Automated refactoring

Tom Rochette <tom.rochette@coreteks.org>

August 30, 2025 — 861fb9d0

0.1 Context

0.2 Learned in this study

0.3 Things to explore

- Abstraction level assignment ((semi-)automatically assign an abstract level to a function such that abstraction levels are respected, use similar abstraction levels in a function)
- Is there any way to think about refactoring in a way that it is applied optimally?
  - Is a pass system the most optimal way to do refactoring? It would potentially require the CFG/DFG to be recomputed every time a function is refactored

1 Overview

One of the goal of code refactoring is to reduce the amount of duplicated code such that it is easier to make changes in a single place rather than make the same change at multiple places.

Another goal of refactoring is to promote code reuse and abstraction.

As a software project matures, its low level functions should require to be changed rarely while higher level features relying on those lower level functions change more often. This would be similar to assembly language changing very rarely compared to higher level languages.

2 What do programmers do to improve code?

(tentatively sorted from easiest to automate to hardest)

- Detect and fix defects
  - Invalid syntax
  - Incorrect logic
  - Use of proper types
- Remove dead code
- Define and follow code style standard
- Use more appropriate data structures for the given use cases
- Reduce code complexity
  - Law of Demeter
  - Limit on functions/methods line count
  - Cyclomatic complexity
- Create test cases to ensure code stability during changes
- Refactor improper architecture
  - Properly define classes responsibilities
  - Properly define classes collaborators
  - Reduce coupling
- Write code in terms of pre/post-conditions and return as soon as possible

2.1 Standard operations

- Remove deprecated code
- Update code relying on deprecated/obsolete APIs
- Simplify logic
- Extract string to resource file

2.2 Classical refactoring methods

- Extract method
- Pull up method
- Form template method
- Substitute algorithm

2.3 Regular refactoring

- Pyramid logic to precondition testing
- Extraction of repetitive conditions
- Restructuring of a function into multiple functions (convert a 100 lines function into 20x 5 liners)
  - The difficulty here is to give significant name to the newly created functions
- Abstract existing code through parameter instantiation (pass a parameter instead of using a hard-coded value)
- Condition rewriting: Inverse the existing condition
- Extraction of valid points of entry
  - All files are executed to test if they compute anything (executable code or only function declaration/library?)
  - Files that may execute code may not have the necessary deep dependencies available, which may make it possible to determine they are not a valid point of entry
  - Dependency extraction may make it possible to determine the most likely call points
- Computation/algorithm/implementation extraction
- Notify programmer when modifying code that has clones
- Check variable safety (against injection)
- Notify programmer of operations that need to be done due to a change initiated by him (e.g., function renaming, new function parameter, different return type, etc.)
- Abstraction level assignment

2.4 High level

- Code restructuring (namespacing/directory structure/file location)

2.5 Rule-based refactoring

- Law of Demeter
- Maximum functions/methods line count
- Maximum cyclomatic complexity
- Maximum parameter count of functions/methods
- Creation of classes for complex return types (prevent returning tuples or arrays)

2.6 Extract method

- Extract loop content (processing on a single item)
- Extract conditions block (logic specific to a state)

3 See also

- Clone detection

4 References

- https://www.jetbrains.com/resharper/features/code_refactoring.html
- https://en.wikipedia.org/wiki/Code_refactoring
- http://autorefactor.org/
- http://refactoring.com/
- Zanoni, Francesco Duplicated Code Refactoring Advisor (DCRA): a tool aimed at suggesting the best refactoring techniques of Java code clones