Lecture 5

Topics on Refactoring

Some materials are based on Martin Fowler's book

Last lecture and tutorial …

Design patterns

- We showed the structures of all the classic design patterns
- We explained some of them and their applications in OpenOME, Protégé and Eclipse
- For your exercise
  - Explain the MVC and plug-in patterns using the classic design patterns
  - Find more cases in OpenOME that can apply the design patterns
- The application of design patterns, can be called "refactoring"

Today …

Topics on Refactorings

1. What is refactoring? Why?
2. How to classify refactorings
3. How to apply refactorings
4. Compare with tuning and design patterns
5. Refactor source code into requirements
6. Summary

References

Martin Fowler. *Refactoring – improve the design of existing code*. http://www.refactoring.com

Tom Mens et al. "A survey of software refactoring". TSE 30(2), 2004.

1. What is refactoring?

- It is a new English word, can be used in part of speech for a noun (countable or uncountable), a verb …
- Its origin = Factoring

Factoring

- In *mathematics*, **factorization** or **factoring** is the decomposition of an object into an expression of smaller objects, or **factors**, which multiplied together give the original
- For example, the number 15 factors into *primes* as 3 × 5; and the *polynomial* x^2 - 4 factors as (x - 2)(x + 2)

Refactoring

- **Refactoring** is the process of rewriting written material to improve its *readability* or structure, with the explicit purpose of keeping its meaning or behavior.
  - The term is by analogy with the *factorization* of numbers and polynomials. For example, x^2 - 1 can be factored as (x + 1)(x - 1), revealing an internal structure that was previously not visible (such as the two zeroes at +1 and -1). Similarly, in software refactoring, the change in visible structure can often reveal the "hidden" internal structure of the original code.
- Extracting common descriptions
  20 + 20 + 20 = (1 + 1 + 1) x 20 = 3 x 20

Software Refactoring

- Software refactoring = "*Restructuring* existing code by altering its internal structure without changing its external behavior"
  – adapted from Martin Fowler's book
- To avoid duplications
  A. Hunt, and D. Thomas. *Pragmatic Programmer*, Addison Wesley,1999.
  Martin Fowler, *Avoid Repetition*, IEEE Software, Jan/Feb 2001 pp.97—99.

More on definitions

Are the following activities refactorings?

- Adding new functionalities
- Fixing correctness bugs
- Tuning performance
- Patching security holes

When to apply refactorings

"Any fool can write code that a computer can understand. Good programmers write code that human can understand"

Bad code smells:

- Duplicate code (clones): *feature envy*
- Complex control, Long method
  use *Hammock graph*: single entry/single exit
  - Comments signal semantic distance
  - Conditional and loops
- Complex data, Long parameter list
- OO specific: large class, switch statements, parallel inheritance, middle man, message change, temporary fields, data class, etc.

The refactoring rhythms

- Development = (Adding features, Refactoring)*
- Refactoring = (Testing, Small Steps) *
- Small Steps = one of the refactoring types

2. Type of refactorings

"Putting things together when changes are together"

- Extract Methods
- Move Methods
- Rename Methods
- Replace Temp with Query
- Replace conditionals with polymorphism
- Replace Type code with State/Strategy
- Self Encapsulate Field
- ……

3. Applications

- We use three examples to explain some basic refactorings
  - Extract method:
    - signalled by comments
    - single-entry, single-exit
    - increase the level of indirection
    - reduce the length of a method
    - increase the chance of reuse
  - Move method:
    - Place method together with the object, *Putting things together when changes are together*
  - Replace conditions with polymorphism
    - Switches are "hard code", polymorphism is better for extensibility in OO

Example 1 – Extract method

```
void f() {
    ...
    // Compute score
    score = a * b + c;
    score -= discount;
}
```

```
void f() {
    ...
    computeScore();
}

void computeScore() {
    score = a * b + c;
    score -= discount;
}
```


Example 2 – Move method

```
class Jar {
   ...
}
class RoboPacker {
   private bool isFragile(Jar foo) {
      switch(foo.material) {
      case GLASS: return true;
      case WOOD: return true;
      case TIN: return false;
      }
   }
}
```

```
class Jar {
   bool isFragile() {
      switch(material) {
      case GLASS: return true;
      case WOOD: return true;
      case TIN: return false;
}}}
class RoboPacker {
   private bool isFragile(Jar foo) {
      return foo.isFragile();
   }
}
```


Example 3 – Replace conditionals with polymorphism

```
class Jar {
   bool isFragile() {
      switch(material) {
      case GLASS:
         // complex glass calculation
      case WOOD:
         // complex wood calculation
      case TIN:
         // complex tin calculation
}}}
```

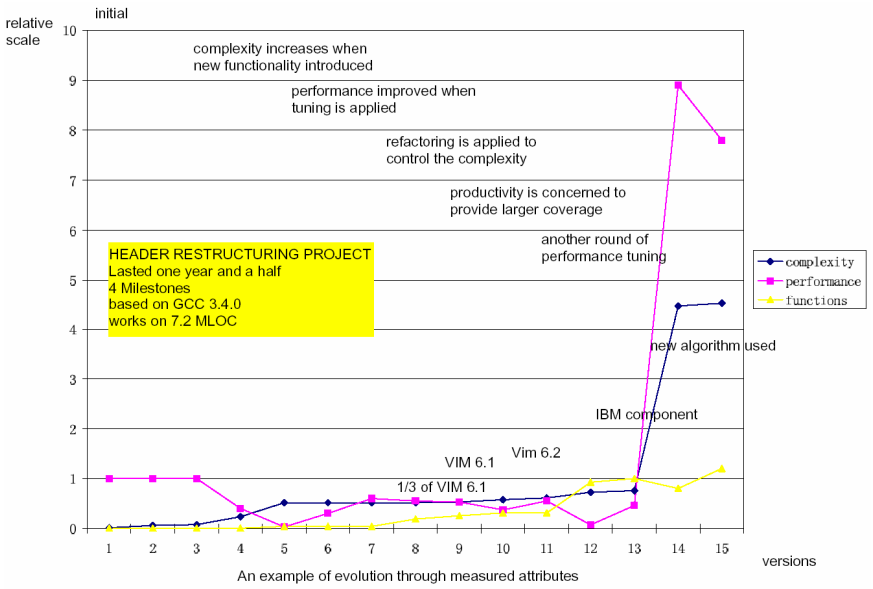
```
class Jar {
   bool isFragile() {
      return material.isFragile();
}}

interface Material { ... }
class GlassMaterial:Material { ... }
class WoodMaterial:Material { ... }
class TinMaterial:Material { ... }
```


4. Refactoring versus Tuning

- Refactoring aims at improve understandability and maintainability
- Tuning aims at improve performance
- They are both non-functional (no new features), but they are different
  - Refactoring can be harmful to performance
  - Tuning can be harmful to maintainability
- You need to know where are the bottlenecks
- Y. Yu et al. “Software refactorings guided by softgoals”, *REFACE workshop in conjunction with WCRE’03*.

The header restructuring project



Your exercise

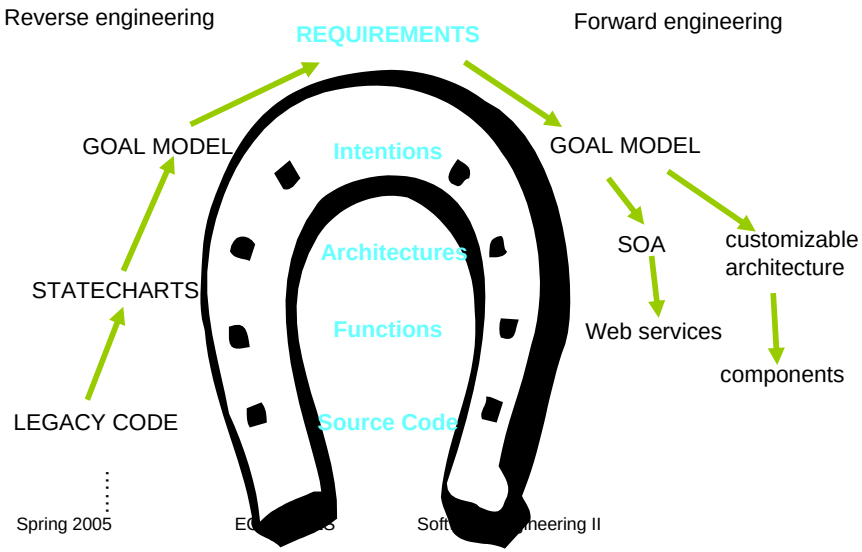
- Monitor the evolution of your software product by measured its metrics
  - Statically:
    complexity metrics: LOC, Halstead, McCabe
  - Dynamically:
    Performance metrics: time (clockticks, #instructions), space (cache misses, L1 instruction, L1 data, L2 cache, etc., memory footprint)
- Decide on which is the urgent non-functional task

5. Refactoring into Requirements

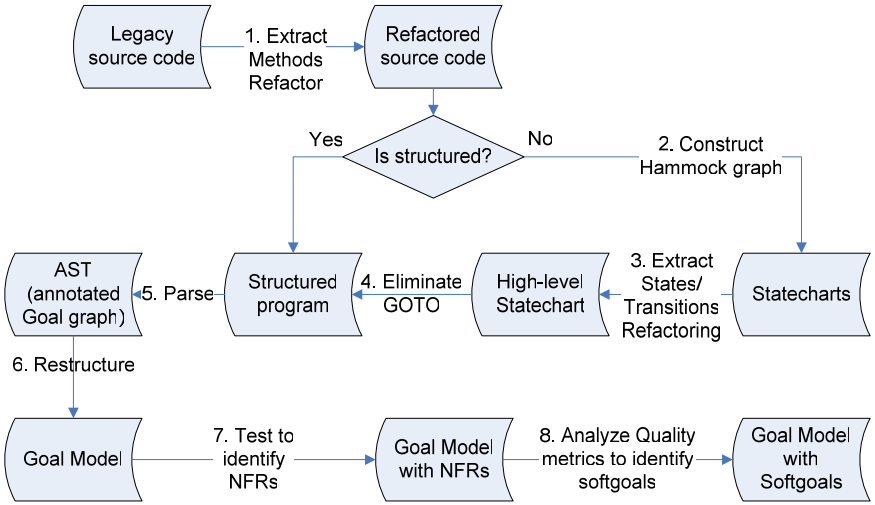
Motivation to recover requirements from source code

- Requirements are lost in documentations, sadly, it is very common in the software development practices
- Legacy software code are not explained in documentation
- Mismatch between implementations and requirements

Huge gap in abstractions



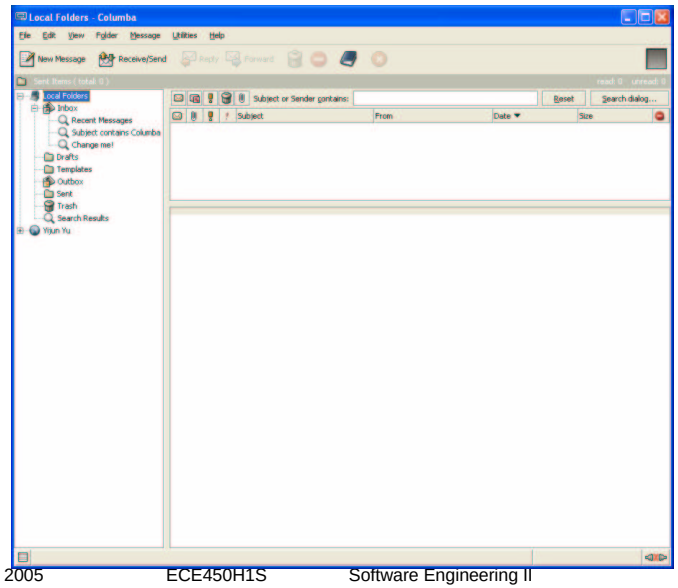
A semi-automatic process



Example. Columba Refactoring

- Search “Java email client” in Google, you will find this software
- It is open-source
- It has 140 KLOC in Java
- It also has plug-in patterns
- First thing, we modify the code base to fit Eclipse development (moving packages, i.e., move all “src” subdirectories including plug-in projects under the same “src” directory)

A screenshot



Where to look at first?

- Secondly, we look for the main routine from the manifest in the JAR file

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.6.2
Created-By: 1.4.2_06-b03 (Sun Microsystems Inc.)
Main-Class: org.columba.core.main.Main
Sealed: false
Class-Path:  lib/usermanual.jar lib/junit.jar lib/lucene-1.3-final.jar
  lib/commons-cli-1.0.jar lib/jwizz-0.1.2.jar lib/plastic-1.2.0.jar li
 b/jhall.jar lib/forms-1.0.4.jar lib/ristretto-1.0_RC2.jar lib/jscf-0.
 2.jar lib/macchiato-1.0pre1.jar lib/frapuccino-1.0pre1.jar lib/winpac
 k.jar lib/jniwrap-2.4.jar lib/jdom.jar lib/jpim.jar lib/je.jar ${lib.
 jdic}
```

The Main routine

```
public static void main(String[] args) {
    Main.getInstance().run(args);
}
```

Thus we look at “run” routine, which has 81 lines of code

The Run routine

```
public void run(String args[]) {
1    ColumbaLogger.createDefaultHandler();
2    registerCommandLineArguments();
3    // handle commandline parameters
4    if (handleCoreCommandLineParameters(args)) {
5       System.exit(0);
6    }
7    // prompt user for profile
8    Profile profile = ProfileManager.getInstance().getProfile(path);
9    // initialize configuration with selected profile
10     new Config(profile.getLocation());
11     // if user doesn't overwrite logger settings with commandline arguments
12     // just initialize default logging
13
14     ColumbaLogger.createDefaultHandler();
15     ColumbaLogger.createDefaultFileHandler();
16
17     for ( int i=0; i<args.length; i++) {
18        LOG.info("arg["+i+"]="+args[i]);
19     }
20     …
```

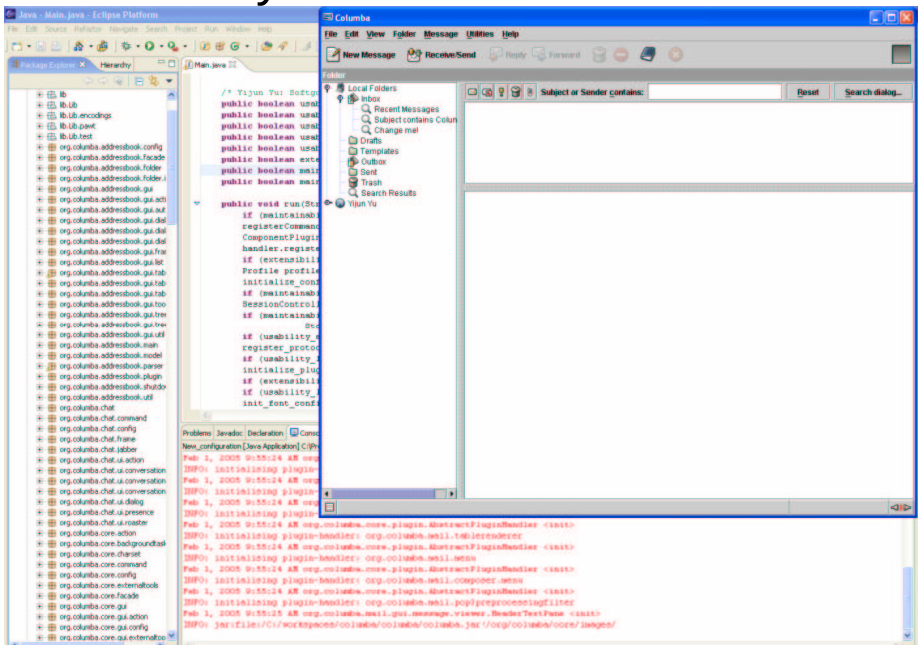

The Run routine refactored

```
public void run(String args[]) {
      ColumbaLogger.createDefaultHandler();
       registerCommandLineArguments();
       ComponentPluginHandler handler = register_plugins();
       handler.registerCommandLineArguments();
       handle_commandline_parameters(args);
       Profile profile = prompt_user_for_profile();
       initialize_configuration_with_selected_profile(profile);
       initialize_default_logging(args);
       SessionController.passToRunningSessionAndExit(args);
       enable_debugging_repaint_manager_for_swing_gui_access();
       StartUpFrame frame = show_splash_screen();
       register_protocol_handler();
       load_user_customized_language_pack();
       initialize_plugins(handler);
       load_plugins();
       set_look_and_feel();
       init_font_configurations();
       set_application_wide_font();
       hide_splash_screen(frame);
       handle_commandline_arguments_of_the_modules(handler);
       restore_frames_of_last_session();
       ensure_native_libraries_initialized();
       post_startup_of_the_modules(handler);
  }
```


Identify NFR and introducing softgoals

```
public boolean usability = false;
public boolean usability_language_customization = false;
public boolean usability_assured_progress = false;
public boolean usability_look_and_feel = false;
public boolean usability_font_configuration = false;
public boolean extensibility = false;
public boolean maintainability_debugging = false;
public boolean maintainability_logging = false;
public void run(String args[]) {
   if (maintainability_logging) ColumbaLogger.createDefaultHandler();
   registerCommandLineArguments();
   ComponentPluginHandler handler = register_plugins();
    handler.registerCommandLineArguments();
   if (extensibility) handle_commandline_parameters(args);
   Profile profile = prompt_user_for_profile();
   initialize_configuration_with_selected_profile(profile);
   if (maintainability_logging) initialize_default_logging(args);
   SessionController.passToRunningSessionAndExit(args);
   if (maintainability_debugging) enable_debugging_repaint_manager_for_swing_gui_access();
         StartUpFrame frame = null;
   if (usability_assured_progress) { frame = show_splash_screen(); }
   register_protocol_handler();
   if (usability_language_customization) load_user_customized_language_pack();
   initialize_plugins(handler);
   if (extensibility) load_plugins();
   if (usability_look_and_feel) set_look_and_feel();
   init_font_configurations();
   if (usability_font_configuration) set_application_wide_font();
   if (usability_assured_progress) hide_splash_screen(frame);
   if (extensibility) handle_commandline_arguments_of_the_modules(handler);
   restore_frames_of_last_session();
   if (extensibility) ensure_native_libraries_initialized();
   if (extensibility) post_startup_of_the_modules(handler);
}
```

The system without the NFRs



6. Summary

- The concepts of refactoring
- The relation to restructuring, reengineering, design patterns, performance tuning, and requirements are explained
- Refactoring is not limited to OO software, that's the major different from the design patterns
- Refactoring is not aiming at all quality attributes, they are mainly for maintenance
- Refactoring is used to reveals new structures, thus it can be used to increase the level of abstraction gradually, leading to even requirements
- A lot research is coming …

Further readings

- Martin Fowler. *Refactoring – improve the design of existing code*.
- Martin Fowler, *Avoid Repetition*, IEEE Software, Jan/Feb 2001 pp.97—99.
- Tom Mens et al. "A survey of software refactoring". TSE 30(2), 2004.
- Y. Yu et al. "Software refactorings guided by softgoals", *REFACE workshop in conjunction with WCRE'03*.
- Y. Yu et al. "Refactor source code into goal models", Technical report.

What's next …

- A Tutorial on more refactoring practices
  - How to use refactoring in Eclipse?
  - How to use statecharts to represent the refactorings for unstructured code (Web-based software) For example, Squirrel Mail.