An Empirical Study of Architectural Decay in Open-Source Software

Duc M. Le

Annual Research Review 4/2017
Center for Systems and Software Engineering

USC Viterbi
School of Engineering
Motivation

Explicitly manifest as Issues

Problems

- Produce unwanted behavior
- Impact the system’s performance
- Cause crashes

- Not necessarily have an outwardly-visible effect
- Impact the system and its maintainability in more subtle ways
- Gradually create vulnerable parts called “hot spots”
- However, in the past, above impacts were reported only based on folklore and experience
Motivation

Explicitly manifest as Issues

Problems

Implicitly result in “smells”
Expectation vs. Reality

- Designed Architecture (clean, organized) vs. Recovered Architecture (chaotic)
Architectural Recovery

- Reflect knowledge of the systems from **multiple viewpoints**
- **PKG (Package Structure):** reflect developer’s perception of a system’s structure
- **ACDC (Algorithm for Comprehension-Driven Clustering):** base on subsystem patterns
- **ARC (Architectural Recovery using Concerns):** base on information retrieval techniques
- Architecture recovery techniques enable the study of ‘as-is’ architectures during a system’s evolution
- Defects in ‘as-is’ architecture can reveal hidden hot spots among systems’ entities
**Architectural Decay**

- The consequence of a number of **bad design practices** or **bad programming practices** during system development and refactoring
- **Negatively impacts** the system’s behavior, performance, and maintainability
- Symptoms observed are known as **architecture “smells”**

---

**Dependency Cycle**

**Link Overload**

**Scattered Parasitic Functionality**
Categories of architectural smells

- Ambiguous Interface
- Unused Interface
- Unused Brick
- Sloppy Delegation
- Brick Functionality Overload
- Lego Syndrome

- Dependency Cycle
- Link Overload

- Duplicate Functionality
- Logical Coupling

- Scattered Parasitic Functionality
- Concern Overload
How to use architecture?

Recovered architectures

Find “smells”

- Architectural smells
- Smells vs. Issues
- Prediction models
ARCADE Framework

- Architecture Recovery with ARCADE
  - Extract multiple architectural views from a system’s codebase
  - Detect architectural smells during the system’s evolution as reflected in those views
  - Study relationships between smells and issues
  - Observe trends of smells and decay
Subject System Analyzed in our Study

<table>
<thead>
<tr>
<th>System</th>
<th>Domain</th>
<th>No. of Versions</th>
<th>No. of Issues</th>
<th>Avg. SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camel</td>
<td>Integration Framework</td>
<td>78</td>
<td>9665</td>
<td>1.13M</td>
</tr>
<tr>
<td>Continuum</td>
<td>Integration Server</td>
<td>11</td>
<td>2312</td>
<td>463K</td>
</tr>
<tr>
<td>CXF</td>
<td>Service Framework</td>
<td>120</td>
<td>6371</td>
<td>915K</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Data Process Framework</td>
<td>63</td>
<td>9381</td>
<td>1.96M</td>
</tr>
<tr>
<td>Nutch</td>
<td>Web Crawler</td>
<td>21</td>
<td>1928</td>
<td>118K</td>
</tr>
<tr>
<td>OpenJPA</td>
<td>Java Persistence</td>
<td>20</td>
<td>1937</td>
<td>511K</td>
</tr>
<tr>
<td>Struts2</td>
<td>Web App Framework</td>
<td>36</td>
<td>4207</td>
<td>379K</td>
</tr>
<tr>
<td>Wicket</td>
<td>Web App Framework</td>
<td>72</td>
<td>6098</td>
<td>332K</td>
</tr>
</tbody>
</table>
Architectural smells are undesirable

- **Hypothesis H1.** Smelly files are more likely to be issue-prone than non-smelly files
  - At 5% (or less) level of significance, this assumption is correct for all subject systems

- **Hypothesis H2.** Smelly files are more likely to be change-prone than non-smelly files
  - At 5% (or less) level of significance, this assumption is correct for most of subject systems
Architectural Smells vs. Buggy Rates

- **Hypothesis H1.** Smelly issues are more likely to be **bug-prone** than non-smelly files

<table>
<thead>
<tr>
<th>System</th>
<th>ACDC</th>
<th>ARC</th>
<th>PKG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Smelly</td>
<td>Clean</td>
<td>Factor</td>
</tr>
<tr>
<td>Camel</td>
<td>58.8%</td>
<td>55.7%</td>
<td>1.05x</td>
</tr>
<tr>
<td>Continuum</td>
<td>69.7%</td>
<td>53.3%</td>
<td>1.31x</td>
</tr>
<tr>
<td>CXF</td>
<td>79.4%</td>
<td>62.9%</td>
<td>1.26x</td>
</tr>
<tr>
<td>Hadoop</td>
<td>71.7%</td>
<td>57.7%</td>
<td>1.24x</td>
</tr>
<tr>
<td>Nutch</td>
<td>70.6%</td>
<td>49.7%</td>
<td>1.42x</td>
</tr>
<tr>
<td>OpenJPA</td>
<td>77.6%</td>
<td>70.5%</td>
<td>1.10x</td>
</tr>
<tr>
<td>Struts2</td>
<td>59.8%</td>
<td>45.6%</td>
<td>1.31x</td>
</tr>
<tr>
<td>Wicket</td>
<td>66.5%</td>
<td>65.1%</td>
<td>1.02x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>ACDC</th>
<th>ARC</th>
<th>PKG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camel</td>
<td>9</td>
<td>0</td>
<td>51</td>
</tr>
<tr>
<td>Continuum</td>
<td>3</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>CXF</td>
<td>34</td>
<td>0</td>
<td>131</td>
</tr>
<tr>
<td>Hadoop</td>
<td>6</td>
<td>0</td>
<td>46</td>
</tr>
<tr>
<td>Nutch</td>
<td>5</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>OpenJPA</td>
<td>9</td>
<td>0</td>
<td>46</td>
</tr>
<tr>
<td>Struts2</td>
<td>3</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Wicket</td>
<td>11</td>
<td>0</td>
<td>76</td>
</tr>
</tbody>
</table>

- Not all smells necessary lead to bugs
- Coupling-based smells are likely to cause bugs
Architectural smells along a system’s evolution

- Percentage of smelly files in Camel (ACDC view). System versions are shown along the bottom; The numbers of system files are on the left; The percentages are on the right.
  - Significant signs of architectural decay were present starting with the initial versions
  - The number of smelly files tends to increase slowly as the size of the system increases over time
Architectural smells along a system’s evolution

- The top-5 long-lived smelly files in Hadoop (left) and Struts2 (right). The X-axis indicates system versions; The Y-axis indicates the number of smells.
  - Long-lived smelly files tend to **continuously be involved** in issues during a system’s lifetime
  - The numbers of issues related to those files are at some times **very large**
Thank you!

• Q&A