An Empirical Study On Integrating Analytical Quality Assurance Into Pair Programming

Dietmar Winkler, Ramona Varvaroi, Gernot Goluch, Stefan Biffl
Vienna University of Technology,
Institute of Software Technology and Interactive Systems

dietmar.winkler@qse.ifs.tuwien.ac.at
http://qse.ifs.tuwien.ac.at

Motivation

- In traditional pair programming the observer performs some quality assurance activities, e.g., implicit continuous reviews.
- This implicit quality assurance is not well defined, not traceable and not repeatable.

- Thus, traditional pair programming is not suitable for environments that need well-defined, traceable and repeatable quality assurance (e.g., security-related application domains).

- This work focuses on the investigation of the effect of defined quality assurance activities in a pair programming team.

  - How to integrate explicit quality assurance in pair programming?
  - How can we show traceability and repeatability?
  - What are the effects of quality assurance activities on defect detection?

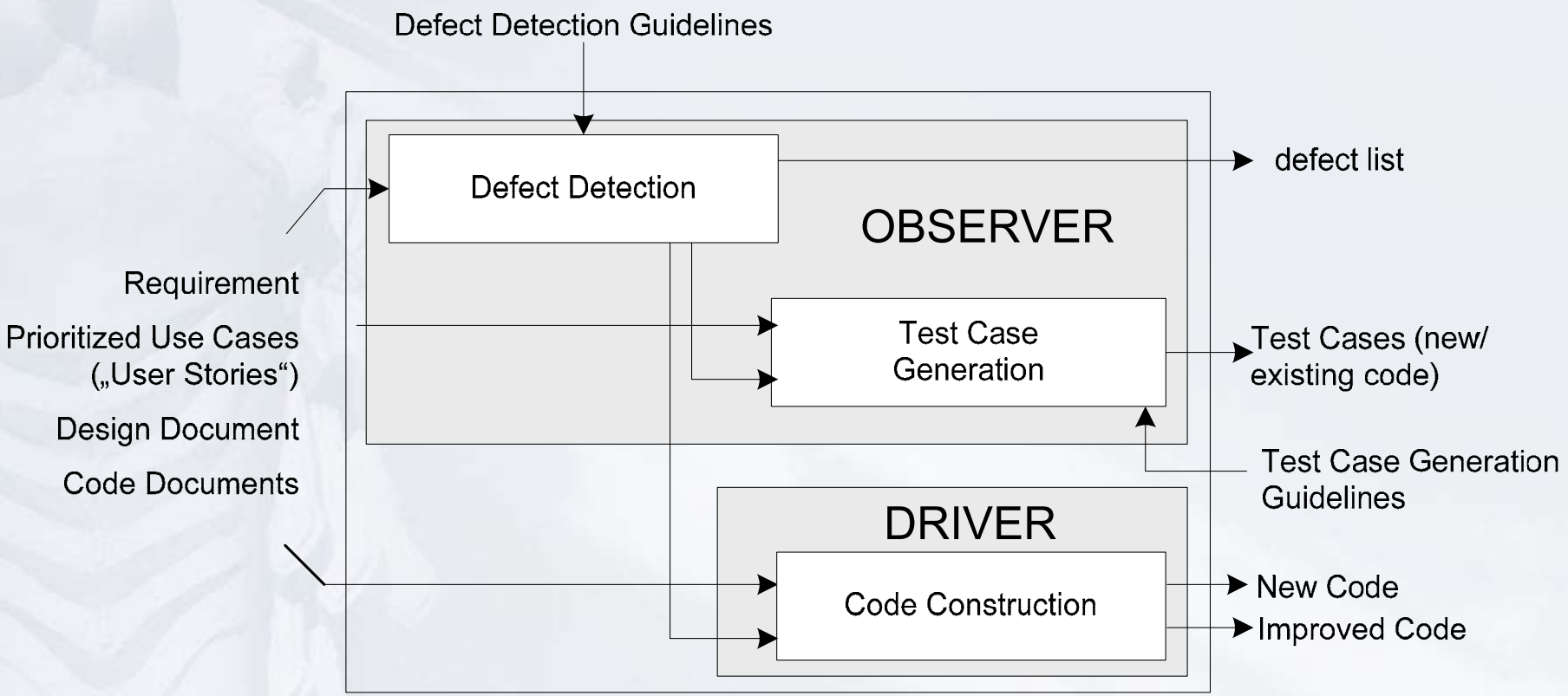
Background

- **Pair Programming**
  - is an flexible and constructive approach for software development in short iterations.
  - supports tight customer interaction and frequent requirements changes.
  - focuses on software construction performed by 2 persons sharing a common working environment.

- **Analytical Quality Assurance (QA) Activities**
  - are sometimes considered as add-on activity in software development (even if time is very short).
  - supports systematic defect detection and product improvement.

- **Well-defined quality assurance activities are:**
  - Best Practice Software Inspection
  - Software Testing (based on requirements)

Integrated Pair Programming (IPP) Approach



Expected Benefits of IPP

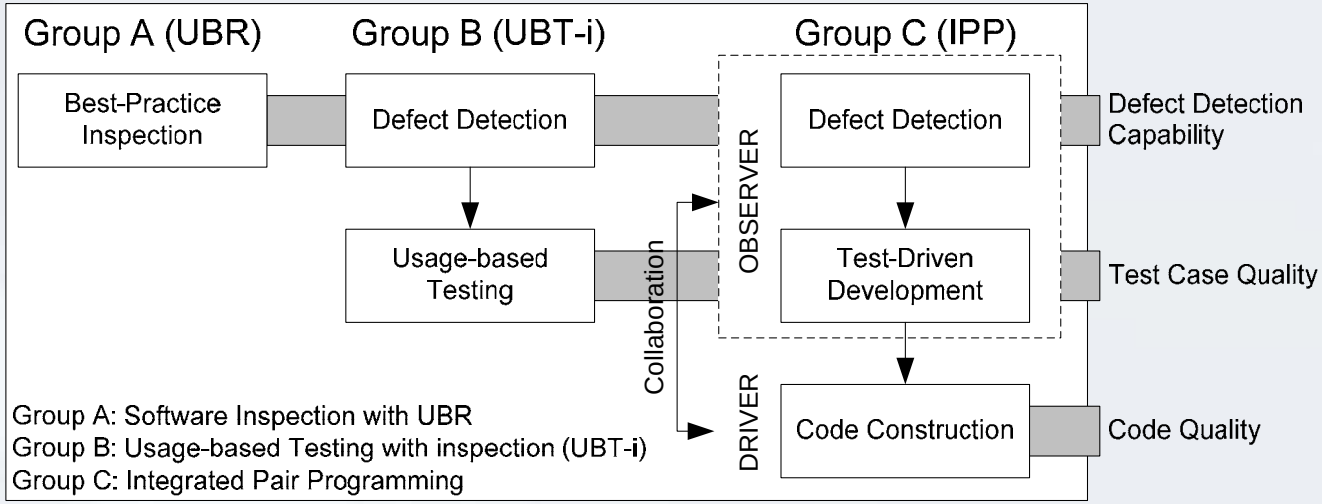
- Flexible (agile) software construction including systematic product quality improvement.
  - Defect Detection (Best-Practice Inspection).
  - Systematic Test Case Generation based on requirements.
- Enhanced Learning effects .
- Systematic and traceable quality activities.
- Enhanced tasks and responsibility for the observer role.
- Application of prioritized use cases according to business value contribution.
- Usage-Based Testing with Inspection enables defect detection AND defect location (comparable to black-box testing based on requirements)

Empirical Study: A Controlled Experiment

Research Objectives:

Investigation & Comparison of

- Defect Detection Capability (Effectiveness, Efficiency)
- Learning Effects
- Test Case / Source Code Quality



Key Features of the Study Design

- Maintenance / evolution process for a commercial application.
- Three Experiment Groups
  (Inspection, Usage-Based Testing with Inspection, Integrated Pair Programming)
- Two session (Central, Taxi)
- Cross check for test case and software code quality by applying several test cases.

Experiment Design

- **Experiment Process** 5 Basic Steps:
  (a) Participant selection, (b) experience collection ((c) experiment preparation for participants, (d) study execution in two sessions including feedback after every session, and (e) data submission.

- **Study Material:**
  Textual requirements, Prioritized Use Cases, Source Code fragments (partially implemented), Guidelines, questionnaires.

- Expert Seeded **Defects:**
  60 defect spread over different document locations (different defect severity classes and types).

- Overall number of 230 **participants** (120 Inspectors, 70 Testers, 40 PP Individuals)

- First results will be available in 10/06.

Thank you!

Dietmar Winkler
Vienna University of Technology,
Institute of Software Technology and Interactive Systems
Favoritenstr. 11/188, A-1040 Vienna, Austria

http://qse.ifs.tuwien.ac.at
dietmar.winkler@qse.ifs.tuwien.ac.at