Refactoring R Programs

Tobias Verbeke

Business & Decision

2008-08-12

Plan of the Presentation

Introduction

Current Results

Future Developments

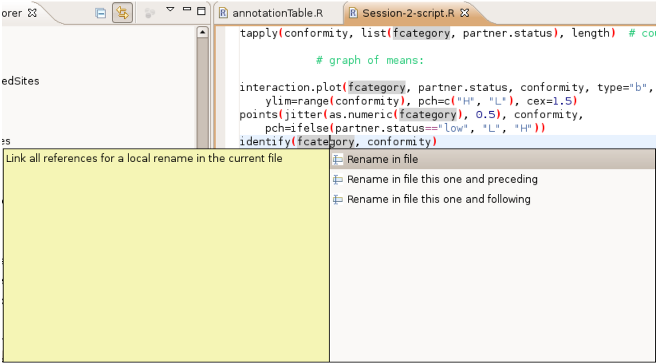
Definition of refactoring

> *Refactoring is the object-oriented variant of restructuring: "the process of changing a [object-oriented] software system in such a way that it does not alter the external behaviour of the code, yet improves its internal structure."*

Opdyke (1992), cited by Mens and Tourwé (2004)

Simple R example in Eclipse

Renaming a variable


orer
edSites
es
annotationTable.R
Session-2-script.R
Rename in file
Rename in file this one and preceding
Rename in file this one and following

```
tapply(conformity, list(fcategory, partner.status), length)  # co

            # graph of means:

interaction.plot(fcategory, partner.status, conformity, type="b",
    ylim=range(conformity), pch=c("H", "L"), cex=1.5)
points(jitter(as.numeric(fcategory), 0.5), conformity,
    pch=ifelse(partner.status=="low", "L", "H"))
identify(fcategory, conformity)
```

Link all references for a local rename in the current file

Some distinctions

- primitive refactorings vs. composite refactorings
- floss refactoring vs. root canal refactoring (Black)
- manual refactoring
  - time consuming
  - error prone

  vs. tool-based refactoring (in a so-called *refactoring* browser)
  - immediate
  - error-free
  - reduction of testing time

Why should you be interested?

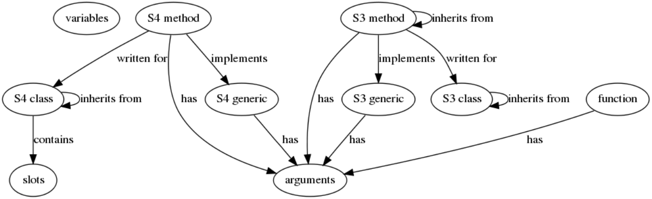
- refactoring is part of test-driven development and eXtreme programming methodologies (test – code – refactor cycle)
- refactoring as part of software reengineering (fate of a consultant. . .)
- with appropriate tool support it can make your programming life even more pleasant.

Refactoring Activities (Mens and Tourwé, 2004)

1. Identify where the software should be refactored.
2. Determine which refactoring(s) should be applied to the identified places.
3. Guarantee that the applied refactoring preserve behaviour.
4. Apply the refactoring.
5. Assess the effect of the refactoring on quality characteristics of the software (e.g., complexity, understandability, maintainability) or the process (e.g., productivity, cost, effort).
6. Maintain the consistency between the refactored program code and other software artifacts (such as documentation, design documents, requirements specifications, tests, etc.).

Meta-model of R

Graph representation of all R-related objects which may be subject to refactoring operations.



Simple tool to keep view of the pre- and post-conditions of a certain refactoring.

Refactoring catalogue

- similar in spirit to the Java and Haskell refactoring catalogues
- adapted to the pecularities of the R language
- template structure for documenting each refactoring
  - name
  - summary
  - R code examples
  - motivation
  - pre-conditions
  - mechanics
- see `http://www.r-developer.org/wiki/refactoring/RefactoringCatalogue`

Detailed example

- name : MergeArguments
- summary : merge two or more arguments of a function into a list object
- motivation : prevent huge argument sequences ; strategy comparable to the `gp` list of graphical parameters in calls to `grid` functions, `control` list in some fitting algorithms (e.g. `nls`)
- preconditions : beware of the dots argument
- mechanics :
  - select the arguments and identify the locations where these are used in statements
  - replace the given arguments by `argList` list
  - replace `arg1` by `argList$arg1` etc.

Upcoming

- complete the Refactoring Catalogue
- Roxygen support (positive side-effect)
- Eclipse Refactoring browser
- work on test artifacts
- software metrics (assess refactoring quality improvements)

Contact Details & Acknowledgements

- `http://www.r-developer.org`
- `tobias.verbeke@gmail.com`
- Sincere thanks to :
  - Stephan Wahlbrink (`http://www.walware.de/goto/statet`)
  - Tom Mens (Université Mons-Hainaut)
  - Johnson & Johnson PRDBE