Modularization in Large-Scale OO System

Jimin Hwa

13th January 2010
Contents

- Introduction

- Issues on Modularization
  - Quality assessment of modular design
  - Modularization techniques

- Research in Progress

- Summary
Introduction
Introduction

- Complexity of software system’s design should be managed during its lifetime

<Developer/maintainer>  Complexity

Costly & faulty

Understanding  Modification

<Software system’s design>
Introduction

- Complexity management is more significant in large-scale software

  the granularity of a class at much too low a level to serve as a unit for software modularization*

* Sarkarn et al., “Metrics for measuring the quality of modularization of large-scale object-oriented software,” *TSE*’08
Modularization

- Modularization provides high-level view of a system by decomposing it into meaningful and manageable subsystems.
**Flat vs. Hierarchical Modularization**

Examples of *Flat & Hierarchical* Modularization

* Flat modular design

* Hierarchical modular design


copyright © 2008 All rights reserved by Jimin Hwa
Issues on Modularization
Issues on Modularization

- Quality assessment of a modular design
- Modularization techniques
- Etc
  - Labeling
  - Visualization
Quality Assessment of Modularization

- To manage the quality of the modular design of a system
- To judge whether improvement of a modular design is needed

![Diagram showing quality assessment over a series of a system with a trend line indicating restructuring is needed at a certain point.](image-url)
Quality Assessment of Modularization

Related Work: Sarkarn et al.’s*

- Provide a set of metrics that characterizes large OO software with respect to the APIs of the modules of a system

* Sarkarn et al., “Metrics for measuring the quality of modularization of large-scale object-oriented software,” TSE’08
Related Work: Sarkarn et al.’s

- Validate the metrics by applying them to open-source software systems
  1. Justify the values on the basis of manual examination of the software systems
  2. Detune the systems and examine the changes of metric values

* Sarkarn et al., “Metrics for measuring the quality of modularization of large-scale object-oriented software,” *TSE’08*
Related Work: *Ours*

- Provide a metric-based model to assess the (hierarchical) modular design of a system in the view of understandability.

* J. Hwa et al., “Hierarchical Understandability Assessment Model for Large-Scale OO System,” *APSEC’09*
Related Work: *Ours*

- Validate the model by applying them to series of a well-managed open-source software system.

![Graph showing measured values over a series of system versions](image)

- [Version shift 0.4 → 0.5]
- **Match?**
- Real change = (+)
- Expected change by manual analysis = (+/-)
Modularization Techniques

- Provide techniques to (re-)build/structure a modular design of a system (semi-)automatically

- Contain criteria of quality modules and a method to build such modules

});

- Homogeneous
  - Structural properties of design/code (Ex: cohesion)
  - Terms in design/code
  - Features in design spec.
  - Features in history

- Heterogeneous
  - Software metrics
  - Concept analysis
  - Data mining

© 2008 All rights reserved by Jimin Hwa
Related Work: B. S. Mitchell et al.’s*

- Propose an approach to (re)structure a flat modular structure of a system
  - Input: Class-level design/source code
  - Criteria: Coupling & cohesion
  - Method: Search-based technique (Hill-climbing, GA)

```
M1 M2
CF(M) = Ratio b/w internal and external edges of a module M
MQ(S) = Aggregation of CF for all modules in system S
```

**Modularization Techniques**

**Related Work: O. Maqbool et al.’s**

- Review several existing hierarchical clustering approaches
  - Input: Structural design/source code written in a procedural language
  - Criteria: Homo/heterogeneity in the use of features of the design/code
    - Features: Functions, global variables and user defined types
  - Method: Hierarchical clustering algorithms

<table>
<thead>
<tr>
<th>Function</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 1 0 1 0</td>
</tr>
<tr>
<td>B</td>
<td>0 1 0 1 1</td>
</tr>
<tr>
<td>C</td>
<td>1 0 1 0 1</td>
</tr>
</tbody>
</table>

*O. Maqbool., “Hierarchical Clustering for Software Architecture Recovery,” TSE, 07*
Modularization Techniques

Related Work: *R. Lutz’s*

- Propose an approach for hierarchical modularization of a system
  - Input: Class-level design/source code
  - Criteria: Min. description length of a modular design (Information-theory)
  - Method: Genetic algorithm (GA)

* R. Lutz, “Evolving good hierarchical decompositions of complex systems,” *TSE’06*
Related Work: R. Lutz’s

- Propose an approach for hierarchical modularization of a system
  - Input: Structural design/Source code
  - Criteria: Min. description length of a modular design (Information-theory)
  - Method: Genetic algorithm (GA)

Compressor

The degree of compression depends on
- #entities/modules/edges in code
- Frequencies of modules’ and entities’ names in code
- Path length in the hierarchy between interacting entities

#Bits/Message length to describe the code

Complexity of the modular design
Evaluating Modularization Techniques

- **External validation**
  - By manually analyzing modular designs created by an approach
    - Soliciting feedback on the clustering results from one or more of the system developers
  - By comparing the modular designs by experts

- **Internal validation**
  - Without benchmark
  - By examining properties which the design by good techniques would have
    - Ex1> Stability: Small changes does not cause dramatic change of the design
    - Ex2> Shapes of hierarchies produced by hierarchical clustering algorithms

---

Copyright © 2008 All rights reserved by Jimin Hwa
Research in Progress
Motivation

- No evidence that the real modularization by designers show the structural properties of the existing techniques
  - The modules organized by designers may/may not show the structural properties defined by the techniques
  - It could be necessary to merge several techniques to cover a designer’s modularization

Modularize with his intention
Related Work: M. G. Bocco et al.’s

- Conduct an experimental analysis of the impact of complexity on the maintenance of the UML class diagrams

Overview of Research

- Assumptions
  - 1\textsuperscript{st} : Well-managed modules have structural properties
  - 2\textsuperscript{nd} : Modules have different structural properties each other → Target to verify
    - It is not that all modules can be defined by same structural properties
    - Modules can be classified by their structural properties

- Experiment plan
  - Modular design #1
  - Measure
  - < Structural properties of modules >
  - Ex> #dependency, cohesion, etc...
  - Analysis

copyright © 2008 All rights reserved by Jimin Hwa
Considerations for Experiment Design

- **How justify that a given system’s modular design is well-managed**
  - Use open-source projects which managed for a long period or known as well-managed systems in literatures
  - Analyze histories and interview the designers to grasp current state of the systems

- **What structural properties are better to be used**
  - Simple and intuitive properties
    - Ex> #dependencies on other modules
  - Complex properties provided by the existing modularization techniques
Summary

- Modularization can be helpful in reducing the complexity of software
  - By decomposing a system into meaningful and manageable subsystems

- Approaches for assessing and automatic-structuring a modular design of a system have been studied
  - Many literatures have used structural characteristics of the design to assess and structure it

- It is necessary to study the relationship between modules designed by experts and structural properties
  - An experiment design for this study is in progress