Identifying and Protecting Architecturally Significant Code

Software Archeology

Mehdi Mirakhori, Jane Cleland-Huang
DePaul University

Contact me: mehdi@cs.DePaul.edu
Architectural Failures

One Illinois hospital jointly managed by the Departments of Veterans Affairs (“VA”) and Defense (“DOD”) failed to achieve ‘interoperability’ between the Departments’ EHR systems, costing the hospital at least $700,000 annually.

This is despite the fact that the DOD and VA have already spent $100 million to achieve this quality.
A few days after the launch of the federal government's Obamacare website, millions of Americans that were looking for information about new health insurance plans were locked out of the system even though the designers of HealthCare.gov endeavored to fix the problem and enhance the availability.

Was it just availability issue?
"I identified a series of steps that could be easily automated to collect usernames, password reset codes, security questions, and email addresses from the system -- without any kind of authentication."

SEBELIUS: “And we immediately corrected that problem, so there wasn't a -- it was a theoretical problem that was immediately fixed. I would tell you we are storing the minimum amount of data, because we think that's very important. The hub is not a data collector. It is actually using data centers at the IRS, at Homeland Security, at Social Security to verify information, but it stores none of that data, so we don't want to be.....”
Detailed Example: An architectural view

Requirements #2: high throughput access to application data

Decision #1: Use Master-slave Architectural Style where slave processes are replicated

Decision #2: Checkpoint updated data, and bundle replicas (send every 2 seconds) in order to meet performance goals.

Decision #3: Use heartbeat tactic to monitor availability of task trackers and data nodes. Heartbeat must beat every .25 seconds to balance availability and performance.

Decision #4: Use proxy handles failure pattern to shield clients from failures, and to support fault tolerance (i.e. service continues in the face of transient failure).

Requirements #1: highly fault-tolerant, where hardware failure is the norm rather than the exception

Each of these decisions are driven by one or more architectural concern.

Unfortunately, many of them are lost in the architectural design, low level design, and code.

Apache Hadoop Architecture
Detailed Example: Architectural Decay

Architectural Decay

Apache Hadoop architecture

Master

Slave

Hadoop core

Master node

HDFS

NameNode

MapReduce Framework

JobTracker

MapReduce Job

InputFormat Implementations

Database Connector

Task with InputFormat

TaskTracker

Node 1

Node 2

Node n

Database

DataNode

Catalog

Loader

TaskTracker

DataNode
Detailed Example in Hadoop:

Developer #1: DataNodes.java, should send several messages to the NameNode.java. Messages such as block reports, heartbeat, blocks to be deleted etc.

Developer #2: So many messages, lets merge them by piggy-backing

Design Decay & Compromising Availability: block reports are usually delayed, system detects the DataNode failure while it is alive and lunches the recovery process

Developer #3: every 10 seconds DataNode reports data or send an empty message for heartbeat

Developer #4: lets make it every 2 seconds

Design Decay & Performance Tradeoff: Performance issues, tradeoff between availability and performance

Issues Reported: HADOOP-4584, HADOOP-178,…
Ideal World: Architectural information is documented during the Architectural design phase and is updated regularly to reflect the current system architecture.
Real World: Architectural information is outdated and does not reflect the current architecture of the system.
Architectural Decay

Eroded architecture becomes complex, difficult to understand and difficult to maintain.

A big ball of mud: Apache Hadoop architecture
Archie: A Smart IDE to Protect Architecture

The vision initially presented at:
Detect and monitor code snippets that implement key architectural decisions in the source code.

Proactively keep developers informed of underlying architectural decisions during maintenance activities.

Automatically trace external architecture specification documents to the source code or design model.

Perform change impact analysis of architectural concerns at both the code and design level.
Decision Detector: A rigorously validated automated technique based on a combination of machine learning, structural analysis, and pattern matching techniques.

Why it works?: Trained by sample source codes of hundreds open source projects.

Code Snippets

```java
public boolean isAuditUserIdentifyPresent(){
    return(this.auditUserIdentify != null);
}

public BigDecimal getAuditSequenceNumber(){
    return(this.auditSequenceNumber;
```
Detect and monitor code snippets that implement key architectural decisions in the source code.
Detect and monitor code snippets that implement key architectural decisions in the source code.
Archie: A Smart IDE to Protect Architecture

- IDEs and Compilers do well on **Syntactical** issues, a little attention to **Semantic** but **Design Rational** is not covered.

- Archie has **features** for **communicating** architectural knowledge.
- **Visualization** module to depict the seams of a software design, the driving requirements, business goals and rationale behind the source code.

Proactively keep developers informed of underlying architectural decisions during maintenance activities.
Archie: A Smart IDE to Protect Architecture

Proactively keep developers informed of underlying architectural decisions during maintenance activities.
An **asynchronous Event-Based** monitoring and notification **infrastructure** has been designed to proactively inform developers of underlying architectural decisions. An initial proof of concept experiment has been conducted.
Archie: A Smart IDE to Protect Architecture

Perform change impact analysis of architectural concerns at both the code and design level.
Archie: A Smart IDE to Protect Architecture

Perform change impact analysis of architectural concerns at both the code and design level.
Archie: A Smart IDE to Protect Architecture

Perform change impact analysis of architectural concerns at both the code and design level.
Archie: A Smart IDE to Protect Architecture

Perform change impact analysis of architectural concerns at both the code and design level.
Archie: A Smart IDE to Protect Architecture

Perform change impact analysis of architectural concerns at both the code and design level.

We utilized the Hadoop change logs for the past four releases, and simulated a change impact analysis scenarios.
Current Research Technology:

A large body of industry level validated automated trace retrieval techniques, released and examined in Tracelab experimental environment.

Supporting traceability of distributed heterogeneous software artifacts.
Kevin E. Greene  
*Program Manager (SwA), DHS S&T Cyber Security Division (CSD)*

“We’re trying to do our job in protecting our nation’s critical infrastructure and providing capabilities to be more proactive instead of reactive to cyberthreats. Along with the technologies I’m developing, I think the SWAMP will definitely be a revolutionary force in the software assurance community. We anticipate advancing some breakthroughs in the SWAMP,” Kevin Greene declares.
The Software Assurance Marketplace

Archie's Test Assessment Runs

The following assessment runs are currently available for project Archie's Test of any package using any tool on any platform limited to 50 items.

<table>
<thead>
<tr>
<th>Date / Time</th>
<th>Package</th>
<th>Tool</th>
<th>Platform</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-04-29 07:23</td>
<td>JSPWiki 2.5.139</td>
<td>Archie 1.3</td>
<td>Fedora Linux 19 64-bit</td>
<td>Finished</td>
</tr>
<tr>
<td>2014-04-28 14:42</td>
<td>Camel 2.11.1</td>
<td>Archie 1.3</td>
<td>Fedora Linux 18 64-bit</td>
<td>Finished</td>
</tr>
<tr>
<td>2014-04-28 14:26</td>
<td>Swamp Java 1.1.0b</td>
<td>Archie 1.3</td>
<td>Ubuntu Linux 12.04 LTS Lucid Lynx 64-bit</td>
<td>Finished</td>
</tr>
<tr>
<td>2014-04-28 14:14</td>
<td>NIST Juliet Java CWE600_01 1.2</td>
<td>Archie 1.3</td>
<td>Red Hat Enterprise Linux RHEL 6.4 32-bit</td>
<td>Finished</td>
</tr>
<tr>
<td>2014-04-28 14:11</td>
<td>Hadoop 1.1.2</td>
<td>Archie 1.3</td>
<td>Fedora Linux 18 64-bit</td>
<td>Finished</td>
</tr>
<tr>
<td>2014-04-22 19:56</td>
<td>Hadoop 1.1.2</td>
<td>Findbugs 2.0.2 (FindSecurityBugs 1.1.0)</td>
<td>Debian Linux 7.0 64-bit</td>
<td>Finished</td>
</tr>
</tbody>
</table>
"All I'm saying is now is the time to develop the technology to deflect an asteroid."
Identifying and Protecting Architecturally Significant Code

Software Archeology

Mehdi Mirakhorli, Jane Cleland-Huang
DePaul University

Contact me: mehdi@cs.DePaul.edu