Identifying and addressing problems in object-oriented framework reuse

Douglas Kirk, Marc Roper and Murray Wood

Empirical Software Engineering (June 2007)

Hyoung-In Ihm
Contents

- Introduction
- Overview
- Related work
- Study 1: identifying the problems
- Study 2: addressing the problems
- Conclusion
- Discussion
Introduction (1/2)

- The object-oriented **framework**
  - Define overall software architecture
    - Classes, objects, collaborations, thread of control, etc
  - Provide large scale software reuse

- The framework documentation
  - Help developers to understand a framework

- Experience of OO framework reuse
  - Appear that many obstacles to reuse
    - One of the major obstacles is **the potential amount of understandings**
Introduction (2/2)

- Past framework documentations
  - Lack in what aspect of framework should be addressed
    - Researchers have no common sense

- Research in this paper
  - Try to identify key aspect of understanding of framework
  - Try to address identified key aspect with documentation
Overview

Problems in OO framework reuse

Lack in past documentation techniques

Find out key aspect of framework

Try to address four problems

Hard to understand reusing framework

What is effective concerning of understanding

Found four problem categories

Try to cover four problems with document techniques

Introduction (1/2)
- The object-oriented framework
  - Define overall software architecture
  - Classes, objects, collaborations, control of control, etc.
  - Provide large-scale software reuse
- The framework documentation
  - Help developers understand a framework
- Experience of OO framework reuse

Related work (1/4)
- Source code (Lahtis and Keller 1994)
  - Approach
    - Tight link between documentation and source code
    - Travels between source code
    - Inferences
    - Edges
    - Nodes
    - Main concerns
    - Understanding navigable structure of the framework

Study 1: identifying the problems (1/4)
- Three scenarios of framework re-user
  - Induced developer
  - A class of software architecture students
  - A group of project students
- Framework for reuse task
  - BidDraw framework
  - Object-oriented and Layer-Aware OOC framework

Study 2: addressing the problems (1/5)
- Two adapted document techniques for BidDraw
  - Patten language
    - To address reuse and architectural problems
  - Micro-architecture description
    - To address concreteness and abstraction problems
- Seven individual framework re-user
  - Two undergraduates
    - Improve experience and developers with BidDraw
  - Three postgraduate students
    - Improve wider spectrum of experience
  - Two software architecture lecturers
  - Improved framework domainFinally developed
Related work (1/4)

- **Source code (Lajoie and Keller 1994 **COODBSE**)**
  - **Approach**
    - Tight link between document and source code
    - Tool to manage source code
      - Together
      - Eclipse
      - Javadoc
  - **Main concern**
    - Understanding static structure of the framework
  - **Lack**
    - Understanding dynamic behavior
Related work (2/4)

- Contracts (Helm et al. 1990 ECOOP)
  - Approach
    - Describe the communication protocol between classes
    - Decompose framework interactions into units (micro-architectures)
  - Main concern
    - Describing key interactions
  - Lack
    - Formal guidance
Related work (3/4)

- Examples (Johnson 1992 OOPSLA)
  - Approach
    - Present examples to represent framework capabilities
    - Featuring fragments of code with explanation
  - Main concern
    - Use examples to understand the framework
  - Lack
    - Problems finding functionality beyond the examples
Related work (4/4)

- Pattern language (Johnson 1992 OOPSLA)
  - Approach
    - Decompose the design problem into sub-problem which is pattern
    - Pattern identifies constrains and proposes a solution
  - Main concern
    - Understand purpose, how to use its part and its design
  - Lack
    - Detailed interactions

- Summary
  - Single documentation can not address all aspects of a framework
  - Lack of agreement on the types of problems
Study 1 : identifying the problems (1/4)

- Three scenarios of framework re-user
  - Individual developer
  - A class of software architecture students
  - A group of project students

- Framework for reuse task
  - JHotDraw framework
    - Mature, well designed, and well documented OO framework
    - Widely used as a case study in research literature

- Data collecting
  - Main concern
    - Encountered problems, what think/search/solution
### Study 1: identifying the problems (2/4)

#### Experiment table

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Individual</th>
<th>Software architecture class</th>
<th>Project students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>1</td>
<td>77</td>
<td>4</td>
</tr>
<tr>
<td>Task</td>
<td>Create orrery application</td>
<td>Five divided exercises from individual scenario, and a modification with own idea</td>
<td>Golf hole designer, UML class diagram editor, Railway track editor, etc</td>
</tr>
<tr>
<td>Period</td>
<td>80 hours</td>
<td>7 weeks</td>
<td>6 months</td>
</tr>
<tr>
<td>Data Collect</td>
<td>Log of thinking and doing. Record every few minutes.</td>
<td>Newsgroup as forum for lecturer and students</td>
<td>Coursework report requires students to describe problems and solutions</td>
</tr>
<tr>
<td>Data Amount</td>
<td>28 papers</td>
<td>216 postings 770 papers</td>
<td>33 papers</td>
</tr>
</tbody>
</table>
Study 1: identifying the problems (3/4)

- **Threat to Validity**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internal</strong></td>
<td></td>
</tr>
<tr>
<td>Time between problems occurring and being recorded</td>
<td>-</td>
</tr>
<tr>
<td>Problem identification relies on the experience of the analyst</td>
<td>-</td>
</tr>
<tr>
<td>Individual problems were hard to identify</td>
<td>-</td>
</tr>
<tr>
<td><strong>External</strong></td>
<td></td>
</tr>
<tr>
<td>Result are limited to one framework</td>
<td>Learning OO framework are similar</td>
</tr>
<tr>
<td>Not representing industrial frameworks</td>
<td>JHotDraw was designed by experience framework designer</td>
</tr>
<tr>
<td>Not perform in a way that is industrial developers</td>
<td>Experience students is comparable to industrial developers</td>
</tr>
</tbody>
</table>
Study 1: identifying the problems (4/4)

• Data analysis
  • Derive reuse related problem from **collected text**
    • Results are 209 problems
  • Manual **data clustering process** on 209 problems
    • Generic types are appear
    • The types can be targeted by high level documentations

• Results

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping</td>
<td>Find the part of framework to work on</td>
</tr>
<tr>
<td>Interactions</td>
<td>Collaboration between parts of framework</td>
</tr>
<tr>
<td>Functionality</td>
<td>What specific part actually do</td>
</tr>
<tr>
<td>Architecture</td>
<td>Understanding high level design constraints</td>
</tr>
</tbody>
</table>
Study 2 : addressing the problems (1/5)

- Two adapted document techniques for JHotDraw
  - Pattern language
    - To address *mapping* and *architectural* problems
  - Micro-architecture description
    - To address *functionality* and *interaction* problems

- Seven individual framework re-user

<table>
<thead>
<tr>
<th>Participant</th>
<th>Represent</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undergraduate</td>
<td>Experienced developer with JHotDraw</td>
<td>2</td>
</tr>
<tr>
<td>Postgraduate</td>
<td>Wide spectrum of experience</td>
<td>3</td>
</tr>
<tr>
<td>Software architecture lecturer</td>
<td>Framework domain friendly developer</td>
<td>2</td>
</tr>
</tbody>
</table>
Study 2 : addressing the problems (2/5)

- **Experiment table**

<table>
<thead>
<tr>
<th>Person</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Create Blocks World (without coding)</td>
</tr>
<tr>
<td>Period</td>
<td>3 hours</td>
</tr>
<tr>
<td>Data Collect</td>
<td>Video to capture documentation reading</td>
</tr>
<tr>
<td></td>
<td>Audio for the talk aloud protocol</td>
</tr>
<tr>
<td>Data Amount</td>
<td>21 hours of Video and audio</td>
</tr>
</tbody>
</table>

- **Plan of analysis**
  - Transcribe the data
    - Into a textual format
  - Cluster data
    - Into categories
  - Use visualization
### Study 2: addressing the problems (3/5)

- **Threat to Validity**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internal</strong></td>
<td></td>
</tr>
<tr>
<td>Unfamiliar documentation</td>
<td>Participants were provided with a period at the start</td>
</tr>
<tr>
<td>Selective coverage of the framework</td>
<td>The task was created to cover a wide range of framework</td>
</tr>
<tr>
<td>Talk aloud intrusion</td>
<td></td>
</tr>
<tr>
<td>Lack of coding</td>
<td></td>
</tr>
<tr>
<td><strong>External</strong></td>
<td></td>
</tr>
<tr>
<td>Selection effect</td>
<td>Can perform similar study without the user’s consent</td>
</tr>
<tr>
<td>Different from genuine reuse task</td>
<td></td>
</tr>
</tbody>
</table>
Study 2: addressing the problems (4/5)

- Data analysis of transcripts

<table>
<thead>
<tr>
<th>Time</th>
<th>Documentation accessed</th>
<th>Talk aloud comments</th>
<th>Non-verbal observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>PL overview</td>
<td>First thing that I’m thinking about is representing the ground. I’m guessing it will be some sort of figure. I’m going to look in the pattern language</td>
<td>Scanning Figure hierarchy</td>
</tr>
<tr>
<td></td>
<td>PL identifying existing figures</td>
<td>It’s annoying me it’s too big (laugh)</td>
<td>Scrolling around</td>
</tr>
<tr>
<td>35</td>
<td>PL overview</td>
<td>I was thinking about looking at the documentation</td>
<td></td>
</tr>
</tbody>
</table>

Possible to understand what problem & what documentation addressed it
Study 2: addressing the problems (5/5)

- Detailed observations
  - Confirmation for four categories of framework reuse problem
    - All problems match into four categories

<table>
<thead>
<tr>
<th>Focus</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Past experience</td>
<td>Solving the problem depend on past experience prone to fault</td>
</tr>
<tr>
<td>Pattern language</td>
<td>Informing user about the availability of parts, example help to identify existing class for reuse and get insight to code</td>
</tr>
<tr>
<td>Micro-architecture</td>
<td>Easy to understand small interaction, but not to large one</td>
</tr>
</tbody>
</table>
Conclusion

- Contributions
  - Make sure the problem categories in OO framework reuse
    - What should be understood by user
  - Try to resolve identified problems of reuse
    - Combined document techniques are experimented

- Future research
  - Validate four categories across range of domains
  - Refine documentation techniques better address the problems
Discussion

- Clarify past experience of problems in programming
  - Functionality and interaction
    - Visible by source code and easy to understand
  - Mapping and architectural
    - Invisible and understand by intuition

- Four problem addressing documentation
  - Really be useful to understand program effectively
Frameworks

- Application framework
  - Network communications
  - Graph modeling
  - Drawing editors
- Middleware
  - CORBA
- Program development environments
  - Eclipse
- Common programming activities
  - JDBC
  - Swing/AWT
  - ASP.NET
Documentations for Framework

- Source code browsers
- Javadoc
- UML diagrams
- Design patterns
  - Pattern languages
- Example based learning
Abbreviation in related work

- **COODBSE**
  - the Colloquium on Object-Orientation in DataBases and Software Engineering

- **ECOOP**
  - European Conference on Object-Oriented Programming

- **OOPSLA**
  - conference on Object-Oriented Programming, Systems, Languages, and Applications

- **ECOOP**
  - European Conference on Object-Oriented Programming
Relate work (5/4)

- A Framework Documentation (Butler et al. 2000)
  - Approach
    - Assess the support provided by many documentation techniques
  - Conclusion
    - Single documentation cannot address all aspects of a framework
JHotDraw

- Features
  - Java implementation
  - Approximately 120 classes
  - Semantic drawing editor
    - Creation/edition of geometric and user defined shapes
    - Creation of behavioral constraint
    - Animation
  - Documentations
    - Source code
    - Javadoc listings
    - Brief design pattern-like descriptions
    - Class diagram
    - Four examples
    - Pattern language

Overview of pattern language
Orrery application in study 1

- The individual developer created initially
  - Mechanical model of the solar system
    - A group of planets and orbits
  - JHotDraw
    - Planet by circles
    - Gravity by line connection
    - Orbit by animation of planet moving

- Five divided exercises in the scenario of classroom
  1. Create a default editing application
  2. Create a representation for a planet
  3. Create a tool to add planets in the editor
  4. Create a representation for gravity constraint
  5. Animate orbiting planets
Threats to validity

- **Origin**
  - One of the structure and components of empirical studies
    - Proposed in the paper
      - Empirical studies of software engineering: A Roadmap

- **Meaning**
  - **Internal validity**
    - Obstacles in right collecting/interpreting of experiment data
  
  - **External validity**
    - The study’s results can generalized to outside
Collected data in study 1

- Text about framework reuse problem
  - 28 pages from individual study
  - 770 pages from coursework reports
  - 216 postings from newsgroup
  - 33 pages from project students interview

- Identify real problems from text
  - Filter not relate to framework reuse
  - Extract the verbatim description
    - 209 problems
Data clustering process

- Cluster analysis
  - Purpose
    - Find generic type of many data
  - Method
    - Position data on a grid based on its relationship to other items
    - Merge and split common properties group of data
    - Form stable representation of data
Pattern language in study 2 (1/2)

- The pattern language of JHotDraw
  - Lack in
    - Detailed guidance
    - Coverage of some important topics
    - Light interconnection
  - Improvement in three directions
    - In the completeness of coverage
      - Increased number of patterns
    - In the technical depth of pattern description
      - Detail was added to each pattern
        - Source code examples
    - In the number of relationships between patterns
      - Network of patterns was enriched
Pattern language in study 2 (2/2)

- Overview

Address mapping and architecture problems

Descriptions & Examples

JHotDraw Pattern Language
Creating Handles

A selected figure displaying resize handles

Direct manipulation of figures on a drawing is achieved through the use of Handles. The AbstractHandle class implements the Handle interface and provides default behaviour for all handles in the framework.

- For further information about the Handle hierarchy see The Handle Hierarchy.

JHotDraw predefines several types of handle; they include ChangeConnectionHandle, ElbowHandle, LocatorHandle and PolygonHandle. It should be noted that because Handles tend to be specific to the figure they were created for, opportunities to reuse across different types of figures are rare. Therefore developers should expect to have to write their own handles either by subclassing AbstractHandle or one of the above classes.

To add a handle to a figure the figures handles() method must be overridden. This method is called by other parts of the framework to draw the selected figures handles on the DrawingView.

Resize handles are often required for figures in JHotDraw applications therefore the framework provides a utility class (BoxHandleKit) which simplifies adding resize handles to a figure.

- How to add handles to a figure (example from GroupFigure).

```java
public Vector handles() {
    Vector handles = new Vector();
    handles.addElement(new GroupHandle(this, RelativeLocator.northWest()));
    handles.addElement(new GroupHandle(this, RelativeLocator.northEast()));
    handles.addElement(new GroupHandle(this, RelativeLocator.southWest()));
    handles.addElement(new GroupHandle(this, RelativeLocator.southEast()));
    return handles;
}
```

- How to add handles to a figure (using BoxHandleKit).

```java
public Vector handles() {
    Vector handles = new Vector();
    BoxHandleKit.addHandles(this, handles);
    return handles;
}
```

When creating custom handles the dynamic behaviour of a handle has to be understood. Handles define three important methods: invokeStart(), invokeStep() and invokeEnd(). These methods are called when the mouse is respectively clicked, dragged and released on top of a handle. Every interaction with a handle will therefore follow a sequence where invokeStart will be called. invokeStep may be called (if the mouse is dragged) and invokeEnd will be called when the interaction ends. This granularity across the interaction allows the developer to control how the handle responds to the user input.

The appearance of a handle can also be altered. This might be appropriate to indicate the action the handle will perform or the current state the handle is in. To change a handles appearance override its draw(Graphics) method.

To create handles at a position on a figure the locate() method should be redefined. This method returns a point around which the Handle will be centred.

- For more information on positioning Handles see [Using Locator]
Micro-architecture in study 2 (1/2)

- Micro-architecture concept
  - Key aspects of document
    - Behavior of object collaborations
  - Representation
    - Decompose the framework into smaller subsystems
      - Base on the key interactions
      - Document each subsystems in isolation

- Develop a micro-architecture based documentation
  - Comprising three views with linkage to source code
    - An interface view
    - A call graph view
    - A hierarchy view
Micro-architecture in study 2 (2/2)

- Overview

Call graph view

Address Interaction and Functionality problems

Interface and hierarchy view
Block World application in study 2

- A simple abstraction of the geometric problem
  - Positioning blocks on a ground

- Why select this application as task
  - The task fits well within the domain of JHotDraw framework
  - The task was clear and simple enough
    - Easy to understand by participants
    - Able to produce solutions within time constraints

An example blocks world