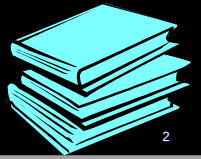
ERICSSON

A Framework for Comparing Efficiency, Effectiveness and Applicability of Software Testing Techniques

Sigrid Eldh, Ericsson AB
IDE, Mälardalens University, Sweden
Supervisors: Prof. Dr. Hans Hansson, Dr. Sasikumar Punnekkat
Colleagues: Daniel Sundmark, Anders Pettersson
Sponsored by: Ericsson AB, The Knowledge Foundation (SAVE-IT)

Leverage from what is done?!

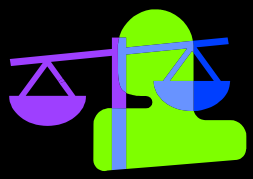
- Rothermel, Harrold, Ostrand, Weyuker, Roper, Wegener, Harman, Gilchrist, Sneed, Basili,, Elbaum, Whittaker, Offutt, Hamlet, DeMillo, Hetzel, Hierons, Holcome, Reid, Soffa, Zhu, Zeller….
- Juristo, Morena, Vegas “Reviewing 25 year of Testing Technique Experiments”
- Numerous PhD Thesis on test techniques
- 100drs of books
- However……



2

ERICSSON

Comparing Test Techniques is not easy



- Important to industry
- The problem of juxtaposing
- Results seldom transferable to other code/system/domain
- Problem of how experiments are performed
  - ANY technique find some (faults) failures
- Problem of fuzziness in techniques
  - Many variants
  - Techniques must be measured for human intervention
  - Unclear what technique or what phase can be automated (efficiently)
- BETTER GUIDELINES IS NEEDED

3

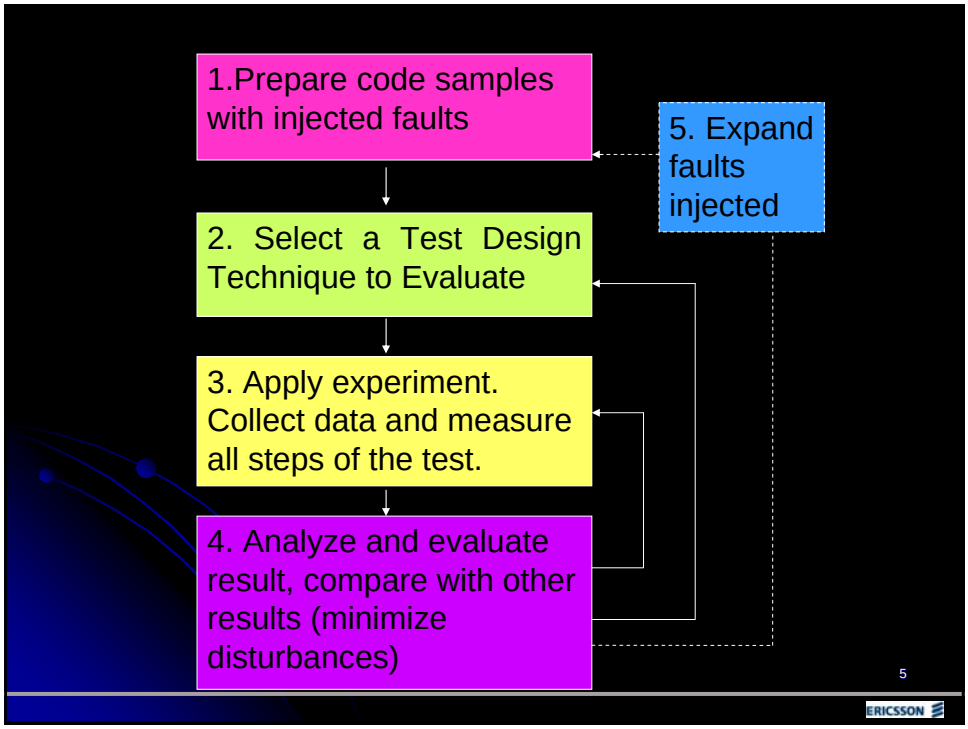
ERICSSON

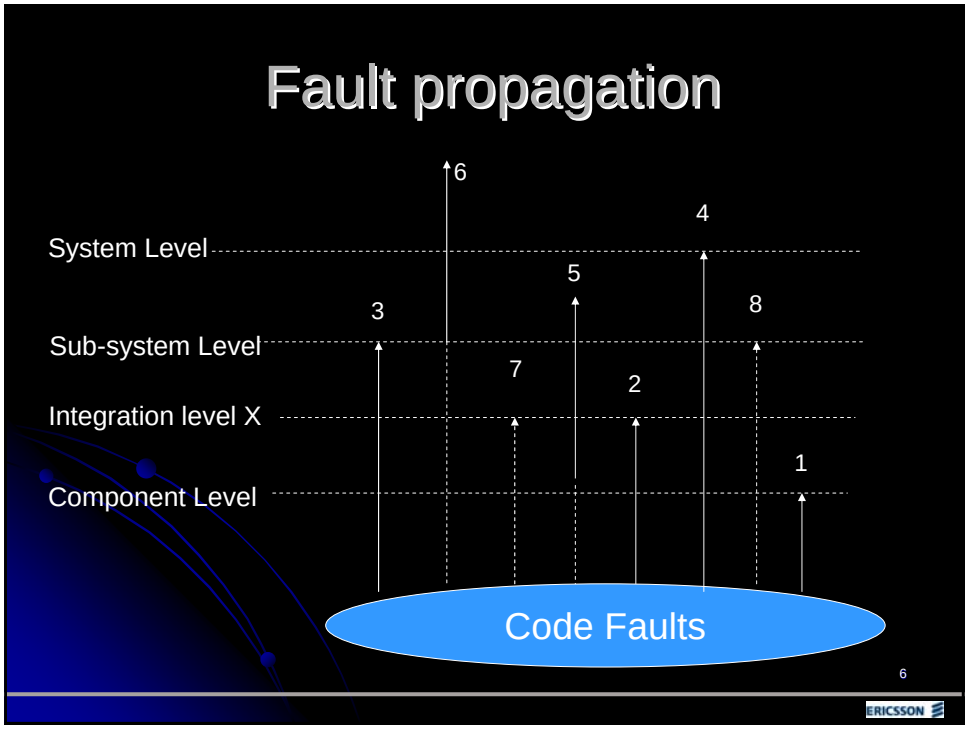
Research Questions

- **What type** and how many occurrences of failures can a particular test technique find?
- How **effectively** and **efficiently** does it find them?

- What is the **applicability** of each technique and its variants?
- How are techniques utilized at different levels in the test phases? (Which, where, when)
- How does the application of the technique determine the outcome? (Implementation, deployment)
- Are these techniques better than the “heuristic” test techniques?

ERICSSON


1.Prepare code samples with injected faults
2. Select a Test Design Technique to Evaluate
3. Apply experiment. Collect data and measure all steps of the test.
4. Analyze and evaluate result, compare with other results (minimize disturbances)
5. Expand faults injected
5
ERICSSON


Fault propagation
System Level
Sub-system Level
Integration level X
Component Level
6
4
5
3
8
7
2
1
Code Faults
6
ERICSSON

Current focus: Fault and its relation to failures

- Most classifications mix fault and failures
- Most classification does not relate to actual fault in code
  - Re-inject – Seeding - Mutations
  - Faults propagation and visibility
- Lends to make forward reasoning
  - Change code (in to simple ways)
- Alt. Use known faults in code (seldom enough for good measurements)
- Classification example: Bus fault, interface fault

7

ERICSSON

The challenge with Faults and Failures (preparing the code)

- Can the fault be generalized to be used everywhere (or how bound is it to the domain/semantics is the fault)?
- What is a good and true fault (failure?) classification sufficient to juxtapose any two techniques?
- What constitutes sufficient information for a designer to be able to classify a fault (from analyzing a failure)?
- What is the scalability and impact of faults?
- How does faults propagate to failures, and what are appropriate levels of testing to localize them?

8

ERICSSON

Test Case Design Technique Analysis for Automation

- Understand, define, classify technique (variations)
- A TC and variants must be separated to phases to understand its possibilities
  1. Test case creation
  2. Test case selection
  3. Test case execution
  4. Test case result analysis
- For each phase, + - , human intervention, dependencies etc
- Fault (and failure) detection possibility
- Level to find fault/failure (dependencies)
- Possibilities and problems of automation of each phase

9

ERICSSON

Measurements of Test Case Design Techniques

- Not only the simple and obvious
  - How many faults found
- All aspects in process are measured
  - Qualitatively (e.g. ease of understand, apply)
  - Quantitative (e.g. actual time, # faults)
  - Automation (implementation) also evaluated
    - Probably many ways to solve problems
- We are measuring *efficiency*, *effectiveness* and *applicability*

10

ERICSSON

More Challenges

- Can TT be assigned to a particular fault or class of faults?
- The efficiency of different techniques depends on where and how they are applied.
- Does a evaluation result scale from code sample to industrial system? When and why?
- Automatic generation time consuming, feasible?
- Reduction of the test set to efficiently instrument and measure coverage. Completeness or cut, prune or determine when the suite is sufficient
- The way *how* execution is automated defines how useful the automation will be

Efficiency

- Actual time, i.e. planning, implementation and execution (manual & automation), (calendar-time and estimated time) for each phase (hours/days) (Quantitative)
- Time to detect faults and/or failures, and also time to identify the fault type (minutes/hours) (Quantitative)
- How long time it takes to find the first fault or failure in minutes (Quantitative)
- The subjects own judgment of every task in the process (Easy, difficult, poses secondary problems etc.). The assumption is, what is easy is also fast. (Qualitative)
- The time to manually create test cases (for one, the first, and many variants) (Quantitative)
- How many unique test cases, and instances of the test case is created (number per test case/number of variants) (Quantitative)

Effectiveness

- Absolute numbers of how many of the seeded (and other) faults were found (isolated) compared to injected faults. (% faults detected, % faults isolated). (Quantitative)
- For each fault found, identify what type, how many of the faults are isolated, and faults severity. % faults detected/type, % faults isolated/type % (Quantitative) faults of each severity (A, B, C) (Qualitative/Quantitative).
- Estimation of “coverage” in % and measured where possible, dataflow and control flow coverage using the technique, as a support to the effectiveness of the test case suite. (Quantitative/Qualitative)

ERICSSON

Applicability

- In what phase faults are found (distributed) over time (quantitative)
- The subject’s own judgment of every task in the process (Easy, difficult, poses secondary problems etc) (Qualitative)
- The ease of learning the technique (Easy, difficult, poses secondary problems etc) (Qualitative)
- Levels (code, component, integration, sub-system, system) where the technique is possible to use (Qualitative)
- The generality of the technique, empirically studied
  - What context (OS, hardware, domain, etc.)
  - Language mapping and constraints (C, C++ Java, and version/compiler)
- Number of variants of the technique within each scope. (Quantitative)
- The evaluation of the applicability of automation of the technique, which is a qualitative assessment. Measurements are ranging from (bad, slow and ineffective) on a floating scale to (good, fast, and perceived effective) (Qualitative)
- Evaluation of the entire process (Qualitative)

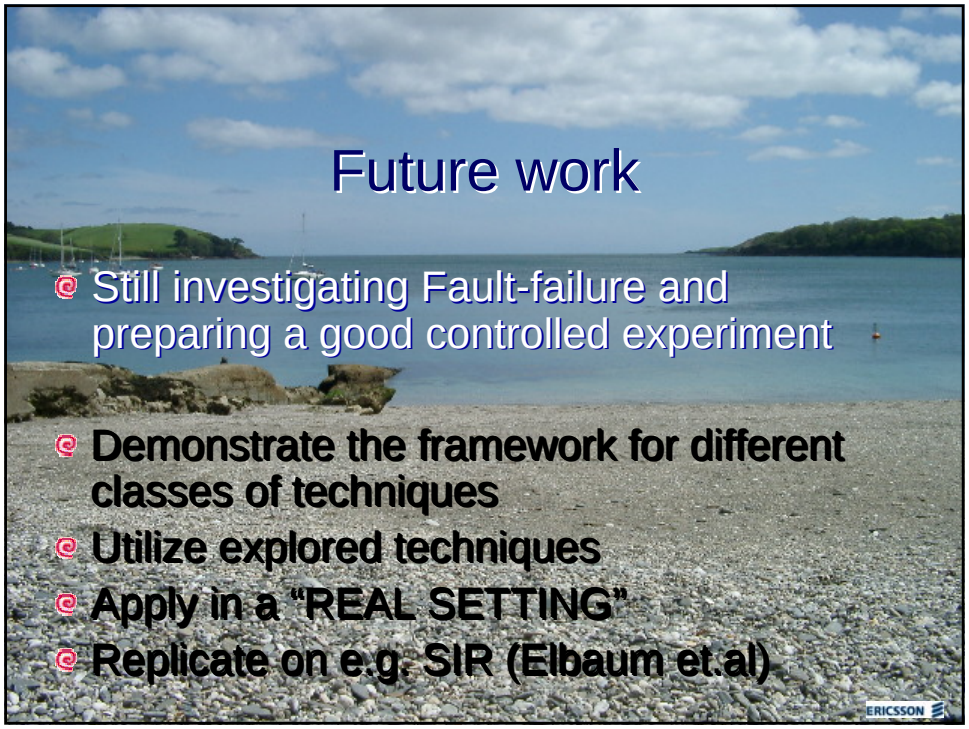
ERICSSON

Creating Guidelines for usage of test techniques for Industry

Test case design Technique	Level	Efficient	Effective	Applicability	Failure relations
Random Input*	All	Execution coverage (yes) Input coverage (no)	Depends on number of values 1 TC	Diminish space Input select Nr 4 (eval.) Implement dependent…..	Input Time Stress
Fast Anti Random	All	Mostly better on x	More than RI	..	--
CTE	All but* Protocol	Yes*	Yes	Human decision	Functional (depending)
Evolutionary	All?	Yes*	Yes	Search function	..

15

ERICSSON



ERICSSON

TAKING YOU FORWARD