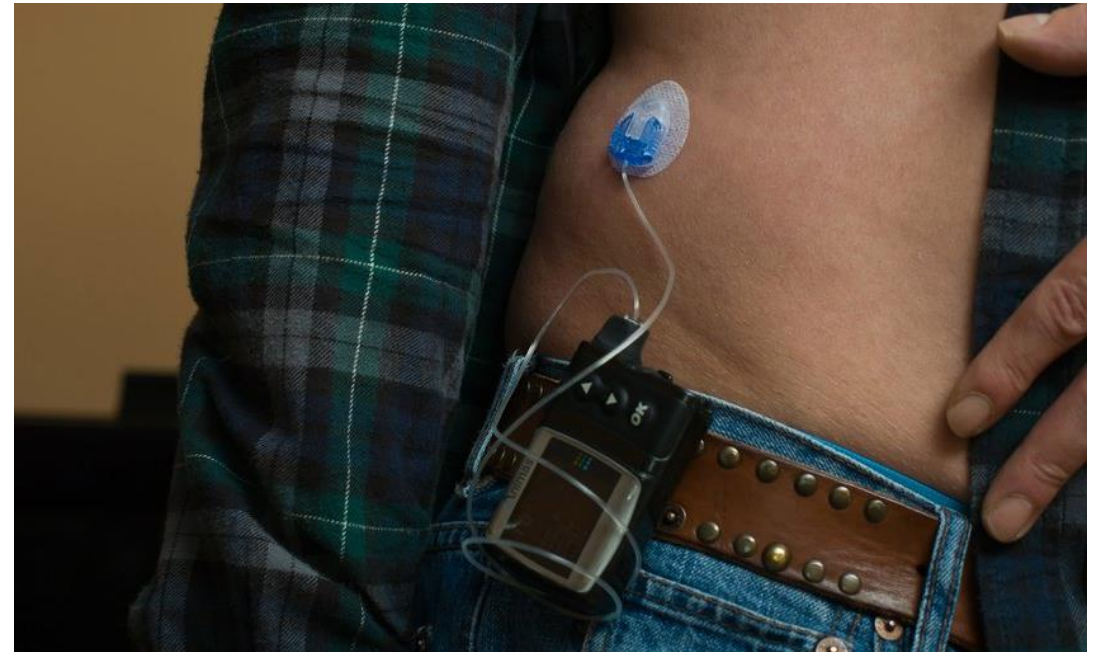


Image from businessinsider.com

Software Testing!

Software testing, or testing, can be defined as the act of:

manually or automatically

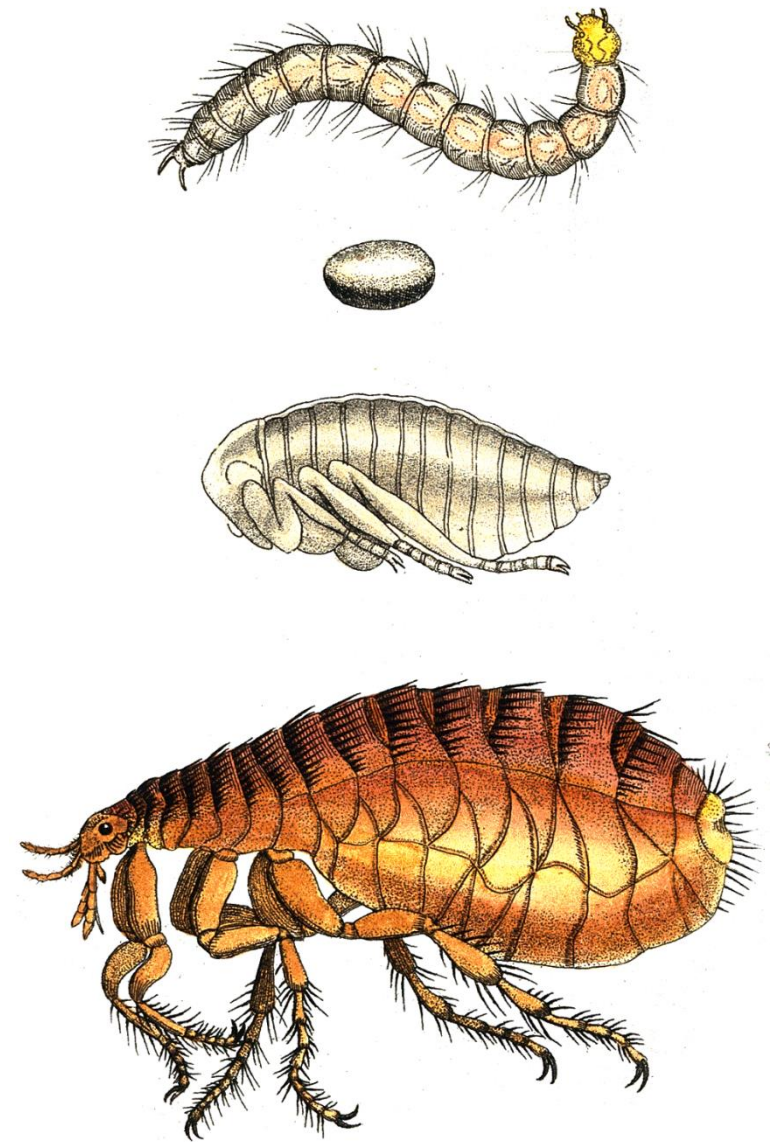
inspecting or executing software

with or without custom hardware

in order to gather information for some purpose:

feedback, quality control, finding issues (“bugs”), building trust, or other.

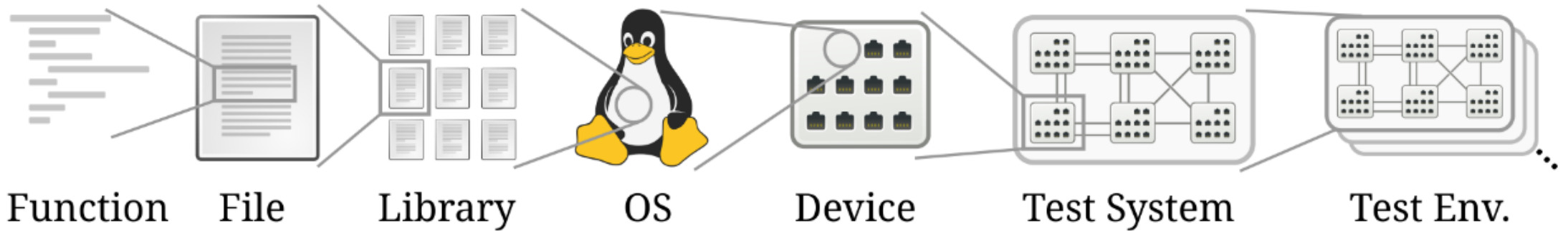
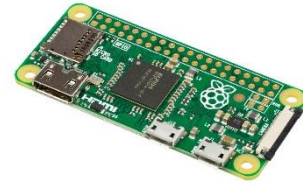
-- Per Erik Strandberg



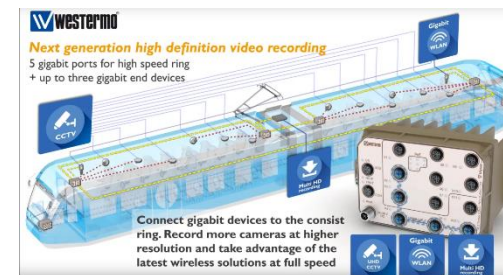
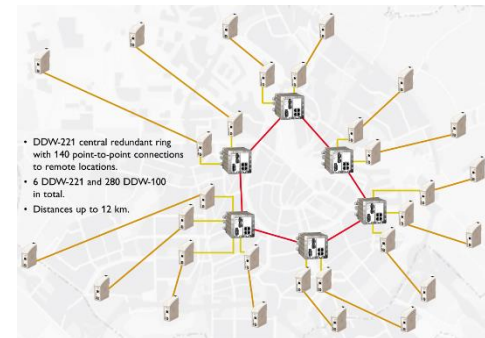
3. Testing Networked Embedded Systems



Test Levels

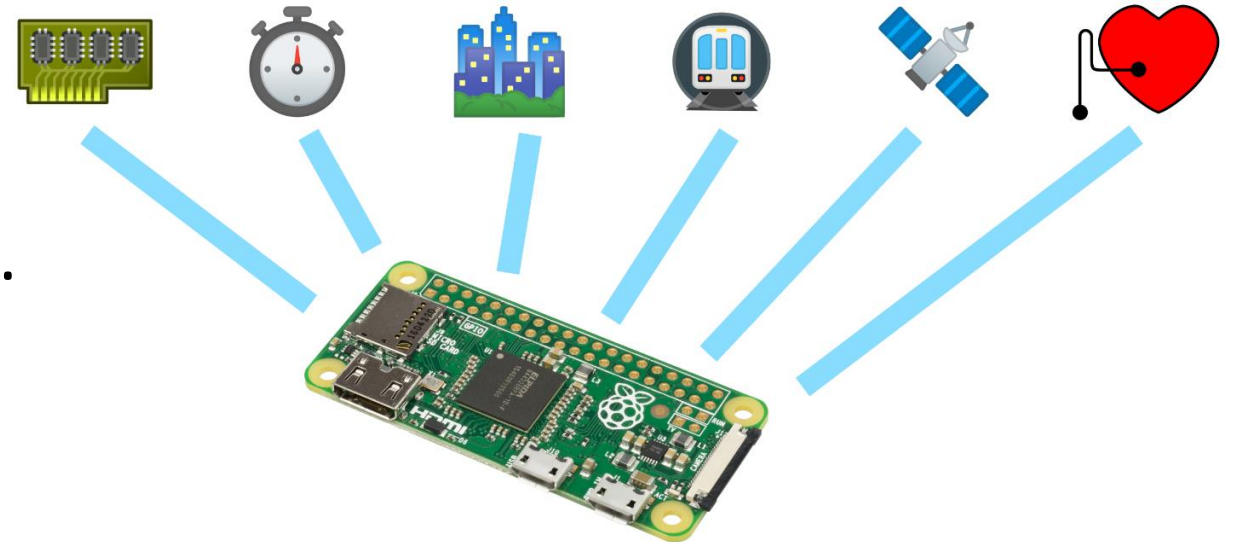


- Left: Unit level testing
 - 0.2 ms/test case (Paper A)
- Right: System level testing
 - 2 minutes/test case (Paper A)



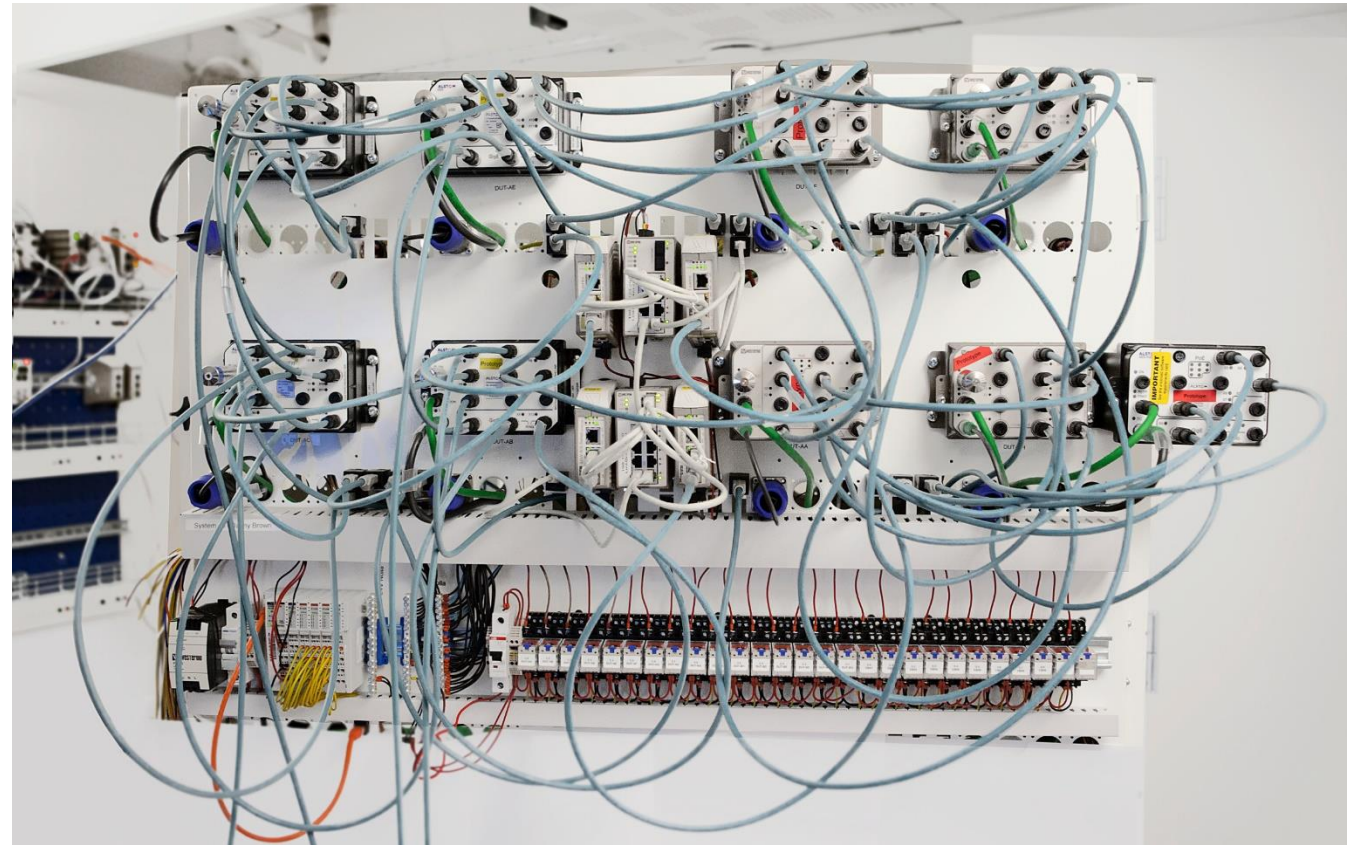
Testing Networked Embedded Systems

- Devices within a larger system.
- We need to test on real hardware.
 - [3] Banerjee et al. (2016)
 - [53] Ronsenkranz et al. (2015)
 - [71] Wolf (1994)
- Build Test Systems



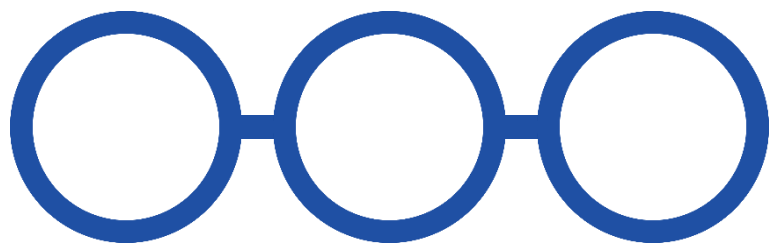
A Westermo Test System

- Devices Under Test (DUTs)
 - Run WeOS
- PC Server (not seen)
- Cables
- Linkbreakers
- IO's
- On a wagon (with wheels)

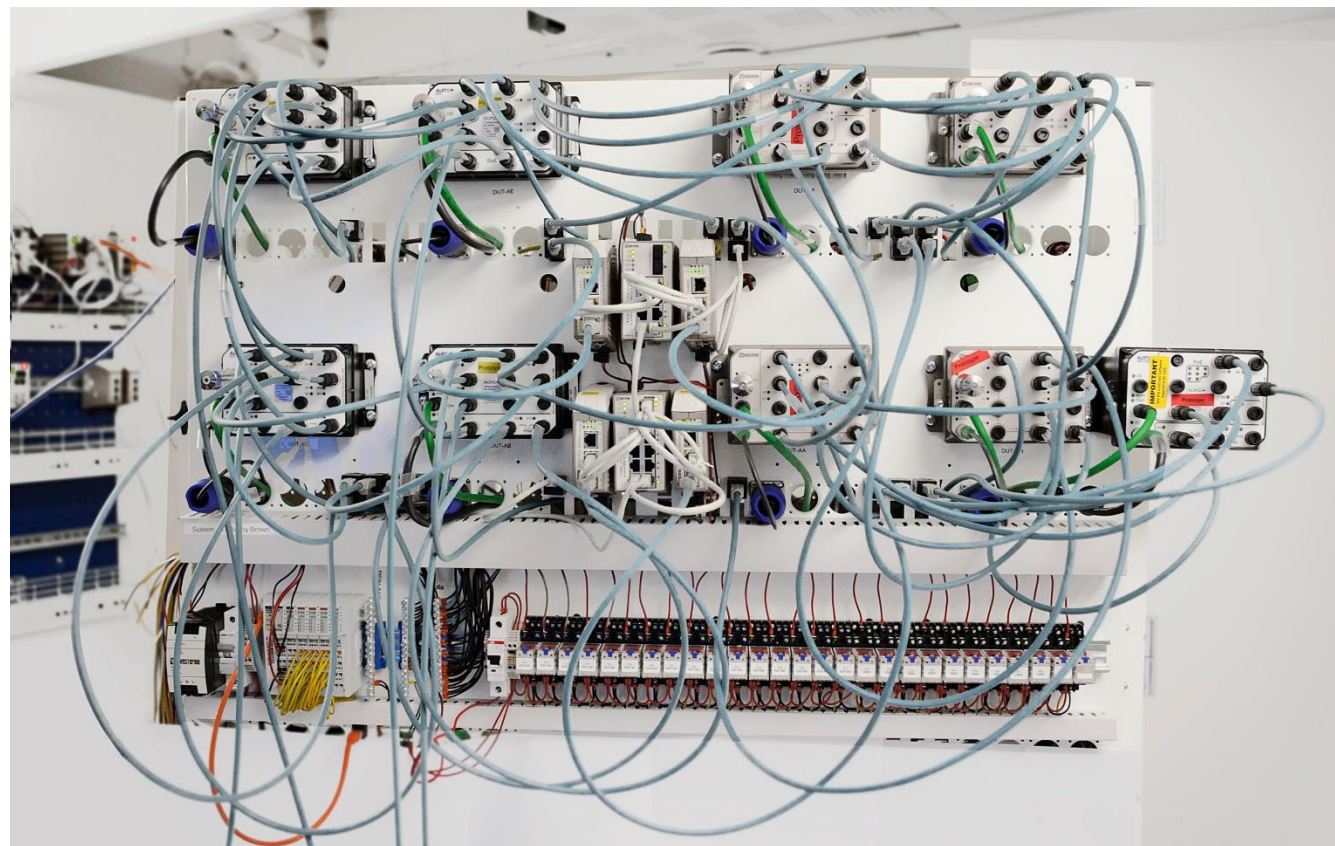


A Westermo Test System. Image from Thesis Introduction.

Firewall: A Typical Test Case

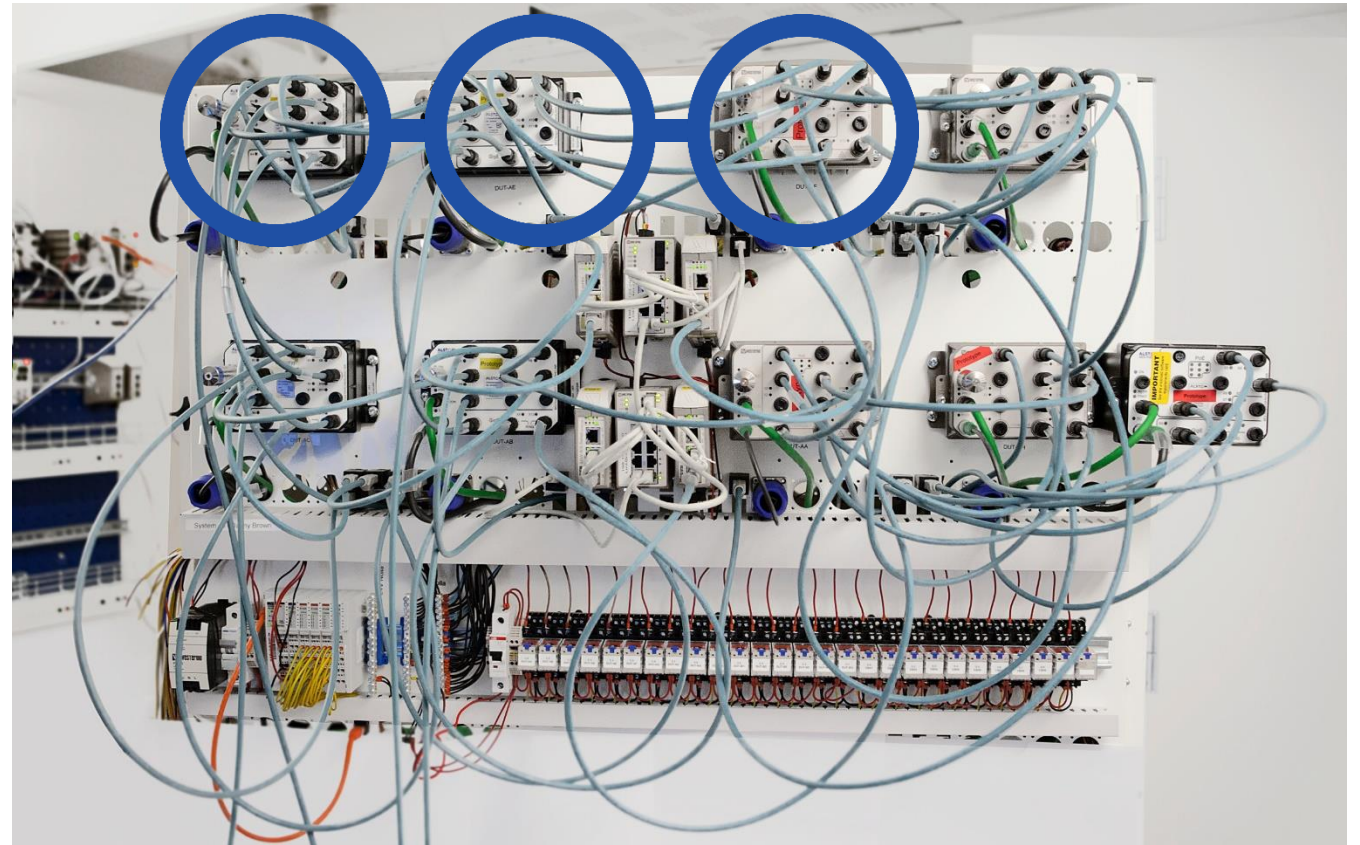


- DUT 1: Inside
- DUT 2: Firewall
- DUT 3: Outside



Firewall: A Typical Test Case

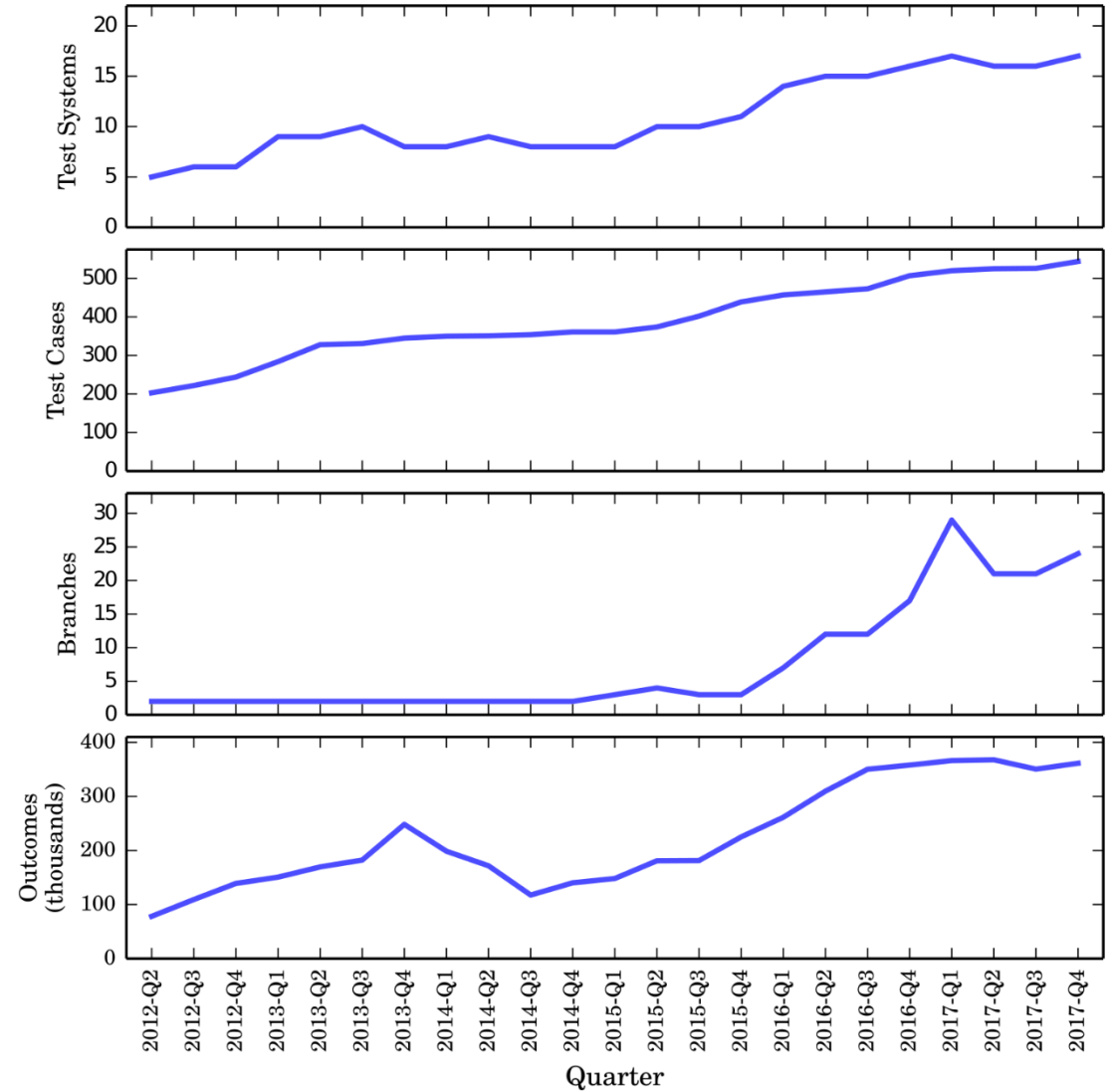
- DUT 1: Inside
 - DUT 2: Firewall
 - DUT 3: Outside
-
- Example of one “Mapping”,
on one test system
 - 15 s/mapping (Paper B)



Automation Challenge

- Westermo is doing well
- New HW products 😊
- New SW features 😊
- New SW development model 😊
- Million outcomes/year 😊

- Test all combinations? 😞
- Who receives the information? 😞



Increasing Combinatorial Complexity. Image from Paper C

Motivation

- Test Automation Challenges, [69] Wiklund et al. (2017)
 - Lack of time
 - Test systems not available
- Automated/Continuous Practices, [56] Shahin et al. (2016)
 - Exponential growth of information
 - Lack of awareness and transparency
- This matches what we see at Westermo

4. Research Questions



Research Goal and Questions

Describe and improve upon industrial automated system level software testing of networked embedded systems.

1. Test selection
2. Test Env. Assignment
3. Information Flow

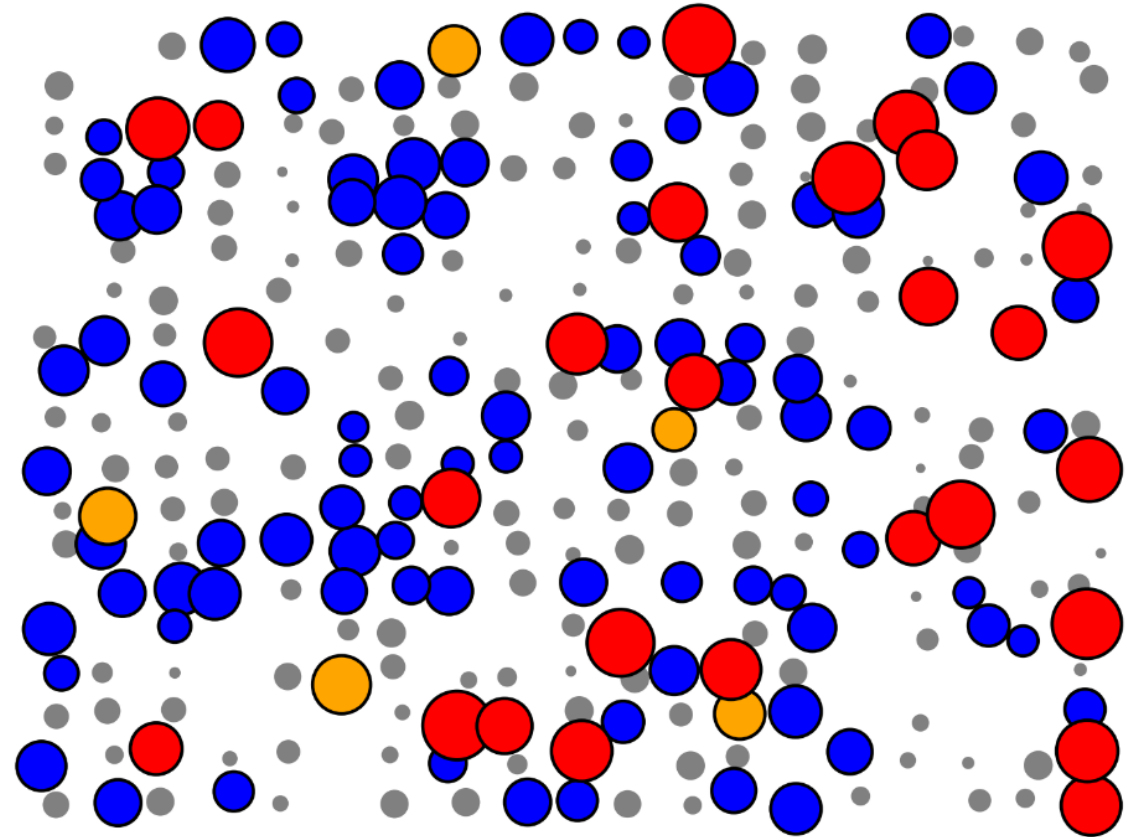


The Goal. Photo: Per Erik Strandberg

Research Goal and Questions

Describe and improve upon industrial automated system level software testing of networked embedded systems.

1. **Test selection**
2. Test Env. Assignment
3. Information Flow

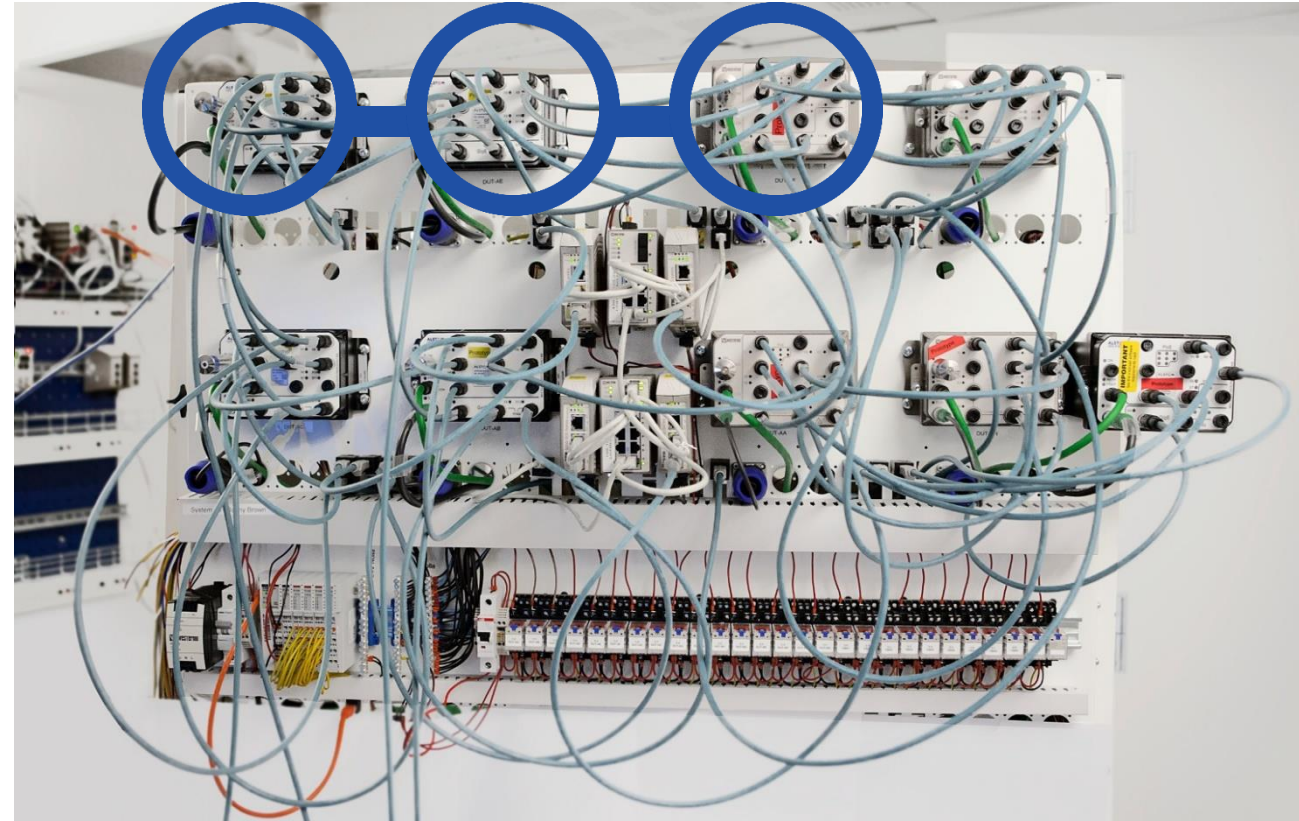


Select from 275 dots. Illustration: Per Erik Strandberg

Research Goal and Questions

Describe and improve upon industrial automated system level software testing of networked embedded systems.

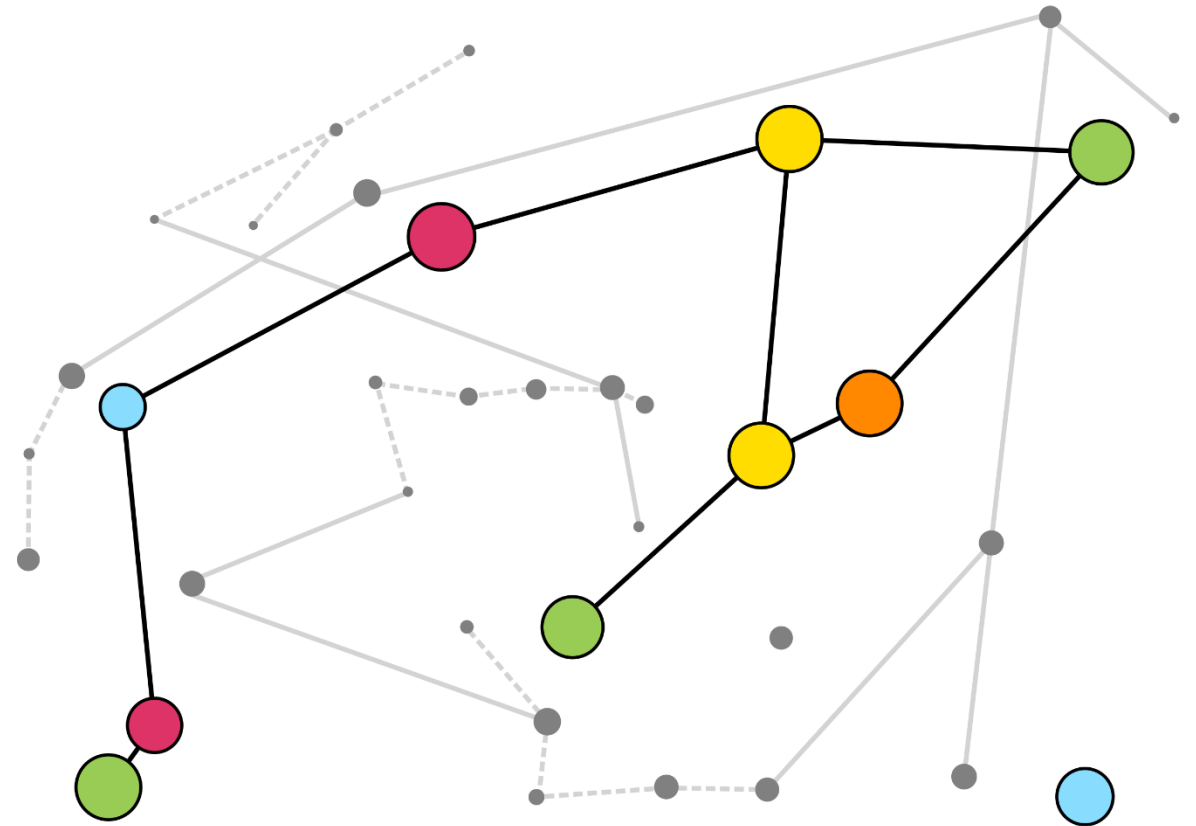
1. Test selection
2. **Test Env. Assignment**
3. Information Flow



Research Goal and Questions

Describe and improve upon industrial automated system level software testing of networked embedded systems.

1. Test selection
2. Test Env. Assignment
- 3. Information Flow**



Abstract Information Flow. Illustration: Per Erik Strandberg

5. Research Methods



Methods/Algorithms/Tools

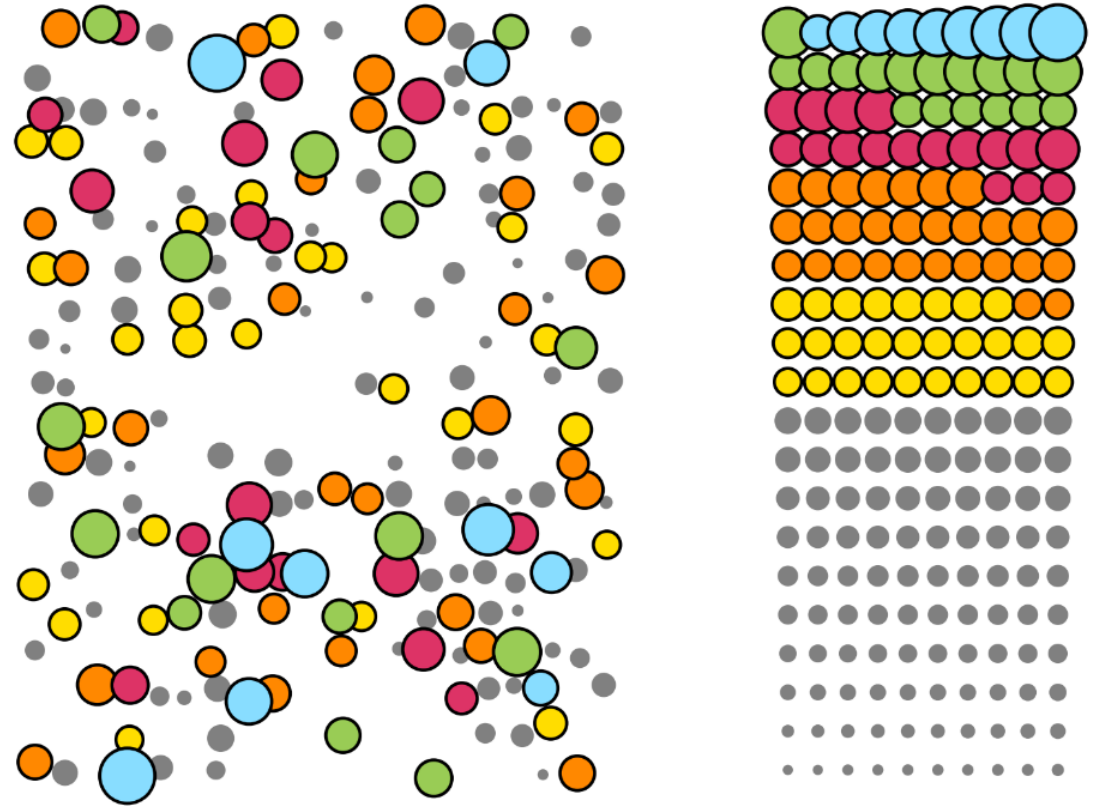
- Identify a problem
 - No time for testing
 - Shortcomings in Mapping
- Implement a tool
 - SuiteBuilder
 - Mapper



Victorinox Huntsman in red. Photo: Victorinox.com

Quantitative Empirical Studies

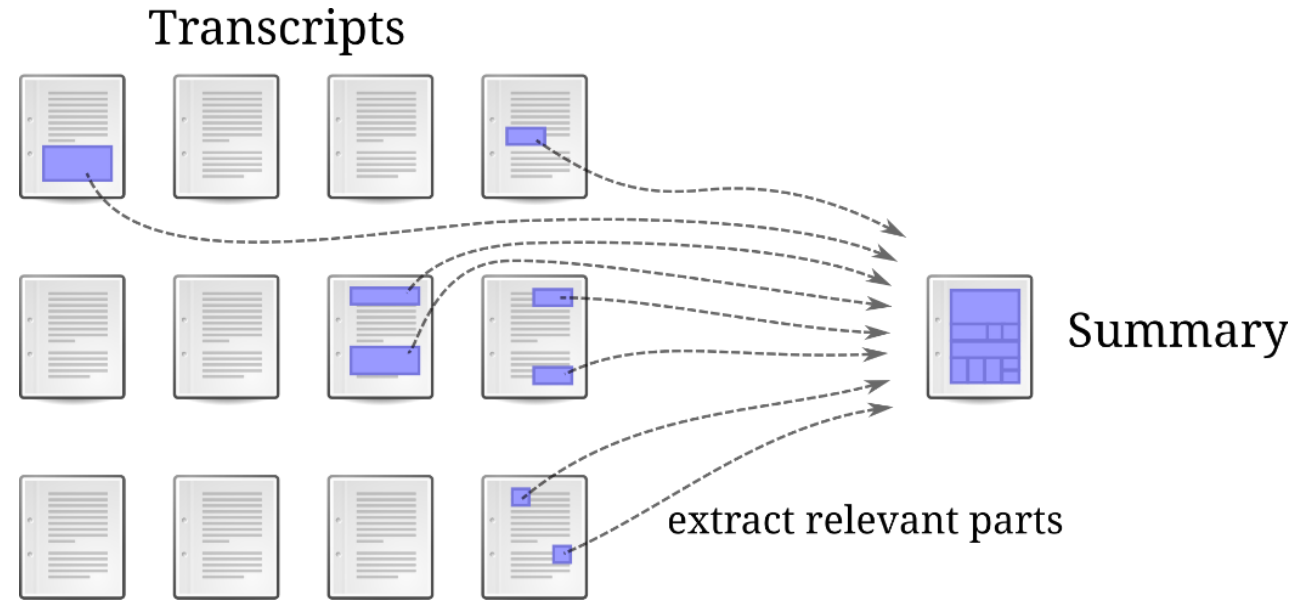
- Compare Before/After
- Measure Improvement
- 4 years of nightly test data
- 10000 mappings



Before and After. Illustration: Per Erik Strandberg

Qualitative Empirical Studies

- Interviews
- Transcription
- Thematic Analysis
 - [7] Braun & Clarke (2006)
 - Also used scripts as aid
- Overall guideline:
 - [40] Linåker et al. (2015)



Scripted Thematic Analysis. Illustration: Per Erik Strandberg

A person wearing a winter hat and jacket stands on a grassy bank, holding a large, dried, feathery plant stem. The sun is low on the horizon behind them, creating a silhouette effect. In the background, there is a body of water and a clear blue sky. The text "6. Research Contributions" is overlaid in the center of the image.

6. Research Contributions

Four Papers in Thesis

A: System Level Regression Test Selection

P. E. Strandberg, D. Sundmark, W. Afzal, T. J. Ostrand, and E. J. Weyuker. Experience Report: Automated System Level Regression Test Prioritization Using Multiple Factors. In ISSRE'16, 2016. Best research paper.

B: Test Environment Assignment

P. E. Strandberg, T. J. Ostrand, E. J. Weyuker, D. Sundmark, and W. Afzal. Automated Test Mapping and Coverage for Network Topologies. In ISSTA'18, 2018.

C: Decision-making and Visualizations

P. E. Strandberg, W. Afzal and D. Sundmark. Decision Making and Visualizations Based on Test Results. In ESEM'18, 2018.

D: Information Flow

P. E. Strandberg, E. P. Enou, W. Afzal, D. Sundmark, and R. Feldt. Information Flow in Software Testing – An Interview Study with Embedded Software Engineering Practitioners. In revision.

2016 IEEE 27th International Symposium on Software Reliability Engineering

Experience Report: Automated System Level Regression Test Prioritization Using Multiple Factors

Per Erik Strandberg¹, Daniel Sundmark¹, Wasif Afzal¹, Thomas J. Ostrand² and Elaine J. Weyuker³

¹ Western Research and Development AB, Västerås, Sweden
Email: perstrandberg@western.se

² Midland University, Wexford, Ireland
Email: thomas@midland.ie

³ Midland University, Wexford, Ireland
Email: eweyuker@midland.ie

Abstract—We propose a new method of determining an effective ordering of regression test cases, and describe its implementation as an extension to our Software Butler developed by Western Research and Development AB. The tool generates an efficient order to run the cases in an existing test suite by using reported or observed failure history and changing priorities. It multiplies failure associated with test cases, including previous fault detection success, interval since last passed, and modification to the code base. The method and tool were developed to address problems in the traditional process of regression testing, such as how to best run a complete regression suite. Failure in defect bugs in time, and suite that are especially useful. The tool has been integrated into the existing nightly test process for Western software that runs on a large number of development systems. The experimental results of the tool show significant improvement in regression testing results. The re-ordered test suite runs within the available time, the majority of fault detecting test cases are located in the first third of the suite, an important test case is omitted, and the severity for missed work on the suites is greatly reduced.

1. INTRODUCTION

Software testing is performed for many reasons. Possible objectives include giving feedback, finding or preventing failures, providing confidence and ensuring quality. Regression testing is an important part of the maintenance process for software systems that undergo periodic revision and enhancement. Whenever a system is updated, either with fixes or improvements to existing code or with new functionality, it is necessary to ensure that the system has not regressed, i.e. that the modifications have not introduced faults that might affect previously satisfactory operation of the system. Regression testing aims to detect such faults in the modified system by running all or some of the existing test cases that were used to evaluate the system's former version.

Because there is frequently not enough time, equipment or personnel available to rerun the entire test suite, regression testers focus on selecting the most effective subset of the test suite, and prioritizing or determining an efficient order to execute the selected test cases.

2330-446X/18/0000-0000-0000
DOI: 10.1109/ISSRE.2016.23

Decision Making and Visualizations Based on Test Results

Per Erik Strandberg¹, Wasif Afzal¹, Daniel Sundmark²

¹ Western R & D AB, Sweden
perstrandberg@western.se

² Midland University, Sweden
daniel.sundmark@midu.ie

ABSTRACT—Testing is one of the main methods for quality assurance in the development of embedded software, an inherent aspect of software engineering in general. Consequently, test results need to be analyzed and visualized in a way that is understandable for business decision makers in software intensive organizations. Aims: This case study examines the use of test results and visualizations for decision making and the visualization of test results. The visualization of test results is used to support decision making and the visualization of test results is used to support decision making. We want to identify the use of the visualization for supporting decision from their specific test data. From their background and at release time. Method: We conducted an embedded case study with multiple areas of analysis by conducting interviews, questionnaires, using archival data and participant observations. Results: Several visualizations and reports that support the test results analysis are utilized in supporting daily work, meeting a business decision maker and at release time. Some important visualizations are: bar charts of failing test cases, area charts and funnel charts. The subjects' perceptions perceived the visualizations as supporting, as valuable, however they also mentioned several areas of improvement such as better ways of visualizing test results. Conclusions: The visualization of test results can be a valuable decision making tool for a variety of roles and tasks in embedded software development, however the visualization needs to be continuously improved to better fit their value for its stakeholders.

CCS CONCEPTS—Software and its engineering — Software creation and management; Software testing and debugging; Empirical software validation.

KEYWORDS—Testing; Visualizations; Decision Making

ACM Reference Format:
Per Erik Strandberg, Wasif Afzal, and Daniel Sundmark. 2018. Decision Making and Visualizations Based on Test Results. In 2018 IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 1–10. New York, NY, USA: ACM. <https://doi.org/10.1145/3201187>

Permission is made digital and analog to publish and to reproduce this paper for personal or professional use, provided that the copyright owner is credited and that the copyright notice is included in the publication. This permission is granted without fee except where indicated otherwise. This permission does not extend to other kinds of copying, such as for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale.

Automated Test Mapping and Coverage for Network Topologies

Per Erik Strandberg¹, Thomas J. Ostrand², Elaine J. Weyuker³

¹ Western Research and Development AB, Västerås, Sweden
perstrandberg@western.se

² Midland University, Wexford, Ireland
thomas@midland.ie

³ Midland University, Wexford, Ireland
eweyuker@midland.ie

ABSTRACT—Communication devices such as routers and switches play a critical role in the reliable functioning of embedded system networks. Devices of such devices may be part of an embedded system network, and they need to be tested in conjunction with various computational elements on actual hardware, in many different configurations that are representative of actual operating networks. An individual physical network topology can be used as the basis for a test suite that can exercise many test cases. Identifying the goal of the physical network topology that corresponds to the configuration required for each individual test case. Limits on the available test resources and a large number of test cases are the main challenge in determining the test suite. Which of the test cases are suitable for executing, for test cases, and to provide the mapping that associates the test case elements of the logical network topology with the appropriate elements of the test system (the physical network topology).

We studied a real industrial environment where this problem was originally handled by a simple software procedure that was very slow to run, and we added the parallel processing coverage of each network's elements. In this paper, we propose both the test system and the test cases to simplify, and develop a new prototype algorithm that identifies whether or not a test case can be mapped to a subgraph of the test system. To simplify fault mappings that do not, and it creates diverse sets of network topology to multi-coverage tests for the test case. The prototype has been implemented and applied to over 10,000 combinations of test cases and test systems, and reduced the computation time by a factor of more than 10 from the original procedure. This results in a more efficient means of network topology coverage, the mapping shows an increased level of thoroughness in covering the elements of each test system.

CCS CONCEPTS—Software and its engineering — Software creation and management; Software testing and debugging; Empirical software validation.

KEYWORDS—Networks; network topology; subgraph isomorphism

DOI: 10.1145/3201187

Information Flow in Software Testing – An Interview Study with Embedded Software Engineering Practitioners

Per Erik Strandberg¹, Eduard Paul Enou², Wasif Afzal¹, Daniel Sundmark¹, and Robert Feldt³

¹ Western Research and Development AB, Västerås, Sweden
perstrandberg@western.se

² Western University, Wexford, Ireland
eduard.paul@midland.ie

³ Western University, Wexford, Ireland
robert.feldt@midland.ie

Corresponding author: Per Erik Strandberg (perstrandberg@western.se)

This work was supported by the Swedish Foundation for Applied Research (2017-01018) and the Swedish Research Council (2018-04518).

2330-446X/18/0000-0000-0000
DOI: 10.1109/ISSRE.2018.2330446

ABSTRACT—Information flow in software testing is a challenge for companies that develop embedded systems where multi-functional teams and technologically different tasks are common. This study aims at exploring the information flow in software testing, the perceived challenges and good approaches for a more effective information flow. We conducted an interview study with twelve software practitioners working at five organizations in the embedded software industry in Sweden. The interviews were analyzed by means of thematic analysis. The data was classified into six themes that affect the information flow in software testing, testing and troubleshooting, communication, processes, technology, artifacts and organization. We interviewed a number of challenges such as poor feedback and understanding exactly what has been tested and approaches such as test feedback as well as process automation test reporting to enhance an improved information flow. Our results indicate that there are many opportunities to enhance the information flow: a first mitigation step is to better understand the challenges and approaches. Future work is needed to realize this mitigation step, for example to shorten feedback cycles between roles, as well as an enhanced exploration and visualization of test results.

INDEX TERMS—Embedded Software Development, Information Flow, Interview Study, Software Testing

1. INTRODUCTION

Software testing includes not only the creation, execution and evaluation of test cases, but also the process of communicating test cases and their results between human beings, or between human and machines. This includes face-to-face communication, as well as producing standardized computerized reports, reading system logs, or making decisions based on test results visualizations. Many practitioners and researchers know little about the information flow in the practice of software testing including documented and non-documented channels of information flow between different stakeholders. There is a broad industrial interest in this topic. Industry standards such as ISO/IEC 29119-1 [3] and IEEE Std 1059-2016 [4] acknowledge that testers need to communicate with

Software Testing Qualifications Board (ISTQB) defines on test automation prescriptive approaches to logging and storing test results [5].

Information flow can be defined by a distributed system made up of agents and their behavioral and structural roles and relationships, as well as producing standardized computerized reports, reading system logs, or making decisions based on test results visualizations. Many practitioners and researchers know little about the information flow in the practice of software testing including documented and non-documented channels of information flow between different stakeholders. There is a broad industrial interest in this topic. Industry standards such as ISO/IEC 29119-1 [3] and IEEE Std 1059-2016 [4] acknowledge that testers need to communicate with

Software Testing Qualifications Board (ISTQB) defines on test automation prescriptive approaches to logging and storing test results [5].

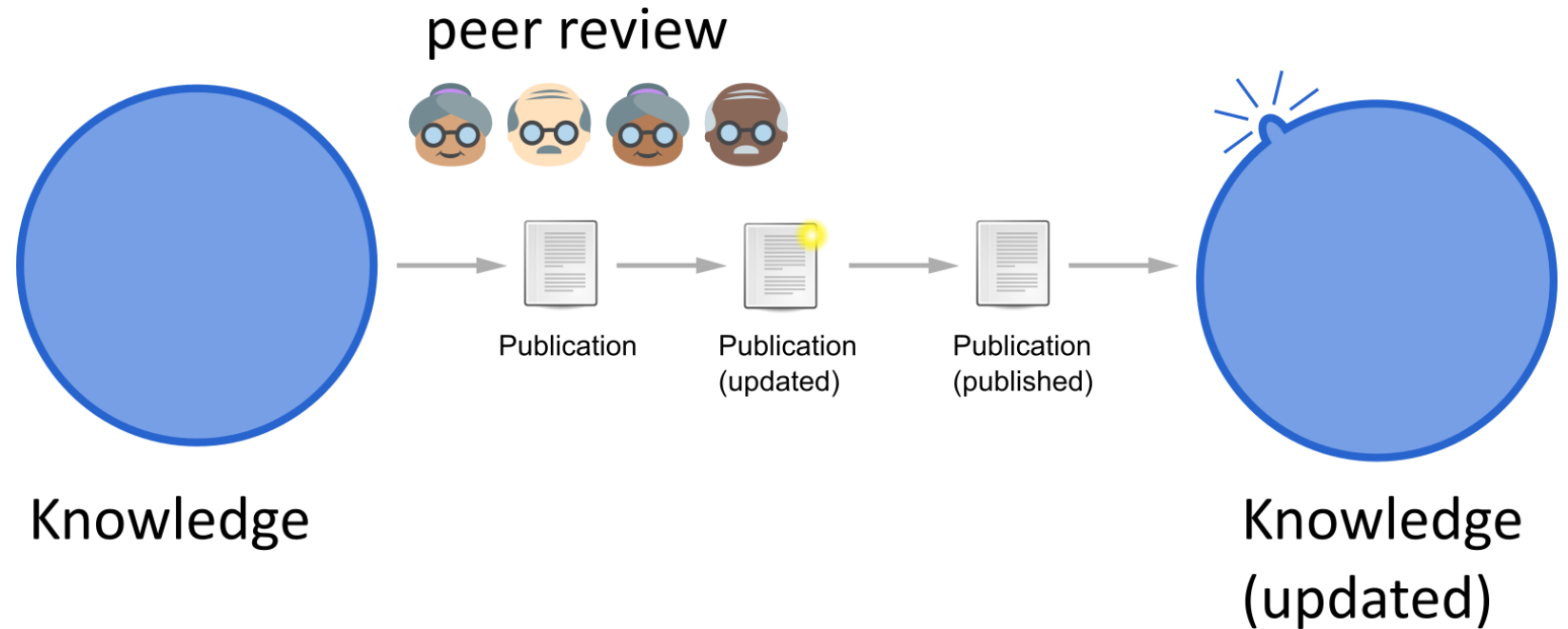
Information flow can be defined by a distributed system made up of agents and their behavioral and structural roles and relationships, as well as producing standardized computerized reports, reading system logs, or making decisions based on test results visualizations. Many practitioners and researchers know little about the information flow in the practice of software testing including documented and non-documented channels of information flow between different stakeholders. There is a broad industrial interest in this topic. Industry standards such as ISO/IEC 29119-1 [3] and IEEE Std 1059-2016 [4] acknowledge that testers need to communicate with

Software Testing Qualifications Board (ISTQB) defines on test automation prescriptive approaches to logging and storing test results [5].

Information flow can be defined by a distributed system made up of agents and their behavioral and structural roles and relationships, as well as producing standardized computerized reports, reading system logs, or making decisions based on test results visualizations. Many practitioners and researchers know little about the information flow in the practice of software testing including documented and non-documented channels of information flow between different stakeholders. There is a broad industrial interest in this topic. Industry standards such as ISO/IEC 29119-1 [3] and IEEE Std 1059-2016 [4] acknowledge that testers need to communicate with

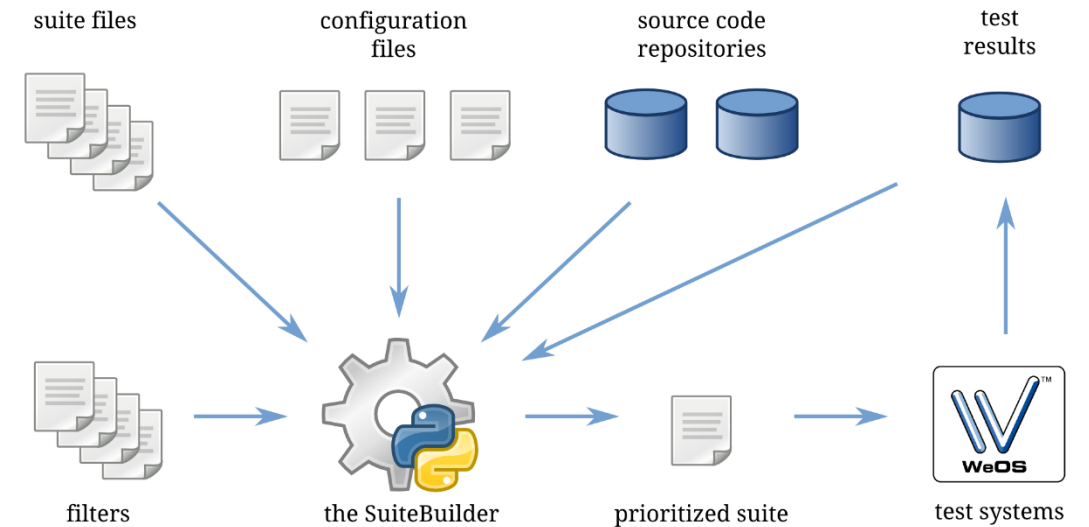
Why Publications?

- One publication
- One kilogram
- One man-month
- One line of code
- One bug
- One “like”



Paper A: System Level Regression Test Selection

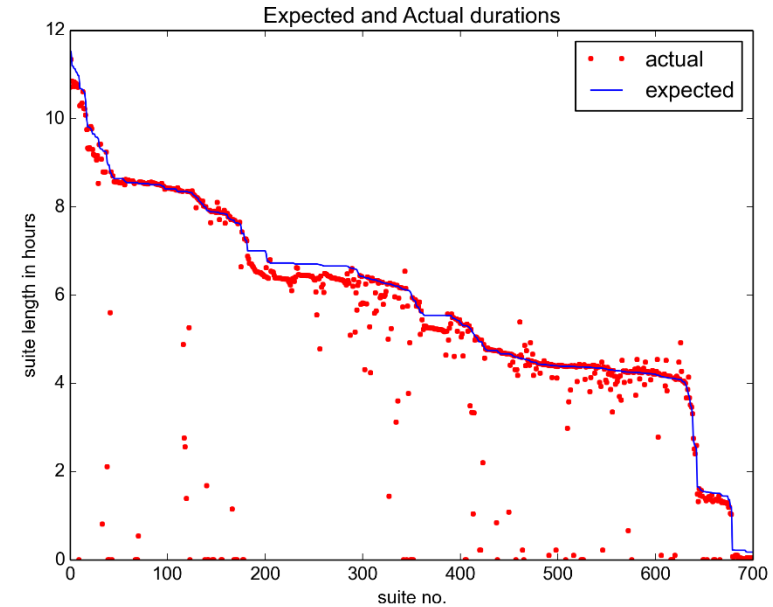
- Problem
 - Nightly testing does not finish on time
 - Manual work (and omitted test cases)
 - No priority
- Approach
 - SuiteBuilder tool
 - Prioritize tests
 - Estimate time and stop



Overall Data Flow in SuiteBuilder. Figure from Paper A

Paper A: System Level Regression Test Selection

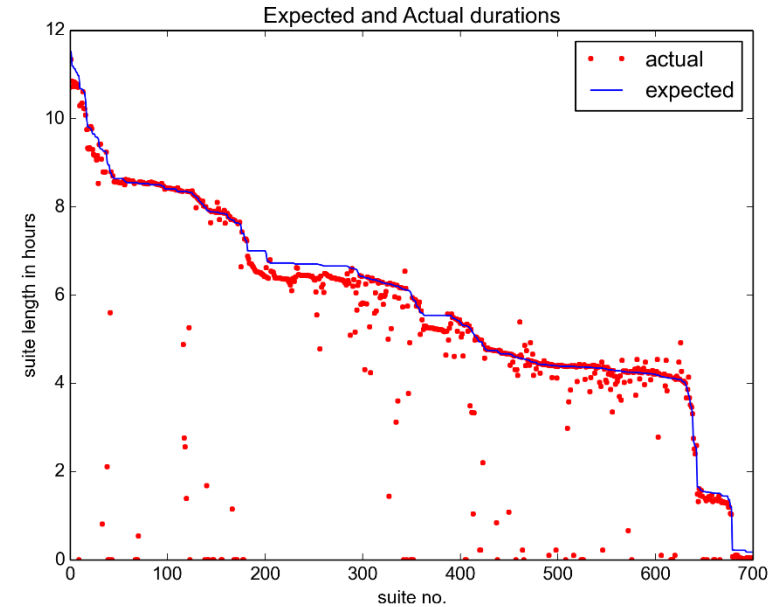
- Results
 - Suites finish on time



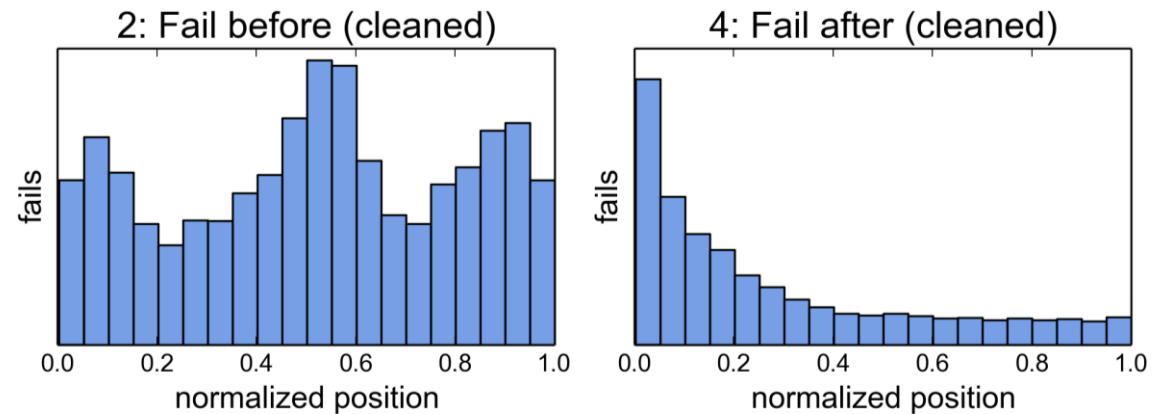
Figures from Paper A

Paper A: System Level Regression Test Selection

- Results
 - Suites finish on time
 - Find faults early

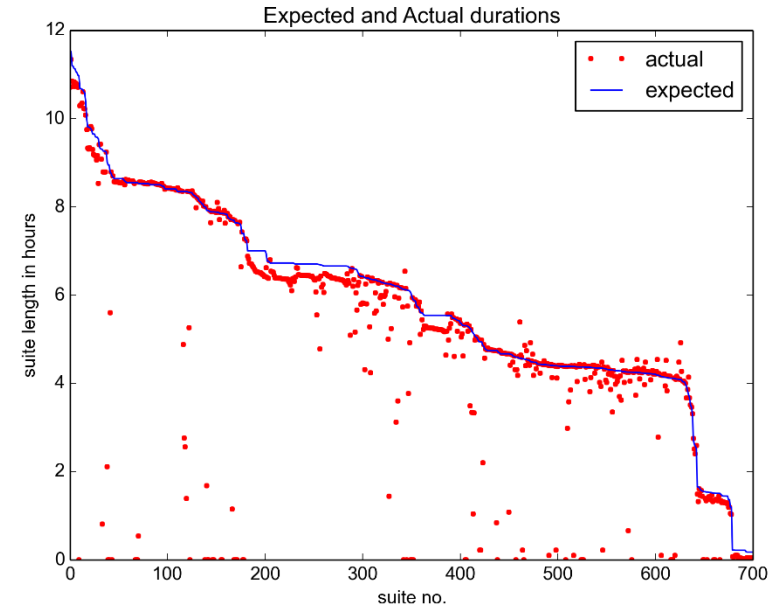


Figures from Paper A

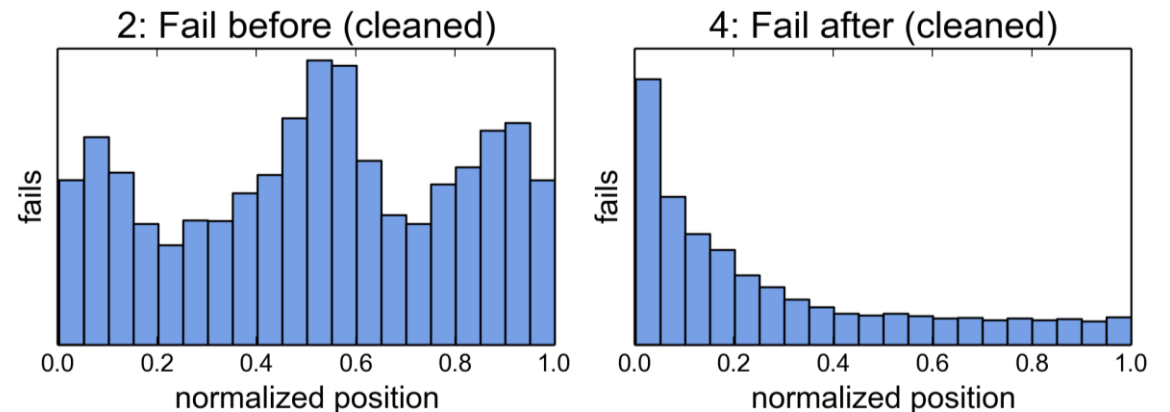


Paper A: System Level Regression Test Selection

- Results
 - Suites finish on time
 - Find faults early
 - Less manual work
- Contributions
 - Automated regression test selection on a *system-level*
 - Solution integrated in nightly testing
 - Industry-grade implementation and evaluation

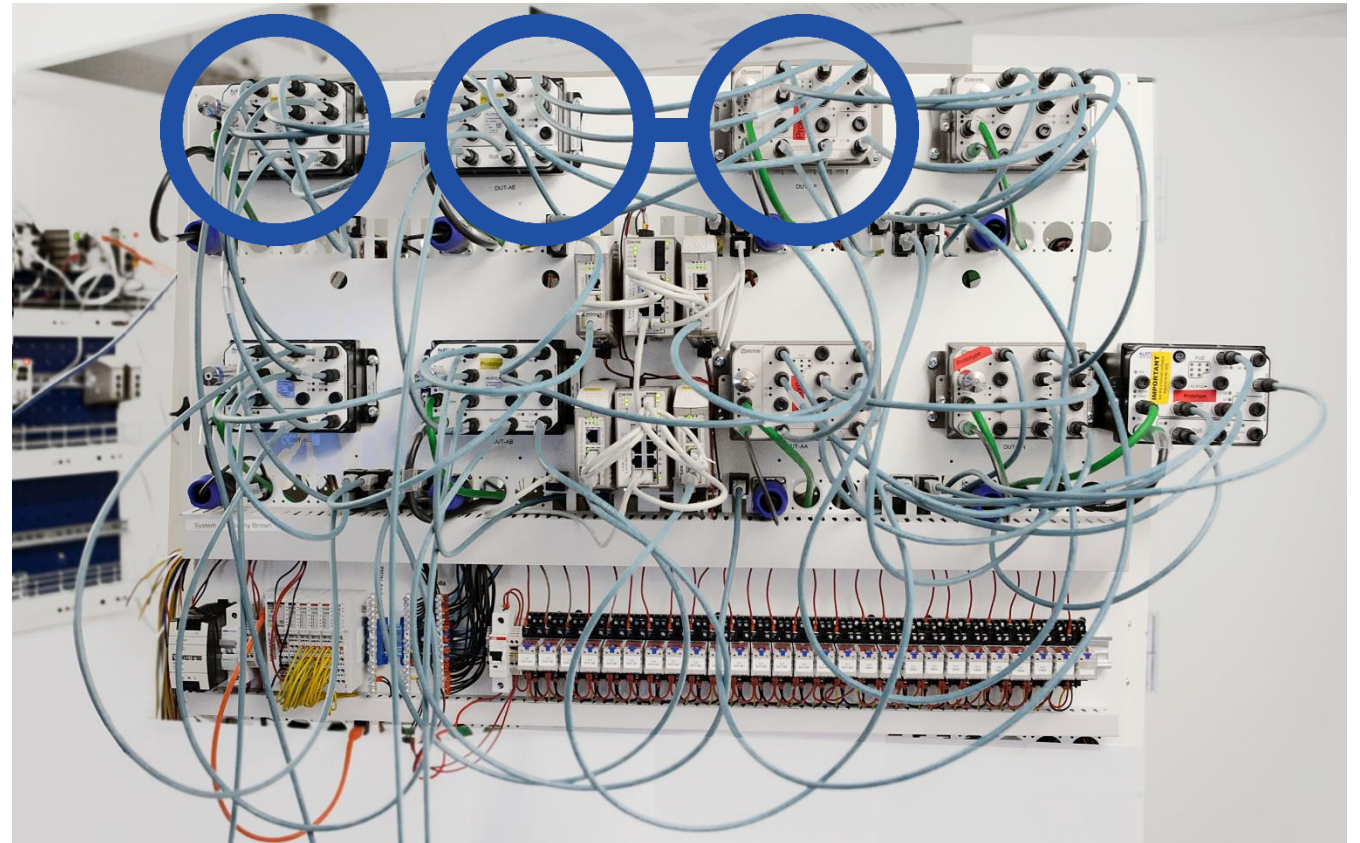


Figures from Paper A



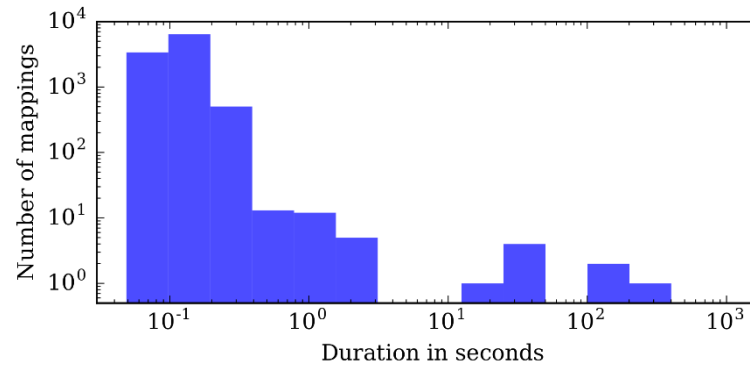
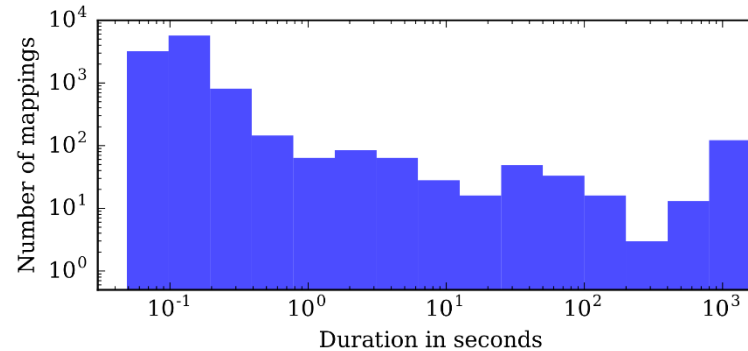
Paper B: Test Environment Assignment

- Problem
 - Performance (old solution)
 - Always the same mapping
- Approach
 - Graph Theory: Subgraph isomorphism problem
 - Search for a mapping
 - Reduce size of search space
 - Remember old mappings and search for unused parts
 - Evaluate with 10000 pairs (17 test systems, 607 test cases)



Paper B: Test Environment Assignment

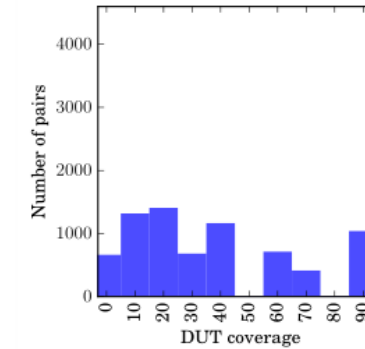
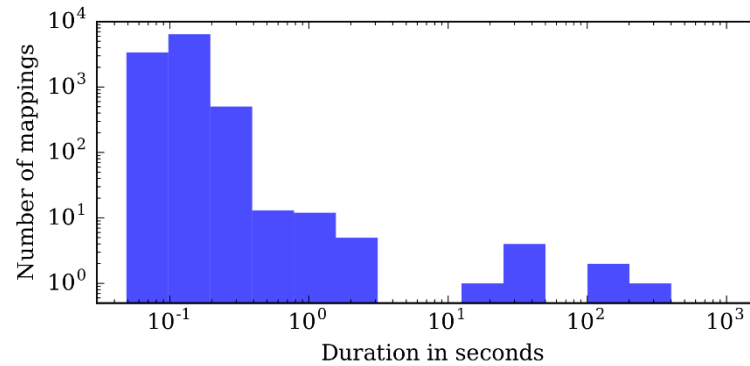
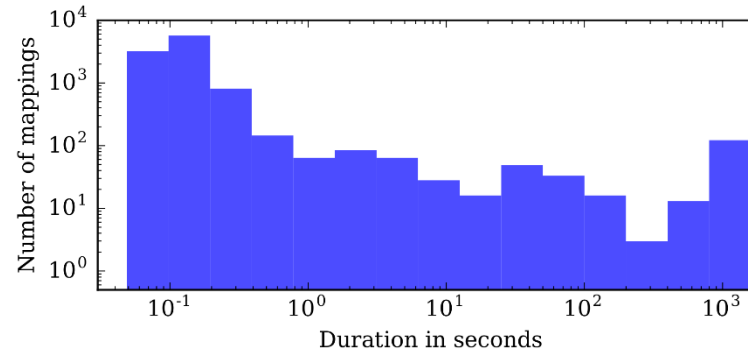
- Results
 - Speedup x 80



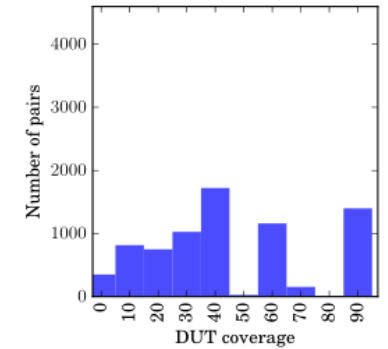
Paper B: Test Env. Ass.

- Results

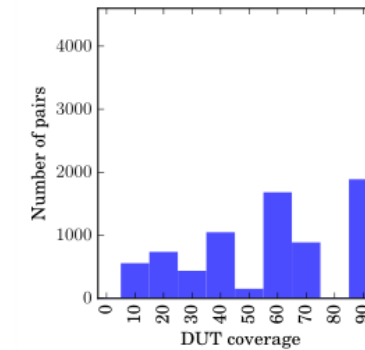
- Speedup x 80
- Coverage: from median of 33% to 100% in 5 iterations



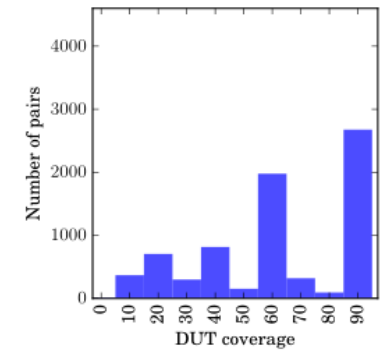
(a) 1 iteration.



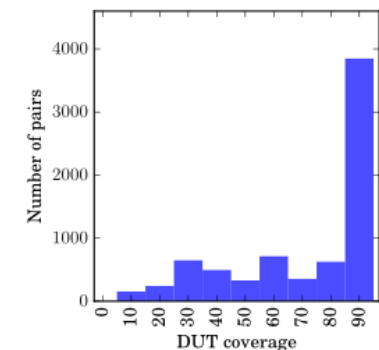
(b) 2 iterations.



(c) 3 iterations.



(d) 4 iterations.



(e) 5 iterations.

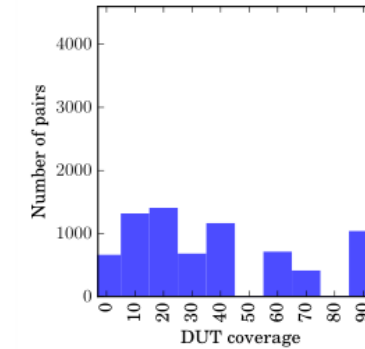
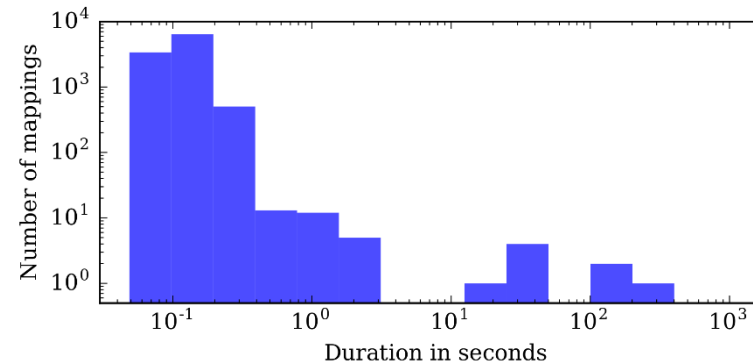
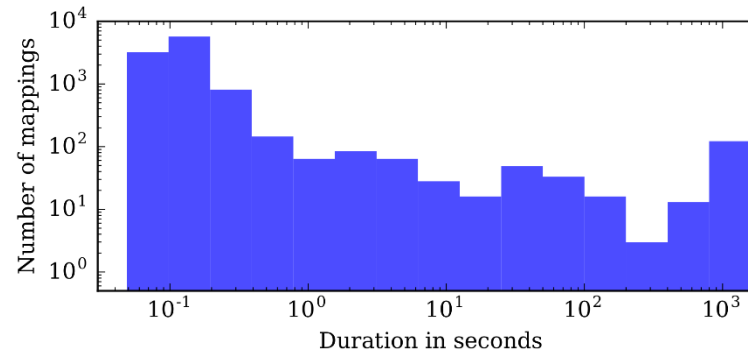
Paper B: Test Env. Ass.

- Results

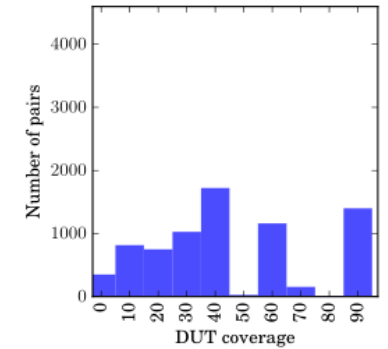
- Speedup x 80
- Coverage: from median of 33% to 100% in 5 iterations

- Contributions

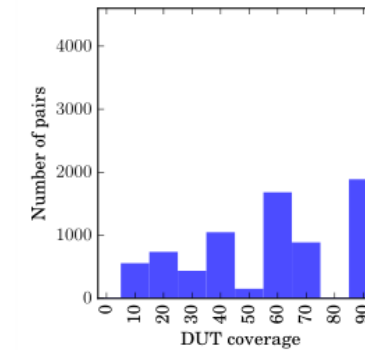
- Industry-grade implementation and evaluation
- First (?) publication with working solution for test environment assignment



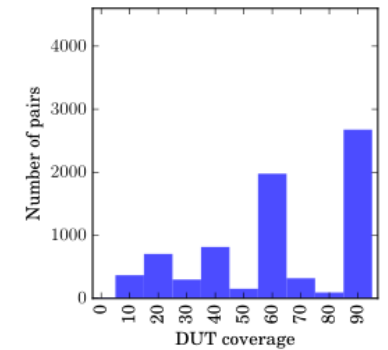
(a) 1 iteration.



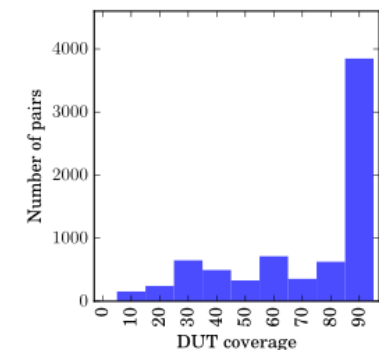
(b) 2 iterations.



(c) 3 iterations.



(d) 4 iterations.



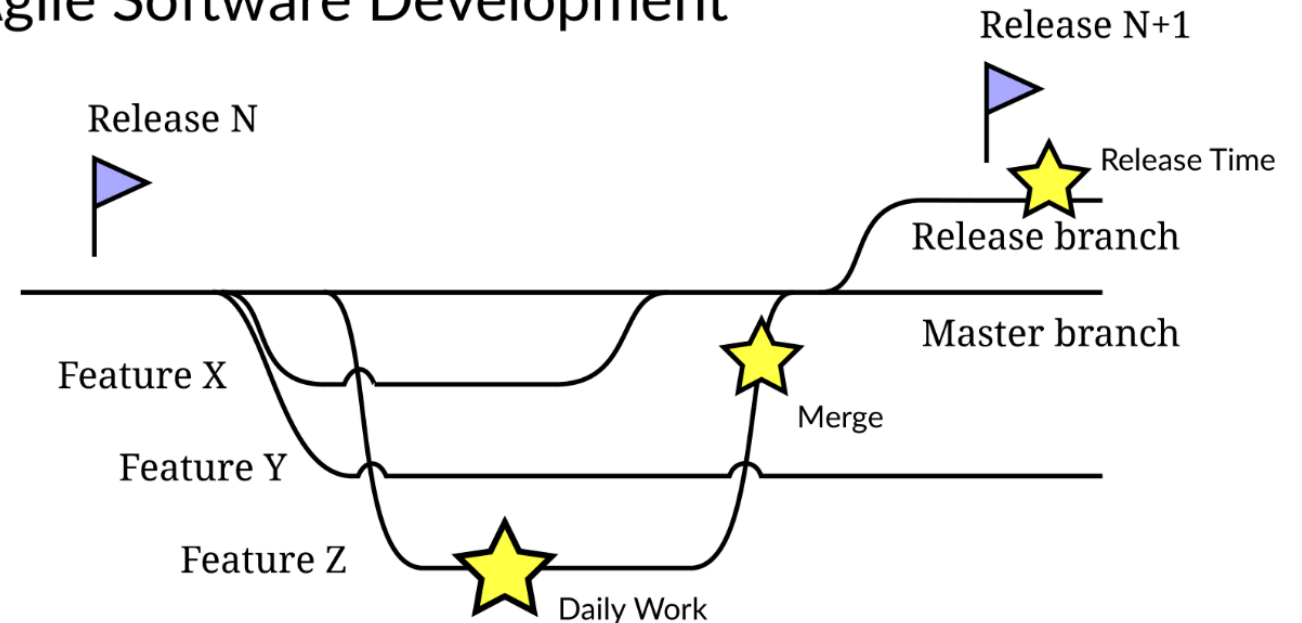
(e) 5 iterations.



Paper C: Decision-making and Visualizations

- Focus
 - How are visualizations made?
 - What is the perceived value?
 - (In daily work, at merge time, and at release time)
- Approach
 - Exploratory
 - Descriptive
 - Embedded case study

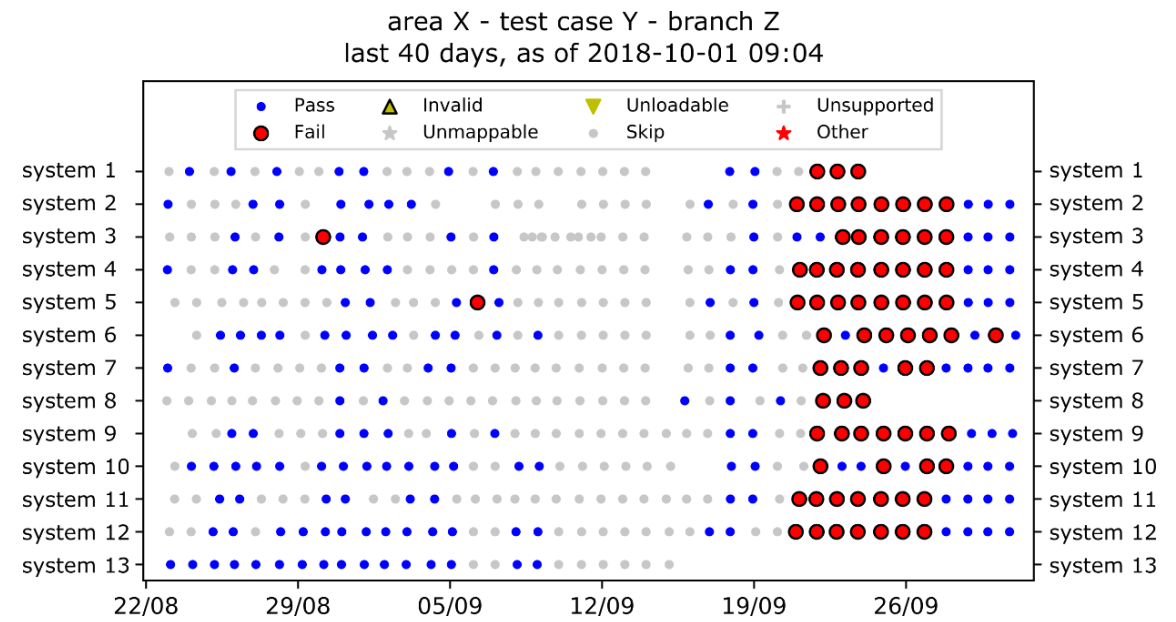
Agile Software Development



Software Development in Branches. Modified from figure in Paper C

Paper C: Decision-making and Visualizations

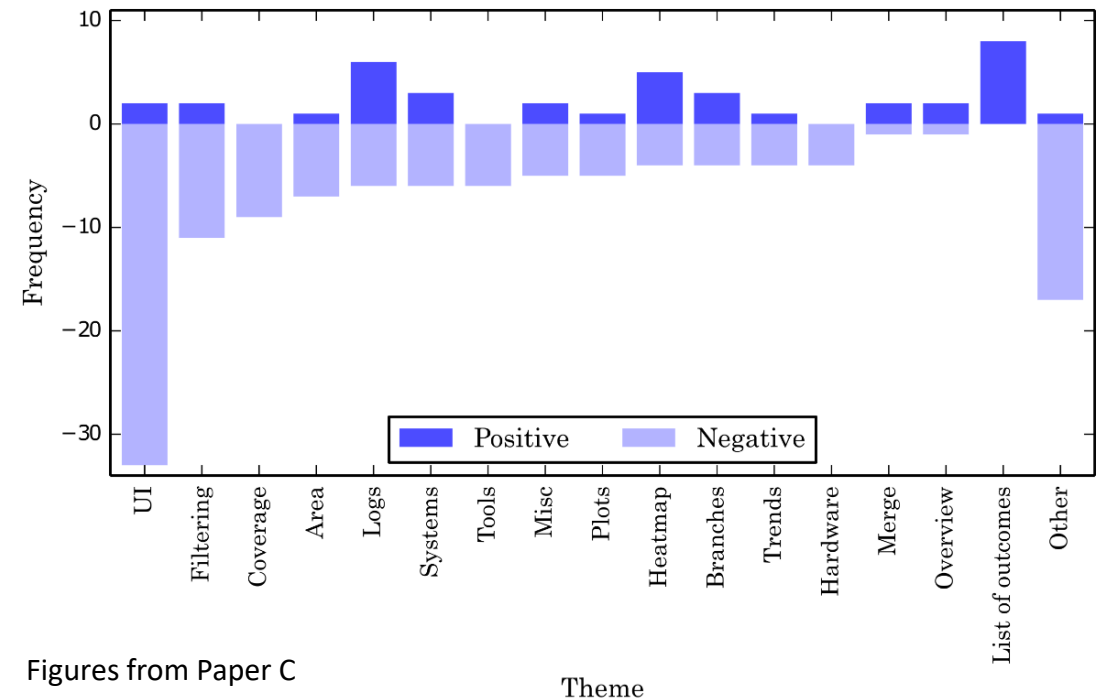
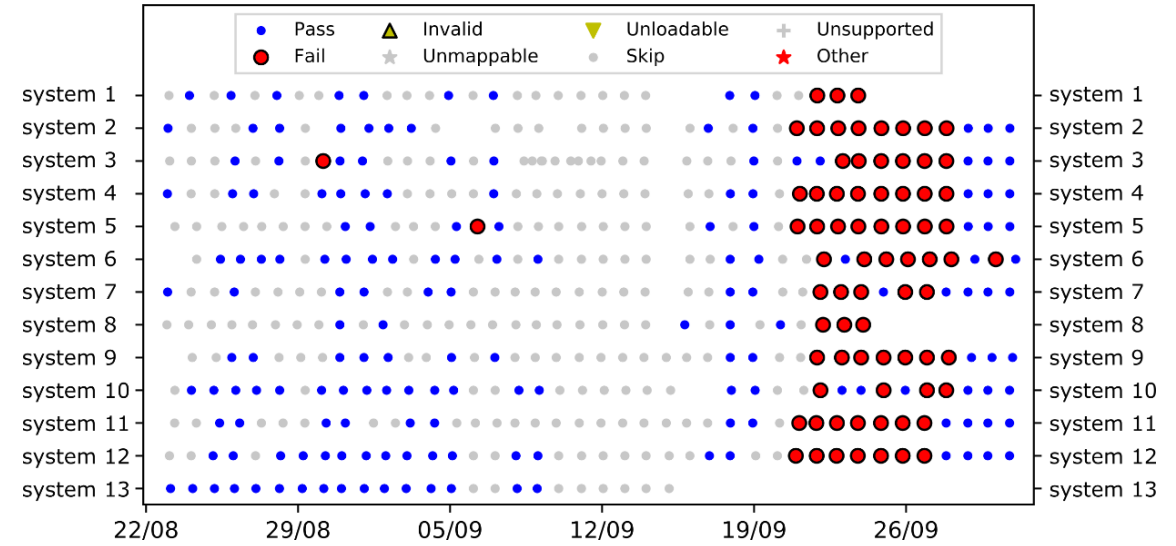
- Results and Contributions
 - Visualizations and scripts for exploring test results.



Paper C: Decision-making and Visualizations

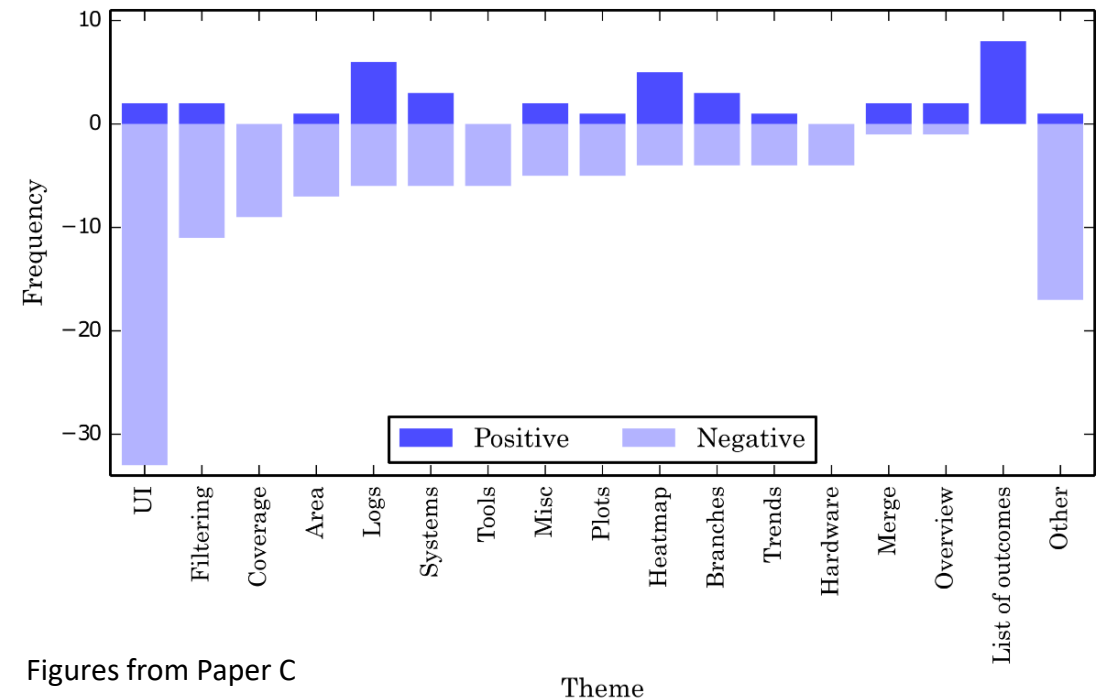
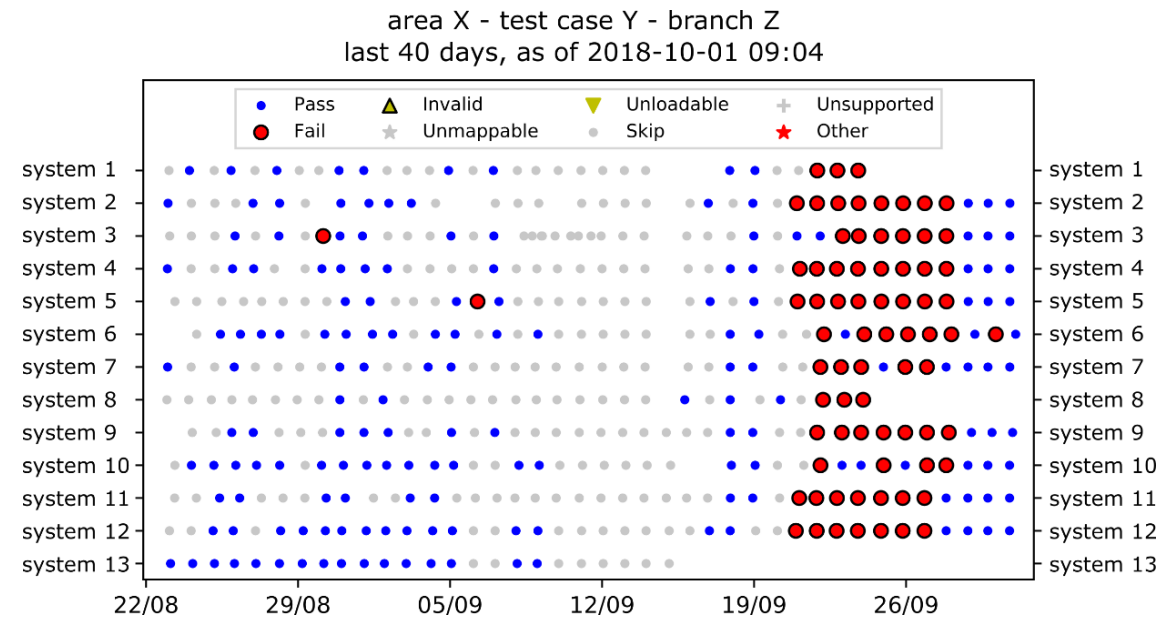
- Results and Contributions
 - Visualizations and scripts for exploring test results.
 - Positive/Negative

area X - test case Y - branch Z
last 40 days, as of 2018-10-01 09:04



Paper C: Decision-making and Visualizations

- Results and Contributions
 - Visualizations and scripts for exploring test results.
 - Positive/Negative
 - Experiences and Evaluation at Westermo
 - Six years of usage.
 - User Stories (Appendix C)



Paper D: Flow of Information SW Testing

- Question:
 - What is the overall flow of information in software testing?

Paper D: Flow of Information SW Testing

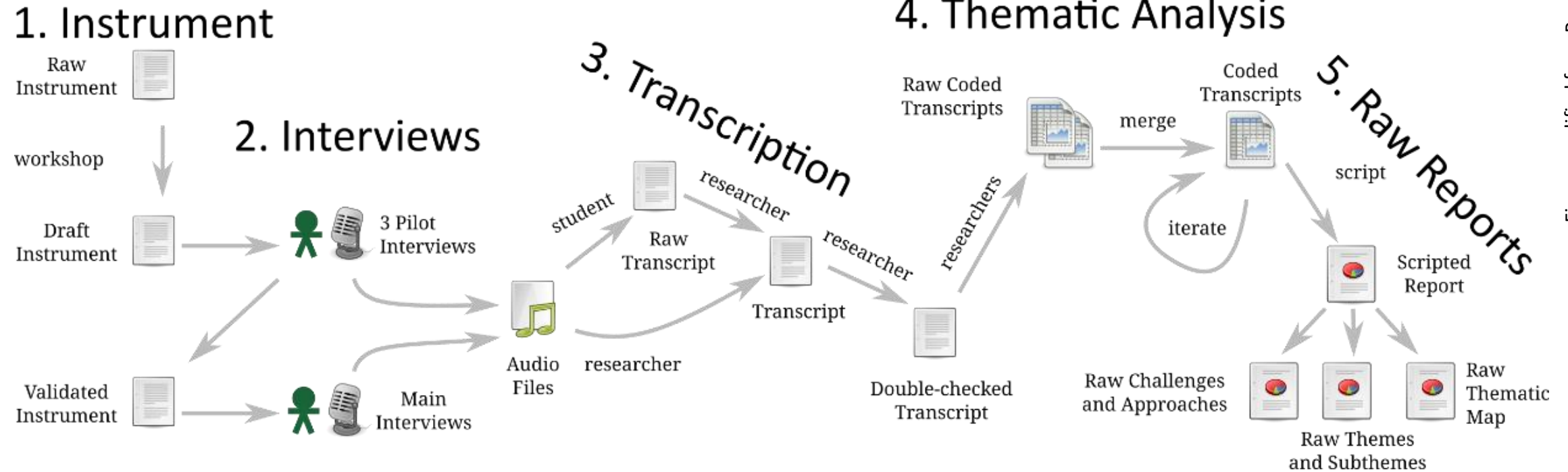


Figure modified from Paper D

- Question:

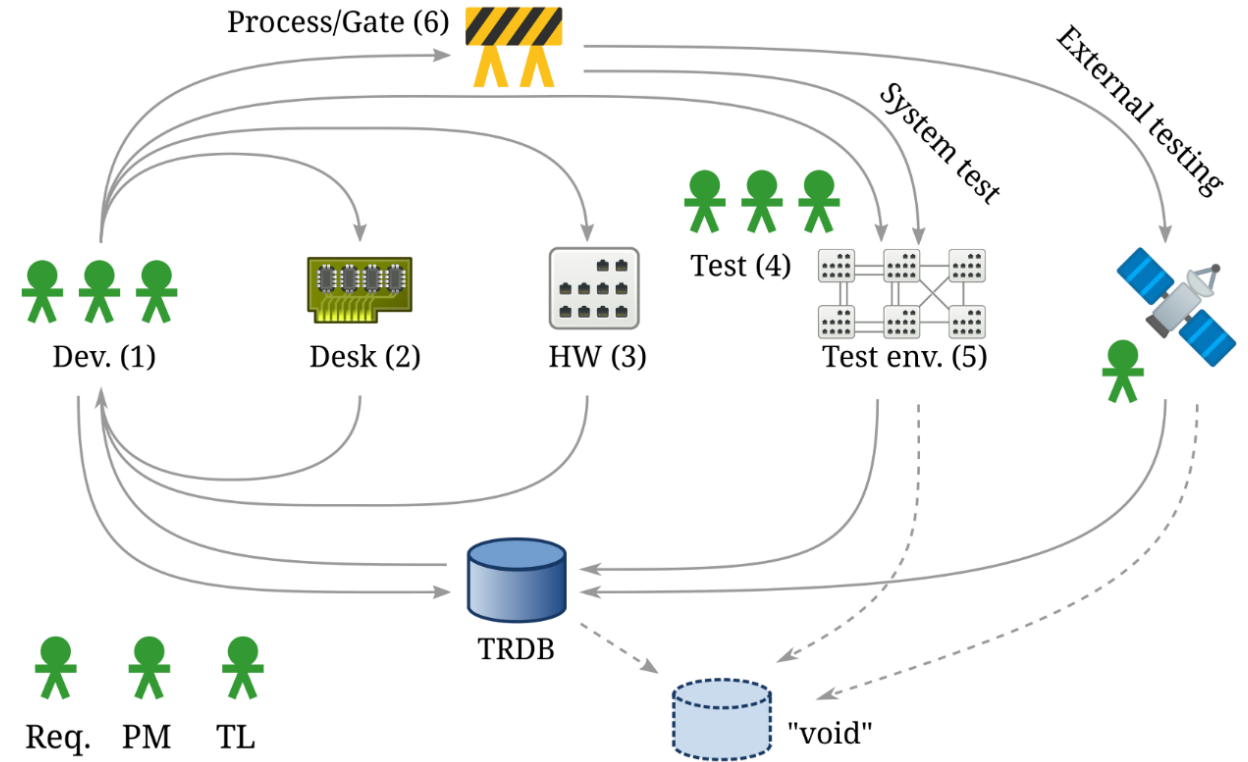
- What is the overall flow of information in software testing?

- Approach:

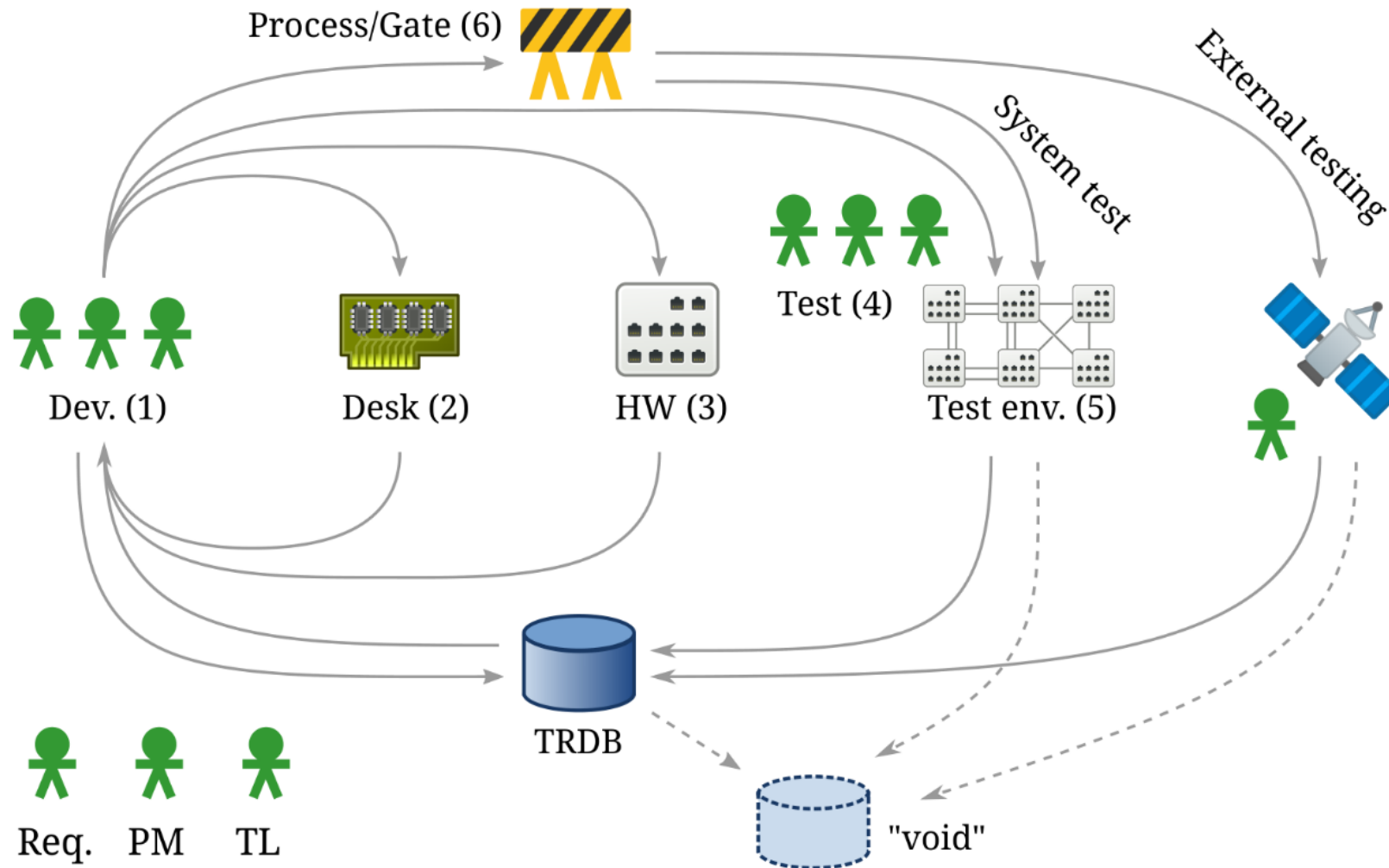
- Ambitious Interview Study
- *5 Companies, 12 Practitioners*

Paper D: Flow of Information SW Testing

- Results and Contributions:
 - Overall Information Flow Model
 - Challenges
 - Approaches
 - Themes



Paper D: Flow of Information SW Testing

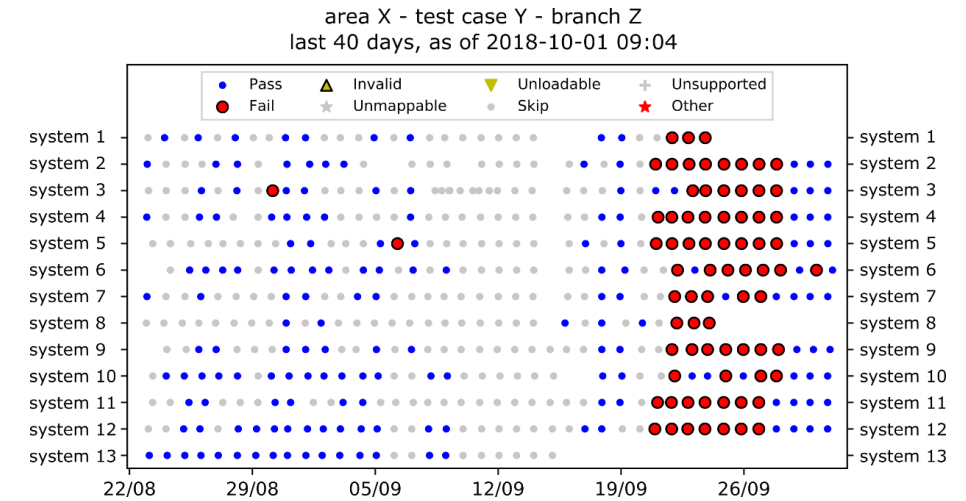




7. Future Research

Future Research

- Improve Test Results Exploration
 - Learn from Paper C and D
 - Learn from others [6, 15, 18, 26, 42, 46, 47, 61]
 - Combine implementation and evaluation
- Will it be harder to understand the results when...
 - Paper A: Not all tests run (and not in the same order)?
 - Paper B: Test cases run in different ways over time?
- Flaky Tests



A person wearing a blue jacket, dark pants, and a backpack is walking away from the camera on a wooden boardwalk. The boardwalk is made of several parallel wooden planks and leads through a field of tall, dry grass and low-lying shrubs. In the background, there is a dense forest of tall, thin trees, likely pines or spruces. The lighting suggests it might be late afternoon or early morning, with soft shadows.

8. Summary

Recap

1. Context
2. What are networked embedded systems?
3. Testing networked embedded systems
4. Research Questions
5. Research Methods
6. Research Contributions
7. Future Research
8. Summary



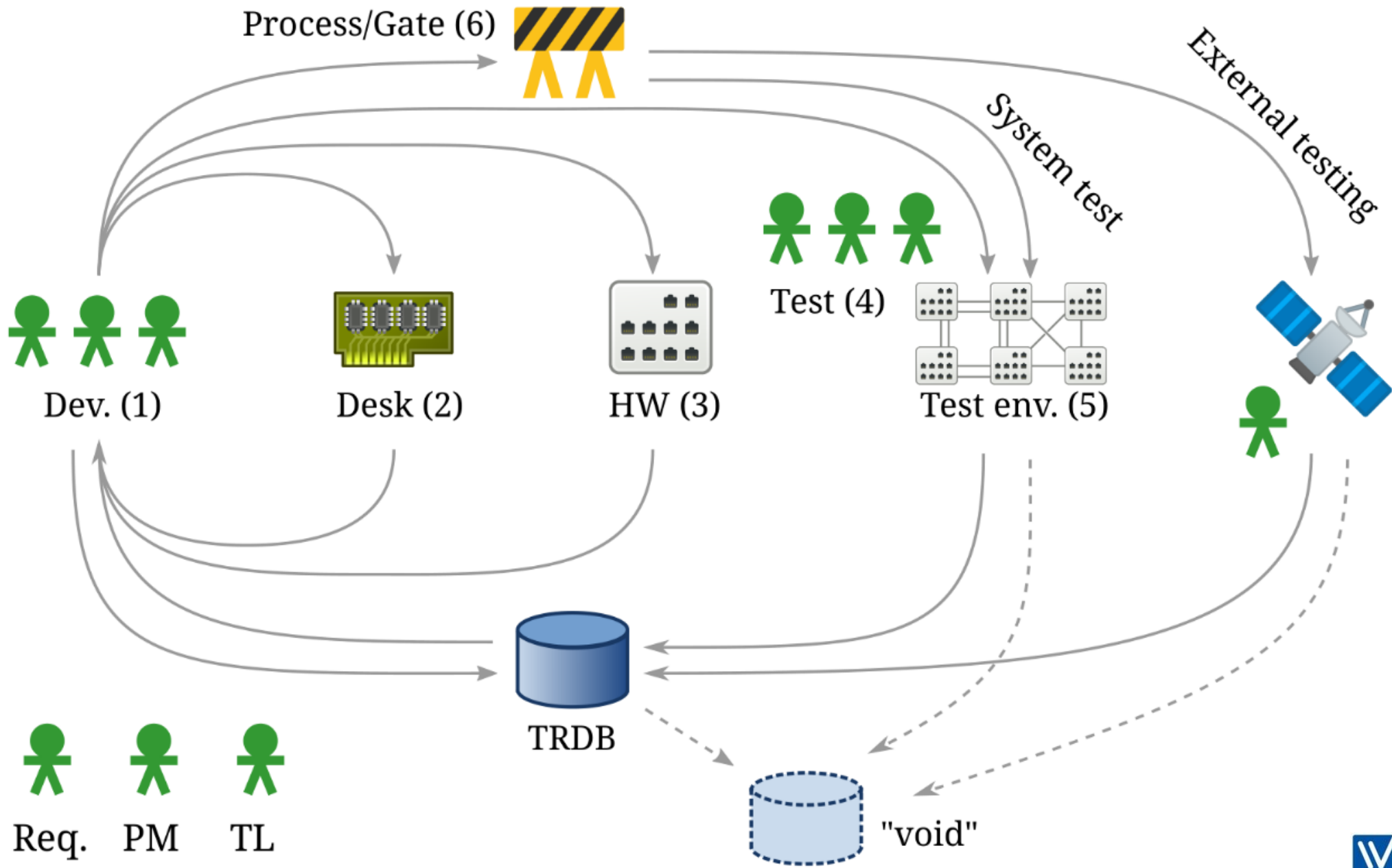
Looking back. Photo: Per Erik Strandberg

Acknowledgments

- This research financed by
 - Westermo Research and Development AB
 - Swedish Knowledge Foundation
 - 20150277 (ITS ESS-H)
 - 20160139 (TESTMINE)

- Co-authors been financed by:
 - Swedish Knowledge Foundation
 - 20130085 (TOCSYC)
 - 20130285 (Volvo Chair)
 - Swedish Research Council
 - 621-2014-4925 (EXACT)
 - European Union's Horizon 2020
 - 737494 (MegaMART2)





Automated System Level Software Testing of Networked Embedded Systems

Mälardalen University Licentiate Thesis 275

Per Erik Strandberg



MÄLARDALEN UNIVERSITY
SWEDEN