Improving Evolvability Through Refactoring

Jacek Ratzinger and Michael Fischer

Vienna University of Technology

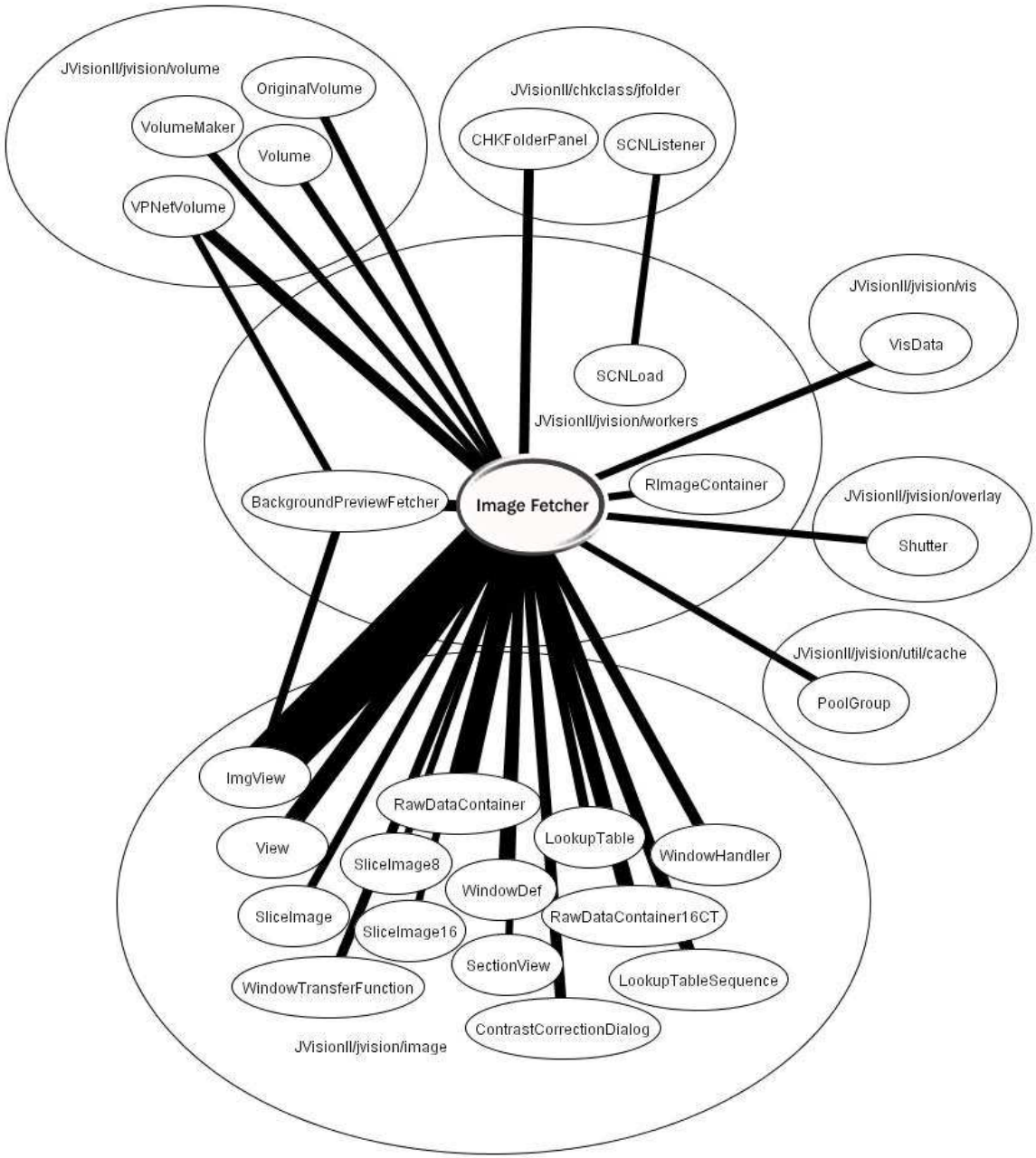
Harald Gall

University of Zurich

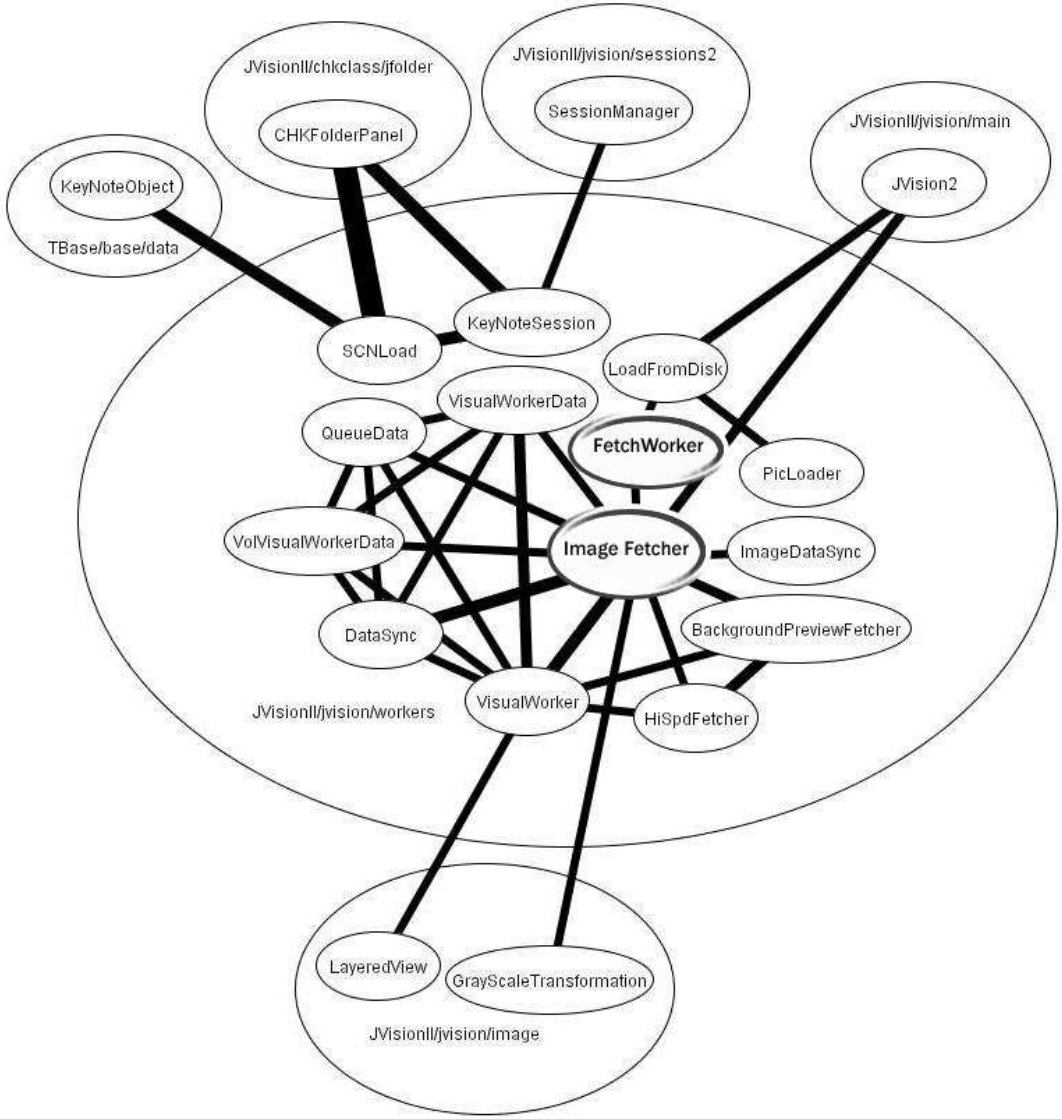
Problem Statement

- Structural and historical information from CVS repository
- Exploitation of change history: change coupling
- Identify *change smells* based on change coupling from RHDB
  - Man-in-the-middle (a central class evolves with many others)
  - Data container (mostly public accessor methods no behavior)
- Structural weaknesses are pointed out and are subject to reengineering activities: remove *bad smells*
- Case study: 500.000 lines industrial Java application
  - 15 months observation
  - Apply refactoring
  - Observe another 15 months to analyze evolution after refactoring

Man-in-the-middle: before ...



... and after refactoring (15 months later)



Results & Future Work

- Industrial case study has shown promising results
- *Change smells* can be used to effectively detect certain cases of *bad smells*
- *Bad change smells* detected:
  - man-in-the-middle
  - data container
- The developer obtains support where to apply refactorings
- Effective monitoring of refactorings
- Provide tool support, e.g., as *Eclipse* plug-in