A Refactoring Tool for Smalltalk

Don Roberts, John Brant, Ralph Johnson
Theory and Practice of Object Systems 1997

Summary by Prof. Thomas LaToza
SWE 795, Spring 2017
Software Engineering Environments
Refactoring: Key Idea

- Concerns may crosscut, making code changes more difficult.

- Important to reorganize code to reduce duplication, organize functionality in appropriate places.

- Refactorings are *behavior preserving* program edits designed to improve design of code (e.g., eliminate redundancy).

- Refactoring tools to assist should:
  - Be completely automated.
  - Provably correct, ensuring no new errors are introduced.
  - Offer more complex refactorings composed from primitives.
Design goals for refactoring

• Integrated into standard development tools
  • Want to integrate so developers cannot help but use

• Be fast: immediately see results of change
  • Refactorings that are slower will not be used

• Avoid purely automatic reorganization
  • Get input from users (e.g., name for new class)

• Be *reasonably* correct
  • Developers must trust them
  • But features like reflection makes it impossible to be completely correct
Refactorings supported

**Instance/Class Variable Refactorings**
- add variable
- rename variable
- remove variable
- push down variable into subclass(es)
- pull up variable from subclass(es)
- create accessors for a variable
- change all variable refs to accessor calls
  (abstract variable)

**Class Refactorings**
- create new class
- rename class
- remove class

**Method Refactorings**
- add method
- rename method
- remove method
- push down method into subclass(es)
- pull up method from subclass(es)
- add parameter to method
- move method across object boundary
- extract code as method
Example: Extract Method

Figure 2 - Screenshot of Refactoring Browser during extract code as method refactoring
Questions for discussion

• Overall reaction to the paper

• What are the barriers to using refactorings today?

• How much trust in the correctness of a refactoring is enough?
  • How much would a developer have to know to even reason about when to trust the tool?

• What additional refactorings might be valuable?