Strategic and Operational Decision Support in

Software Quality

Management

Prof. Per Runeson

Who am I?

- Professor in Software Engineering, Lund University
- Leader for the Software Engineering Research Group at LU and the EASE industrial excellence center
- Sabbatical at North Carolina State University, 2011-12
- Sony Ericsson, part time 2010
- LU since 1998
- Q-Labs 1991-1998

EASE Embedded Applications Software Engineering

SWELL - Swedish V&V Excellence

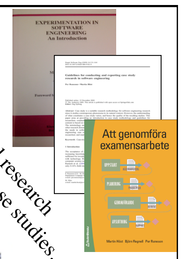
Research interests

1995

- Reliability testing
- Inspection methods
- System validation
- Agile management
- Test management
- Unit testing
- Regression testing
- Product line testing

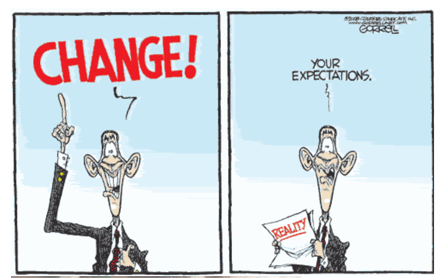
2011

Empirical research – surveys, case studies, experiments



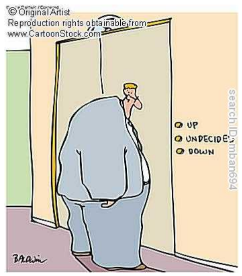
ICST

Expectations?



My goal

Provide some insight to **empirical evidence** available for **strategic and operational** decision support on **software quality**.



Outline

- **Definitions**
- Strategic decision support
- Operational decision support
- Making change

Strategic vs. operational

- Long-term
- One-off
- Functional organization

- Mid to short-term
- Continuous
- Project organization

Example Strategic Questions

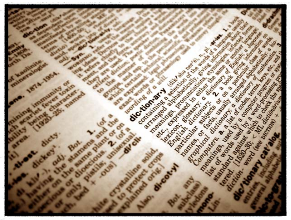
- Which quality assurance activities, like inspection and testing, are conducted when, by whom, and to what extent?
- Which testing should be automated first, and what should not be automated?
- How much testing to spend on the products vs. testing the platform?

Example Operational Questions

- How many defects are found in unit testing?
- How many test cases remain to be run in system test?
- “When to stop testing" is an issue for every project manager.

Terms in empirical software engineering

- Case study
- Evidence
- Experiment
- Mapping study
- Survey
- Systematic review



Outline

- Definitions
- **Strategic decision support**
- Operational decision support
- Making change

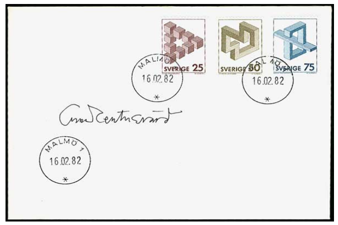
Test strategy

- What…
- When…
- By whom..
- To what extent..

… to be **tested**…
… and why?

The testing paradox

Testing purpose
- Find faults
- Demonstrate quality

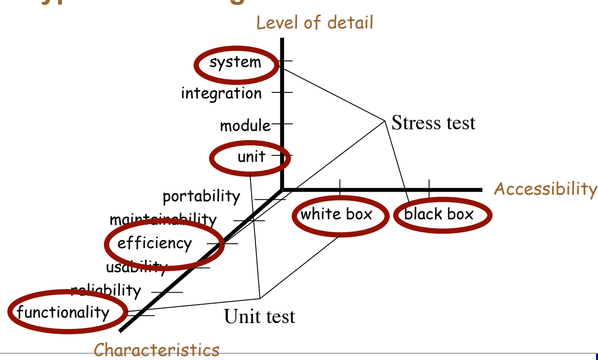


What is a failed test?

- One finds a fault?

- One that fails to reveal a fault?

Types of Testing



Strategic decision support based on systematic literature reviews

Systematic literature reviews 1(3) [Kitchenham 2007]

Planning the review

- Identification of the need for a review
- Commissioning a review
- Specifying the research question(s)
- Developing a review protocol
- Evaluating the review protocol

Systematic literature reviews 2(3) [Kitchenham 2007]

Conducting the review

- Identification of research
- Selection of primary studies
- Study quality assessment
- Data extraction and monitoring
- Data synthesis

Lund University / Faculty of Engineering/ Department of Computer Science / Software Engineering Research Group

Systematic literature reviews 3(3) [Kitchenham 2007]

Reporting the review

- Specifying dissemination mechanisms
- Formatting the main report
- Evaluating the report

Lund University / Faculty of Engineering/ Department of Computer Science / Software Engineering Research Group

Example SLR: What do we know about defect detection methods?


feature

defect detection

What Do We Know about Defect Detection Methods?

Per Runeson, Carina Andersson, and Thomas Thelin, *Lund University*
Anneliese Andrews, *University of Denver*
Tomas Berling, *Ericsson Microwave Systems*

A survey of defect detection studies comparing inspection and testing techniques yields practical recommendations: use inspections for requirements and design defects, and use testing for code.

Detecting defects in software product development requires serious effort, so it's important to use the most efficient and effective methods. *Evidence-based software engineering* can help software practitioners decide which methods to use and for what purpose.[1] EBSE involves defining relevant questions, surveying and appraising available empirical evidence, and integrating and evaluating new practices in the target environment.

Strategic question: testing or inspection?
Available evidence, comparing testing and inspections (2006):

- 10 experiments
- 2 case studies

Scale of experiments: 60-2400 LOC
Scale of case studies: 250-6300 defects

of Computer Science / Software Engineering Research Group

What do we know about defect detection methods?


feature defect detection

What Do We Know about Defect Detection Methods?

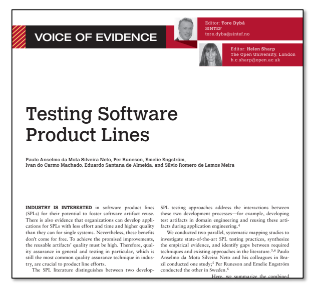
- requirements defects, use inspection (no empirical evidence)
- design specification defects, inspections are more efficient and more effective than functional testing.
- code defects, functional or structural testing is more effective or efficient than inspection in most studies.
- effectiveness is low;
  25 to 50 % of an artifact's defects found in inspection,
  30 to 60 % found using testing.
- efficiency 1 - 2.5 defects per hour

Group work

- Glance through ”What do we know…”
- How valid are the recommendations?
- For which companies?
- What is the alternative?

Strategic decision support based on mapping study

Example Mapping Study: Testing Software Product Lines


VOICE OF EVIDENCE

Testing Software Product Lines

- Strategic questions: how to test a product line?
- Available evidence: 64+45 (=76 unique) papers
- Type of evidence: 40-52% solution proposals

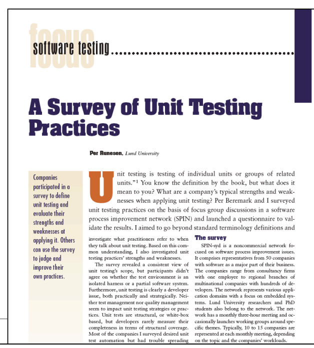
Testing Software Product Lines

- SPLT is a ”discussion topic”

Topics:
- Testing strategy
- Testing levels
- Product variability and traceability
- Effort reduction
- Test organization and process

Strategic decision support based on survey and benchmarking

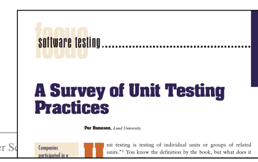
Example survey: Industry practice on Unit Testing



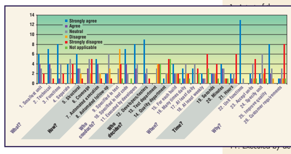
- Benchmarking against industry peers
- Focus group format
- Non-competing companies

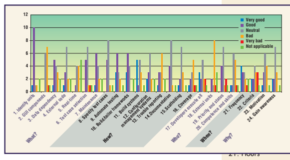
Industry practice on Unit Testing

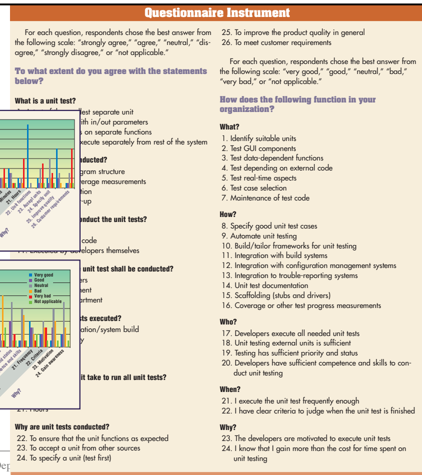
- What?
  - Technical!
  - Specified or not?
- How?
  - Structure-based
  - Not formally
- Who?
  - Developer (team)
  - Not test or QA
- When?
  - Each build/day/week
  - Takes seconds to hours to run
- Why?
  - Assuring functionality



Benchmarking







Questionnaire Instrument

For each question, respondents chose the best answer from the following scale: "strongly agree," "agree," "neutral," "disagree," "strongly disagree," or "not applicable."

To what extent do you agree with the statements below?

What is a unit test?

…lest separate unit
…ith in/out parameters
…s on separate functions
…xecute separately from rest of the system

…nducted?

…gram structure
…erage measurements
…tion
…-up

…nduct the unit tests?

… code
… developers themselves

… unit test shall be conducted?

…ers
…ent
…artment

…sts executed?

…ation/system build
…y

…it take to run all unit tests?

21. Hours

Why are unit tests conducted?

22. To ensure that the unit functions as expected
23. To accept a unit from other sources
24. To specify a unit (test first)
25. To improve the product quality in general
26. To meet customer requirements

For each question, respondents chose the best answer from the following scale: "very good," "good," "neutral," "bad," "very bad," or "not applicable."

How does the following function in your organization?

What?

1. Identify suitable units
2. Test GUI components
3. Test data-dependent functions
4. Test depending on external code
5. Test real-time aspects
6. Test case selection
7. Maintenance of test code

How?

8. Specify good unit test cases
9. Automate unit testing
10. Build/tailor frameworks for unit testing
11. Integration with build systems
12. Integration with configuration management systems
13. Integration to trouble-reporting systems
14. Unit test documentation
15. Scaffolding (stubs and drivers)
16. Coverage or other test progress measurements

Who?

17. Developers execute all needed unit tests
18. Unit testing external units is sufficient
19. Testing has sufficient priority and status
20. Developers have sufficient competence and skills to conduct unit testing

When?

21. I execute the unit test frequently enough
22. I have clear criteria to judge when the unit test is finished

Why?

23. The developers are motivated to execute unit tests
24. I know that I gain more than the cost for time spent on unit testing

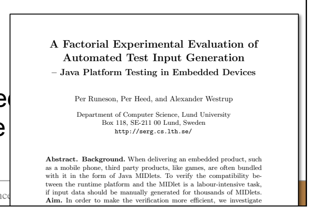
Strategic decision support based on specific experiment

Background

- Mobile phones come with third party MIDlets, e.g. games
- Verifying compatibility with Java platform is an extensive task, even with test scripting

Aim

- Investigate different automate… methods which do not require…



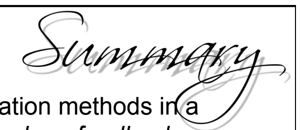
A Factorial Experimental Evaluation of Automated Test Input Generation
– Java Platform Testing in Embedded Devices

Per Runeson, Per Heed, and Alexander Westrup

Department of Computer Science, Lund University
Box 118, SE-211 00 Lund, Sweden
http://serg.cs.lth.se/

Abstract. Background. When delivering an embedded product, such as a mobile phone, third party products, like games, are often bundled with it in the form of Java MIDlets. To verify the compatibility between the runtime platform and the MIDlet is a labour-intensive task, if input data should be manually generated for thousands of MIDlets. **Aim.** In order to make the verification more efficient, we investigate

Empirical study



- **Method**: Evaluate input generation methods in a factorial design experiment: *random, feedback based, with and without a startup sequence*
- **Results**:
  - Pure *random* or *feedback* based is not enough
  - The *startup sequence* improves. The feedback method is somewhat better, but at the cost of real-time measurements, which decreases the run speed of the tests.
- **Conclusion**: The *random method with startup sequence* is the best trade-off in the current setting

Outline

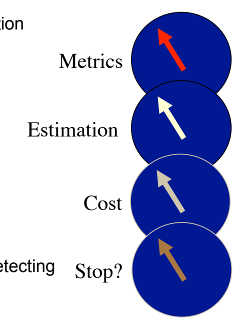
- Definitions
- Strategic decision support
- **Operational decision support**
- Making change

Operational decisions support – test management

- Monitoring
  - Check status
  - Reports
  - Metrics
- Controlling
  - Corrective actions

Test Monitoring

- Status
  - Coverage metrics
  - Test case metrics: development and execution
  - Test harness development
- Efficiency / Cost metrics
  - How much time is spent?
  - How much money is spent?
- Failure / Fault metrics
  - How much is accomplished?
  - What is the quality status of the software?
- Effectiveness metrics
  - How effective is the testing techniques in detecting defects?

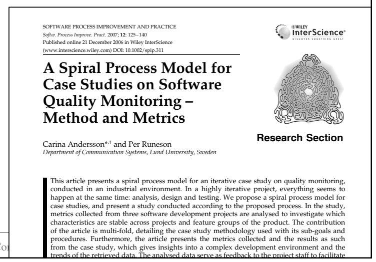


Operational decision support based on specific model

Goal

- Quantified management decision support
- Understanding of observed phenomena

”At delivery date – how many defects remain?”
”How come defects are found later for A than B”


SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. 2007; **12**: 125–140
Published online 21 December 2006 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/spip.311

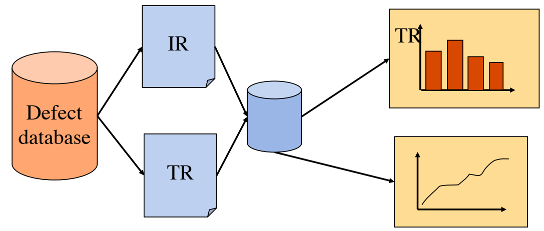
A Spiral Process Model for Case Studies on Software Quality Monitoring – Method and Metrics

Research Section

Carina Andersson*,† and Per Runeson
Department of Communication Systems, Lund University, Sweden

This article presents a spiral process model for an iterative case study on quality monitoring, conducted in an industrial environment. In a highly iterative project, everything seems to happen at the same time: analysis, design and testing. We propose a spiral process model for case studies, and present a study conducted according to the proposed process. In the study, metrics collected from three software development projects are analysed to investigate which characteristics are stable across projects and feature groups of the product. The contribution of the article is multi-fold, detailing the case study methodology used with its sub-goals and procedures. Furthermore, the article presents the metrics collected and the results as such from the case study, which gives insights into a complex development environment and the trends of the retrieved data. The analysed data serve as feedback to the project staff to facilitate

Procedures



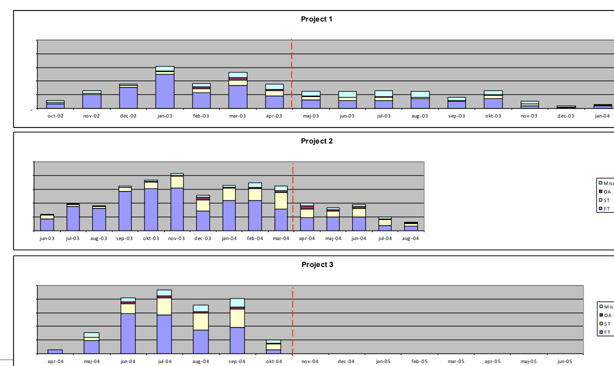
From IR:
- **Date found**
- **Reporter**

From TR:
- **Module**
- **Status**

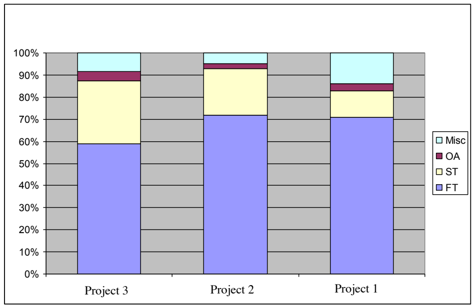
Data

- 3 projects
  - Different views
    - Complete project
    - Function groups
- Dimensions
  - Time
  - Test activity
    - Function test
    - System test
    - Operator acceptance
    - Miscellaneous
  - Function groups

Calendar time view



Distribution (FT, ST, OA, Misc)



Data used for prediction

- Until Alpha: ~80% of FT faults reported
- The distribution for (→ RTL)

FT	67%
ST	25%
OA	3%
Misc	5%

Prediction (example)

- At Alpha: 100 faults reported in FT
- To expect at RTL:
  - Total: 100/0.8/0.67 = 187.5

  - FT: 187.5*0.67 ≈ 125
  - ST: 187.5*0.25 ≈ 47
  - OA: 187.5*0.03 ≈ 6
  - Misc: 187.5*0.05 ≈ 9

Group work

- How valid are the recommendations?
- For which companies?
- For how long?

Defect content estimation: Basic approach

Lincoln-Peterson estimation model

N – estimated number of faults

n_1, n_2 – faults found by reviewer 1 and 2 respectively

m_2 – fault fund by both

$$\hat{N} = \frac{n_1 \times n_2}{m_2}$$

More advanced Capture-Recapture Models

- Four basic models used for inspections
  - Degree of freedom
- Prerequisites for all models
  - All reviewers work independently of each other
  - It is not allowed to inject or remove faults during inspection

Synthesized evaluations of CRC

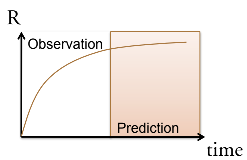
1. Most estimators underestimate,
2. Mh–JK is the best estimator for software inspections,
3. Mh–JK is appropriate to use for four reviewers and more,
4. DPM is the best curve fitting method, and
5. Capture–recapture estimators can be used together with PBR.

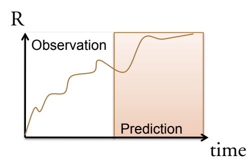
Operational decision support based on general models

(Software Reliability Growth)

Measurements for software reliability

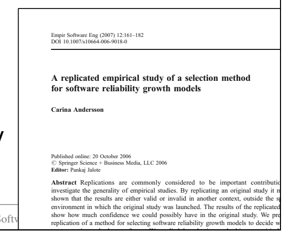
- MTBF = Mean Time Between Failure
- R = Probability for failure-free execution (under specified conditions and time)





Software reliability growth models

- Selection of appropriate models
  - 2 concave models: Goel-Okumoto, Yamada exponential
  - 2 S-shaped models: Delayed S-shaped, Gompertz
- Evaluated in terms of
  - Prediction stability
  - Curve fit
- Applied on function test data and system test data separately


Empir Software Eng (2007) 12:161–182
DOI 10.1007/s10664-006-9018-0

A replicated empirical study of a selection method for software reliability growth models

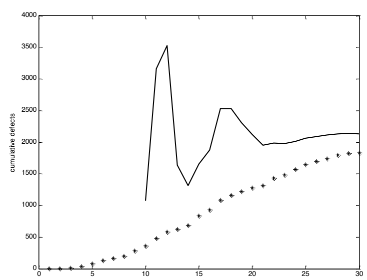
Carina Andersson

Published online: 20 October 2006
© Springer Science + Business Media, LLC 2006
Editor: Pankaj Jalote

Abstract Replications are commonly considered to be important contributi
investigate the generality of empirical studies. By replicating an original study it m
shown that the results are either valid or invalid in another context, outside the s
environment in which the original study was launched. The results of the replicated
show how much confidence we could possibly have in the original study. We pre
replication of a method for selecting software reliability growth models to decide w

Lund University / Faculty of Engineering/ Department of Computer Science / Softv

Predictions per week

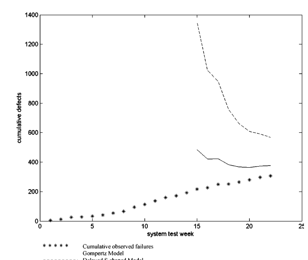


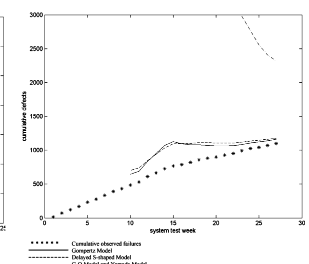
Predicted values from week 11

Gompertz model

Lund University / Faculty of Engineering/ Department of Computer Science / Software Engineering Research Group

Other models





Lund University / Faculty of Engineering/ Department of Computer Science / Software Engineering Research Group

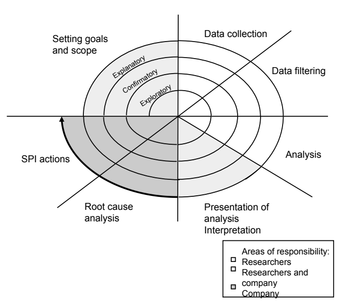
Group work

- How valid is the information gained using quantiative models?
- How relevant are they?
- What is the alternative?

Outline

- Definitions
- Strategic decision support
- Operational decision support
- **Making change**

Test process monitoring




SOFTWARE PROCESS IMPROVEMENT AND PRACTICE

A Spiral Process Model for Case Studies on Software Quality Monitoring – Method and Metrics

Carina Andersson* † and Per Runeson

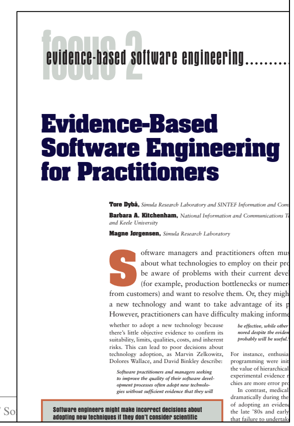
Department of Communication Systems, Lund University, Sweden

Research

1. INTRODUCTION

EBSE

1. Convert problem into a question
2. Search the literature for the best available evidence
3. Critically appraise the evidence
4. Integrate with customer's values and circumstances
5. Evaluate performance and seek ways to improve it

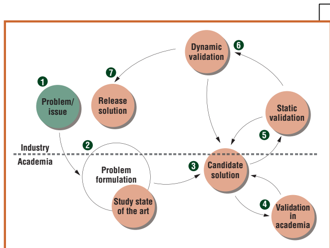

focus 2 evidence-based software engineering

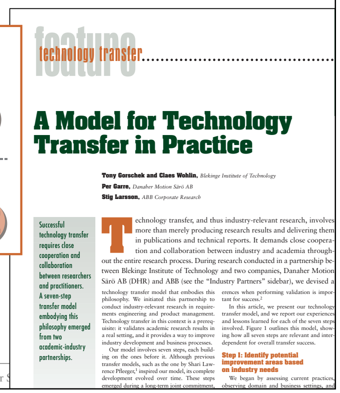
Evidence-Based Software Engineering for Practitioners

Tore Dybå, Barbara A. Kitchenham, Magne Jørgensen

Software engineers might make incorrect decisions about adopting new techniques if they don't consider scientific

Tech Transfer model





feature

technology transfer

A Model for Technology Transfer in Practice

Tony Gorschek and Claes Wohlin, *Blekinge Institute of Technology*
Per Garre, *Danaher Motion Särö AB*
Stig Larsson, *ABB Corporate Research*

Successful technology transfer requires close cooperation and collaboration between researchers and practitioners. A seven-step transfer model embodying this philosophy emerged from two academic-industry partnerships.

Technology transfer, and thus industry-relevant research, involves more than merely producing research results and delivering them in publications and technical reports. It demands close cooperation and collaboration between industry and academia throughout the entire research process. During research conducted in a partnership between Blekinge Institute of Technology and two companies, Danaher Motion Särö AB (DHR) and ABB (see the "Industry Partners" sidebar), we devised a technology transfer model that embodies this philosophy. We initiated this partnership to conduct industry-relevant research in requirements engineering and product management. Technology transfer in this context is a prerequisite: it validates academic research results in a real setting, and it provides a way to improve industry development and business processes.

Our model involves seven steps, each building on the ones before it. Although previous transfer models, such as the one by Shari Lawrence Pfleeger,[1] inspired our model, its complete development evolved over time. These steps emerged during a long-term joint commitment, erences when performing validation is important for success.[2]

In this article, we present our technology transfer model, and we report our experiences and lessons learned for each of the seven steps involved. Figure 1 outlines this model, showing how all seven steps are relevant and interdependent for overall transfer success.

Step 1: Identify potential improvement areas based on industry needs

We began by assessing current practices, observing domain and business settings, and

Wrapping up

- Definitions
- Strategic decision support
- Operational decision support
- Making change

Literature

Barbee E. Teasley, Laura Marie Leventhal, Clifford R. Mynatt, Diane S. Rohlman, "Why software testing is sometimes ineffective: Two applied studies of positive test strategy." *Journal of Applied Psychology*, Vol 79(1), Feb 1994, 142-155.

Håkan Petersson, Thomas Thelin, Per Runeson and Claes Wohlin, "Capture-Recapture in Software Inspections after 10 Years Research - Theory, Evaluation and Application", *Journal of Systems and Software*, 72(2):249-264, 2004.

David J. Anderson, Making the Business Case for Agile Management - Simplifying the Complex System of Software Engineering, *Motorola S3 Symposium*, 2004.

Daniel Karlström and Per Runeson, "Combining Agile Methods with Stage-Gate Project Management", *IEEE Software*, May/ June, pp.43-49, 2005

Tore Dybå, Barbara Kitchenham, Magne Jørgensen, Evidence-Based Software Engineering for Practitioners, IEEE Software, January, pp 58-65, 2005

Per Runeson, Carina Andersson, Anneliese Andrews, Tomas Berling and Thomas Thelin, "What Do We Know about Defect Detection Methods?", *IEEE Software*, pp. 82-90, May/June 2006

Per Runeson, "A Survey of Unit Testing Practices", *IEEE Software*, pp. 22-29, July/August 2006.

Tony Gorschek, Per Garre, Stig Larsson, Claes Wohlin, "A Model for Technology Transfer in Practice," *IEEE Software*, pp. 88-95, November/December, 2006

Barbara A. Kitchenham, Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3. Technical Report S.o.C.S.a.M. Software Engineering Group, Keele University and Department of Computer Science

Carina Andersson, "A Replicated Empirical Study of a Selection Method for Software Reliability Growth Models", *Empirical Software Engineering*, 12(2):161-182, April 2007

Carina Andersson and Per Runeson, "A Spiral Process Model for Case Studies on Software Quality Monitoring - Method and Metrics", *Software Process Improvement and Practice*, 12(2):125-140, 2007.

Per Runeson, Per Heed, and Alexander Westrup, *A Factorial Experimental Evaluation of Automated Test Input Generation – Java Platform Testing in Embedded Devices*, PROFES 2011.

Paulo Anselmo da Mota Silveira Neto, Per Runeson, Ivan do Carmo Machado, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira, Emelie Engstrom, "Testing Software Product Lines," *IEEE Software*, pp. 16-20, September/October, 2011