Component Reuse in Software Design: An observational Study

By

D. Budgen & A. Pohthong

Led by: Tanmay Bhowmik

Mississippi State University
Outline

• Component
• Different component-based development strategies
• The empirical study
• Authors’ suggestions
• Some discussion about the study
• Question discussion
This paper presents

• An empirical study that the authors have performed to investigate how designers’ strategies evolve using Unix processes as components (according to the abstract)
What is a component?
What is a component?

• Oxford dictionary says, “An artifact that is one of the individual parts of which a composite entity is made up”

• Pretty vague term in our field of software engineering

• Wide variation in the form, may be
  • Subsystems
  • Processes
  • Objects
  • Idea or concept…..etc.
Component cont.

• Widely used in hardware area to save fabrication cost
• Ease of changing is a problem for software
  – Too soft to be standardized
• As Victor Basili said:
  reused verbatim $\rightarrow$ slightly modified $\rightarrow$
  extensively modified $\rightarrow$ new
Component cont.

• Primary reuse barriers are nontechnical
  – New technology requires due consideration of organizational factors

• Sometimes creation might be easier than looking for an existing one
Component-based development strategy

- Framework First (FF)
- Element First (EF)
- Opportunistic Strategy
- Labels for plan
Framework First (FF)

- Determine the architectural form of solution
- Choose the components that conform to the needs of the architecture
  - To build a building the framework might be:
Element First (EF)

• Identify set of suitable components or elements
• Decide a most suitable architectural form to use the components or elements
• Example??
• Restricted scope??
Strategy cont.

• Opportunistic Strategy
  – Problem driven rather than method driven
  – One step at a time

• Labels for plans
  – Reusing previous experience

• Opportunistic + Element First $\rightarrow$ Macgyver Strategy?
  – Know the current situation
  – Know the goal
  – Identify the available resources/components
  – Proceed one step at a time?
The Empirical Study

• The resources were:
  – Unix utilities together with shell programming
  – ViBSSE: A shell script editor providing searching facilities and covert data collection facility about user actions
  – A group 8 subjects (in their 1st study, the second study involved the most experienced 2) with varying degrees of expertise
The ViBSSE
Experimental tasks

• Task 1
  – Develop a text count utility to display count of lines, sentences, words/exact matches of words, and characters in a given file with an optional string pattern

• Task 2
  – Develop a count utility that can display a count of directories, normal files, linked files, and empty files found in specified path name, depending on the options chosen
Collected data

• User action log
  – Logging time, tools used, keystroke counts, etc.

• Task records
  – Logging shell script whenever edit buffer is saved

• Observation of the subjects

• Interviewing
Evolution of solution: Subject 1, task 1
Evolution of solution: Subject 2, task 1

- Probably did not have prior idea about the use of awk
Comparison table

<table>
<thead>
<tr>
<th>Subject</th>
<th>Tools</th>
<th>Study 1</th>
<th>Study 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Task 1</td>
<td>Task 2</td>
</tr>
<tr>
<td>S1</td>
<td>SEA</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>IND</td>
<td>73</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>GTN</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>FDM</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>PRO</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Grade</td>
<td>Strategy</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EF</td>
<td>FF</td>
</tr>
<tr>
<td>S2</td>
<td>SEA</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>IND</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>GTN</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>FDM</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>PRO</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Grade</td>
<td>Strategy</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EF</td>
<td>(EF)</td>
</tr>
</tbody>
</table>
Interpretation of subjects’ behavior

• Neither subject used any strategy exclusively
• Adopted whatever strategy seemed appropriate at that time (opportunistic?)
• Strategy for the second task seemed more organized (labels for plans?)
• More frequent reference to the documents to assist design decisions if the components are not developed by the users
Authors’ suggestion to leverage component reuse

• Tool support for a range of design strategies
• Availability of a range of search mechanisms for finding components
• Proper and clear component documentation
Validity analysis of this study

- There are four types of threats to validity of an Empirical research:
  - Internal validity
  - Construct validity
  - External validity
  - Conclusion validity
- Authors admit that it has an external validity threat
- Only 2 subjects….!
- No statistical analysis on the collected data
- Conclusions are drawn based on observation only—may be some conclusion validity threat as well
Methods of validation

• According to M. V. Zelkowitz [1, 2] and D. R. Wallace [2]
  – Project monitoring
  – Case study
  – Field study
  – Literature search
  – Legacy
  – Lessons learned
  – Static analysis
  – Replicated
  – Synthetic
  – Dynamic analysis
  – Simulation
  – Assertion
  – Theoretical
The empirical context

- Measurement scales [3]:
  - Nominal scale
  - Ordinal scale
  - Interval scale
  - Ratio scale
Questions

• Why do you think the author says that the same money-saving benefits for reusing hardware do not exist for reusing software? Do you agree with him? After all software and firmware are both logical components of a system.

• Why do you think the people that were more experienced in UNIX in this study did more poorly at the reuse aspect and did not use the Element First strategy of the novices?

• Do you think this study would have provided different results if the programmers were using an actual programming language instead of the UNIX scripting environment? If so, what do you think would happen?
Questions cont.

• How the fact that selecting more experienced people for the second study can affect the way that people organize the problem-solving activities in a component-based software development? Why didn’t they choose all range of experience people?

• What other empirical or non-empirical techniques could be used to do the experiments?

• What are the threats of validity in this study (internal, external, construct)?

• Is using student subject justified in such studies?

• I quite not understand the meaning of figure 1, what is “Basili's spectrum of reuse”? (the second last paragraph of page 4).
Questions cont.

• What are the differences between “problem driven” and “method-driven”?
• Considering the fact that the notion for component is wide-ranging, so what is the relation among the component reuse, feature-oriented reuse and requirement reuse? Is component reuse a superset of the other two?
• Which strategy is preferable (Framework first or Element first). Why?
• What is the significance of component documentation, when the designer making decisions?
• What are the major advantages of using components?
• Why we employ an Opportunistic strategy?
Questions cont.

• With the wide interpretation of the word “component”, how has languages like UML compensated for this?
• What is DESMET classification of evaluation practices?
• Why is this not called a "case study in problem solving?“
• What are the other types of methods similar to Component Reuse, used for software design?
References


Thank you!