An Empirical Study of Software Architecture Change in Open-Source Software Systems

Duc Minh Le¹, Pooyan Behnamghader¹, Joshua Garcia²
Arman Shahbazian¹, Daniel Link¹, Nenad Medvidovic¹

University of Southern California¹
George Mason University²

The 12th Working Conference on Mining Software Repositories (MSR 2015)
Outline

• Introduction and Motivation

• Architecture Recovery, Change, and Decay Evaluator (ARCADE) Framework

• Empirical Study of Software Architecture Change

• Summary and Future Work
Architecture of Two Versions of a Software
How Many Systems End Up

C. A. Mattmann et al., Revisiting the Anatomy and Physiology of the Grid
In Journal of Grid Computing, March 2015, Volume 13, Issue 1, pp 19-34
Software Architecture

• **Prescriptive vs. Descriptive Architecture**
  • Designed vs. Implemented
  • Intent vs. Realization

• **Architecture Recovery**: The process of determining a system’s architecture from its implementation-level artifacts

• “As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.” (Lehman)
Architecture Change

• **Introduction of design decisions** into the descriptive architecture that are either
  – Unforeseen by the prescriptive architecture (drift)
  – Violating the prescriptive architecture (erosion)

• Architecture change can **point to factors** that cause architecture decay

• Benefits of knowing your architecture and how it changes
  – Less unexpected complexity
  – Reduces maintenance effort
  – Regain full control of development
Outline

• Introduction

• Architecture Recovery, Change, and Decay Evaluator (ARCADE) Framework

• Empirical Study of Software Architecture Change

• Summary and Future Work
What is ARCADE

• **A toolset** for software architecture comprehension

• ARCADE provides following features
  – **Software architecture recovery**
    • Integrates 10 recovery methods
    • Supports Java and C code
  – **Architecture change metrics**
    • System level
    • Component level
  – Architectural smell detection
  – Decay metrics
  – Mining implementation issues
The Big Picture of ARCADE

ARCADE Toolset

ARCADE Runner

SoftEvo
ARCADE Runner

- **GUI front end** for ARCADE
  - Makes using it more fun and easy
  - Prevents errors
  - Generates cluster graphs and counts code with UCC “for free” (in separate threads/processes)

- Requirements: Java 8
SoftEvo

- A shell script toolset
  - Uses Amazon EC2 cloud
  - Semi-automatically runs large-scale software architecture recovery and analyses

- SoftEvo’s features:
  - Downloads the source code
  - Compiles different versions of an application
  - Recovers the architecture
  - Analyzes the result
  - Highly portable, modifiable, parallel, and fast
Demo Time – ARCADE Runner
Demo Time - SoftEvo
Outline

• Introduction and Motivation

• Architecture Recovery, Change, and Decay Evaluator (ARCADE) Framework

• Empirical Study of Software Architectural Change

• Summary and Future Work
Architectural Change Metrics

- **a2a** – architecture-to-architecture metric
  - System-level
  - A **distance measure** between two architectures, inspired by MoJoFM metric*

- **cvg** – cluster coverage metric
  - Component-level
  - The extent to which certain components **existed in an earlier version** of a system or were **added in a later version**

* Z. Wen et al., An effectiveness measure for software clustering algorithms, In International Workshop on Program Comprehension (IWPC), 2004
a2a vs. cvg

• **a2a** measures architectural distance based on **five operations**
  – Additions, removals and moves of implementation-level entities
  – Additions and removals of clusters

• **cvg** metric is based on similarity between two components

\[
a2a = 1 - \frac{mto(V \downarrow 1, V \downarrow 2)}{mco(V \downarrow 1) + mco(V \downarrow 2)} = 1 - \frac{3}{9\times7} = 83%
\]

\[
cvg(V \downarrow 1, V \downarrow 2) = \frac{3}{4} = 75%
\]
Empirical Study Setup

• Study architecture change in
  – 14 Apache open-source software systems (FOSS)
  – 572 versions
  – 3 architecture recovery techniques
    • ARC, ACDC, PKG

• Research questions
  – In what ways do architectures change at the system level and at the component level?
  – Do architecture changes at the system and component levels occur concurrently?
  – When does significant architecture change occur?
<table>
<thead>
<tr>
<th>System</th>
<th>Application Domain</th>
<th>Versions</th>
<th>Time</th>
<th>MSLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveMQ</td>
<td>Message Broker</td>
<td>20</td>
<td>8/04-12/05</td>
<td>3.4</td>
</tr>
<tr>
<td>Cassandra</td>
<td>Distributed DBMS</td>
<td>127</td>
<td>9/09-9/13</td>
<td>22.0</td>
</tr>
<tr>
<td>Chukwa</td>
<td>Data Monitoring</td>
<td>7</td>
<td>5/09-2/14</td>
<td>2.2</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Data Processing</td>
<td>63</td>
<td>4/06-8/13</td>
<td>30.0</td>
</tr>
<tr>
<td>Ivy</td>
<td>Dependency Manager</td>
<td>20</td>
<td>12/07-2/14</td>
<td>0.4</td>
</tr>
<tr>
<td>JackRabbit</td>
<td>Content Repository</td>
<td>97</td>
<td>8/04-2/14</td>
<td>34.0</td>
</tr>
<tr>
<td>Jena</td>
<td>Semantic Web Framework</td>
<td>7</td>
<td>6/12-9/13</td>
<td>2.7</td>
</tr>
<tr>
<td>JSPWiki</td>
<td>Wiki Engine</td>
<td>54</td>
<td>10/07-3/14</td>
<td>1.2</td>
</tr>
<tr>
<td>Log4j</td>
<td>Logging</td>
<td>41</td>
<td>01/01-06/14</td>
<td>2.4</td>
</tr>
<tr>
<td>Lucene</td>
<td>Search Engine</td>
<td>21</td>
<td>12/10-1/14</td>
<td>5.1</td>
</tr>
<tr>
<td>Mina</td>
<td>Network Framework</td>
<td>40</td>
<td>11/06-11/12</td>
<td>2.3</td>
</tr>
<tr>
<td>PDFBox</td>
<td>PDF Library</td>
<td>17</td>
<td>2/08-3/14</td>
<td>2.7</td>
</tr>
<tr>
<td>Struts</td>
<td>Web Apps</td>
<td>36</td>
<td>3/00-2/14</td>
<td>6.7</td>
</tr>
<tr>
<td>Xerces</td>
<td>XML Library</td>
<td>22</td>
<td>3/03-11/09</td>
<td>2.3</td>
</tr>
</tbody>
</table>
**Idealized Versioning Scheme of FOSS**

- **Major.Minor.Patch** release*
  - Major version:
    - Any change can be made
    - Significant change
  - Minor version
    - Introduce new features
    - Smaller change than major
  - Patch version
    - Bug fix release
    - Smallest change
- Prerelease version: alpha, beta, release-candidate

* http://apr.apache.org/versioning.html

Does significant architecture change occur between **minor system versions** within a **single major version**?

* 1.5.0
* 1.5.1
* 1.6.0
* 1.6.1
* 2.0.0
Recovery Techniques

• **PKG** – package structure recovery

• **ACDC** – algorithm for comprehension-driven clustering

• **ARC** – architecture recovery using concerns

* V. Tzerpos et al., ACDC: an algorithm for comprehension-driven clustering, In Working Conference on Reverse Engineering (WCRE), 2000

** J. Garcia et al., Enhancing architectural recovery using concerns, In International Conference on Automated Software Engineering (ASE), 2011
RQ1 – How Architecture Changes

- **Different views** of the system architecture **complement** each other

### Average a2a values between versions

<table>
<thead>
<tr>
<th>System</th>
<th>ACDC</th>
<th>ARC</th>
<th>PKG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Major</td>
<td>MinMaj</td>
<td>Minor</td>
</tr>
<tr>
<td>ActiveMQ</td>
<td>62</td>
<td>69</td>
<td>95</td>
</tr>
<tr>
<td>Cassandra</td>
<td>42</td>
<td>80</td>
<td>77</td>
</tr>
<tr>
<td>Chukwa</td>
<td>-</td>
<td>-</td>
<td>78</td>
</tr>
<tr>
<td>Hadoop</td>
<td>17</td>
<td>73</td>
<td>86</td>
</tr>
<tr>
<td>Ivy</td>
<td><strong>50</strong></td>
<td>67</td>
<td>91</td>
</tr>
<tr>
<td>JackRabbit</td>
<td>38</td>
<td>76</td>
<td>84</td>
</tr>
<tr>
<td>Jena</td>
<td>-</td>
<td>-</td>
<td>88</td>
</tr>
<tr>
<td>ISPWiki</td>
<td>18</td>
<td>30</td>
<td>86</td>
</tr>
<tr>
<td>Log4j</td>
<td>9</td>
<td>13</td>
<td>64</td>
</tr>
<tr>
<td>Lucene</td>
<td><strong>12</strong></td>
<td>8</td>
<td>96</td>
</tr>
<tr>
<td>Mina</td>
<td>28</td>
<td>30</td>
<td>92</td>
</tr>
<tr>
<td>PDFBox</td>
<td>-</td>
<td>-</td>
<td>97</td>
</tr>
<tr>
<td>Struts2</td>
<td>-</td>
<td>-</td>
<td>90</td>
</tr>
<tr>
<td>Xerces</td>
<td>21</td>
<td>54</td>
<td>92</td>
</tr>
<tr>
<td><strong>AVG</strong></td>
<td><strong>30</strong></td>
<td><strong>50</strong></td>
<td><strong>87</strong></td>
</tr>
</tbody>
</table>

Value unit is percentage
Lower numbers mean more change
Empty table cells indicate versions that do not exist for a given system
RQ2- System vs. Component Level

- **Architecture changes** occur **within software components** even when the system’s overall architectural structure remains relatively stable.

- **Semantic-based** architectural perspective provides an **addition insight** into the system beside structure-based ones.

Ivy
RQ3 – When Significant Change Occurs

- **Dramatic architecture change** can occur **across minor versions** of a software system.

- Engineers **do not consider** architecture change as a factor when determining version numbers.

<table>
<thead>
<tr>
<th>System</th>
<th>ACDC</th>
<th>ARC</th>
<th>PKG</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveMQ</td>
<td>86</td>
<td>78</td>
<td>84</td>
</tr>
<tr>
<td>Cassandra</td>
<td>60</td>
<td>55</td>
<td>49</td>
</tr>
<tr>
<td>Chukwa</td>
<td>72</td>
<td>73</td>
<td>76</td>
</tr>
<tr>
<td>Hadoop</td>
<td>57</td>
<td>51</td>
<td>72</td>
</tr>
<tr>
<td>Ivy</td>
<td>85</td>
<td>65</td>
<td>79</td>
</tr>
<tr>
<td>JackRabbit</td>
<td>76</td>
<td>69</td>
<td>74</td>
</tr>
<tr>
<td>Jena</td>
<td>83</td>
<td>77</td>
<td>86</td>
</tr>
<tr>
<td>JSPWiki</td>
<td><strong>47</strong></td>
<td><strong>55</strong></td>
<td><strong>58</strong></td>
</tr>
<tr>
<td>Log4j</td>
<td>62</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>Lucene</td>
<td>96</td>
<td>89</td>
<td>90</td>
</tr>
<tr>
<td>Mina</td>
<td>93</td>
<td>92</td>
<td>98</td>
</tr>
<tr>
<td>PDFBox</td>
<td>87</td>
<td>87</td>
<td>87</td>
</tr>
<tr>
<td>Struts2</td>
<td>79</td>
<td>80</td>
<td>83</td>
</tr>
<tr>
<td>Xerces</td>
<td><strong>41</strong></td>
<td><strong>37</strong></td>
<td><strong>48</strong></td>
</tr>
</tbody>
</table>

**AVG** | 73 | 69 | 75

Minimum a2a values between minor versions.
Outline

• Introduction and Motivation

• Architecture Recovery, Change, and Decay Evaluator (ARCADE) Framework

• Empirical Study of Software Architecture Change

• Summary and Future Work
Summary

• We introduced ARCADE, a novel automated workbench for software architecture recovery and analysis.

• Our study, facilitated by ARCADE, presented the largest empirical study to date of architecture change in long-lived software systems.

• The study resulted in several important findings:
  – FOSS versioning scheme is not an accurate indicator of architecture change.
  – The package structure is not a complete representation of the system’s architecture.
  – Our study points to the significance of a semantics-based architectural perspective.
Future Work

• Integrate more *architecture recovery* and *code analysis* methods
  – Additional architectural constructs

• Catalogue of *smells* and *patterns indicating decay*
  – *Correlation* of architecture change & decay with code decay, refactoring activities, and implementation issues

• Leverage ARCADE to enable *prediction of*
  – Architectural decay
  – Major architecture change
Q&A

• Contact
  – General questions: ducmle@usc.edu
  – Arcade Runner: dlink@usc.edu
  – SoftEvo: pbehnamg@usc.edu
What’s Going On?

• Introduction of design decisions into the descriptive architecture that are either
  – **Unforeseen** by the prescriptive architecture (**drift**)  
  – **Violate** the prescriptive architecture (**erosion**)

• **Decaying** systems begin to “smell” on the levels of architecture and code
  – Will not necessarily lead to bugs
  – But have been shown to **negatively impact** a system’s lifecycle properties
SoftEvo (cont.)

The Pipeline of the Architecture of SoftEvo

1. Get source
2. Get commits/versions/tags list
3. Download/Compile/Recover
4. Architectural change and decay metric analyses
5. Recover selected versions
6. Analyze versions hierarchy
7. Statistical Analysis
8. Represent Results
9. Start

Cloud

Instance
cvg – Component Level Metric

• Based on \( c2c(c_1, c_2) \) – similarity of components \( c_1 \) and \( c_2 \)
• \( cvg(v_k, v_{k+1}) \) – coverage metric
  • Percentage of covered components in version \( v_k \) at threshold \( th_{cvg} \)

With \( th_{cvg} = 80\% \), \( cvg(v_1, v_2) = 66\% \)