Industrial Experiences with Automated Software Architecture Measurement

By Yuanfang Cai and Rick Kazman

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Software Health Management System

From research to practice

- **Metrics for comparison and contrast**
  Monitor the degradation, variation, and evolution of software projects

- **Hotspot detection**
  Pinpoint design flaws with ever-increasing maintenance costs

- **Quantification**
  Calculate the cost of each flaw and estimate the benefits of refactoring
Software Health Management System

Systems we have analyzed

- 252 Open Source Projects
- 50+ Industrial projects
General Work Flow

Overview

Step 1: Data Collection
Code dependency, history, issue record

Step 2: Automated Analysis
Measurement, hotspot detection, cost calculation

Step: Collect Feedback
Surveys and Interviews

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Step 1: Data Collection

1. SDSM Generator
2. HDSM Generator

Step 2: Automated Analysis

3. DL & PC Calculator
4. ArchIssue Detector
5. ArchRoot Detector

DV8 Tool Suite

6. ArchIssue Cost Quantification
7. Arch Debt Quantification

Step 3: Collect Feedback

Surveys
Phone Interviews

Input files
Output files
3rd-party Tool
Metrix
Hotspot
Quantification

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Step 1: Data Accepted by DV8 tool suite

Beyond source code

Dependency information
- Currently support Understand Cytoscape output
- Extensible to other dependency formats through standard interfaces

History information (optional)
- Commit history such as git record, svn record
- Extensible to other history record types through standard interfaces

Issue information (optional)
- Issue tracking info such as Jira record, MTF work item, Bugzilla record etc.
- Extensible to other records through standard interfaces
A DRSpace is composed of a meaningful subset of a system’s files along with the architectural connections among these files.

- Any subset of files may form a design space
- Architectural connections
  - Structural couplings: call, inherit, aggregate, etc..
  - Evolutionary couplings
  - One or multiple types
  - Implicit or explicit
Design Rule Space (DRSpace)

A new architecture model supported by DV8

- Non-trivial software system must contain multiple design spaces:
  - each feature implemented
  - each pattern applied
  - each concern addressed

- Each file can participate in multiple design spaces

- We propose that software architecture can, and should be modeled as multiple, overlapping design spaces

- Each design space can be visualized as a Design Structure Matrix (DSM)
Design Rule Space (DRSpace) in Design Structure Matrix (DSM)

A small calculator example

DRSpace 1: Polymorphism

DRSpace: formed by “extend” and “implement”

File set

File 5: mij.io.InputPipe extends File 2 mij.io.Pipe


[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Design Rule Space (DRSpace)

A small calculator example

DRSpace 1: Polymorphism DRSpace: formed by “extend” and “implement”
Design Rule Space (DRSpace)

A small calculator example

DRSpace 2: Visitor Pattern DRSpace: formed by multiple dependency types

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ext, dp dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ext, dp dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Ext, dp dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Ext, dp dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Ext, dp dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>dp Imp dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>dp dp dp dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>dp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dp dp (10)</td>
</tr>
</tbody>
</table>

Ext: Extend  Impl: Implement  
dp: General dependencies such as call, use, and create

- Visitor interface: mij.ast.TreeVisitor
- Element interface: mij.ast.Node
- Concrete elements of the pattern: m(rc3-7)
- Concrete visitor role: Calculator

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Design Rule Space (DRSpace)

A small calculator example

DRSpace 2: Visitor Pattern DRSpace: Viewing the evolutionary coupling and the modular structure of a DRSpace simultaneously helps to reveal flawed architectural connections.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ext, dp, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ext, dp, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Ext, dp, 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Ext, dp, 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Ext, dp, 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ext: Extend  Impl: Implement
dp: General dependencies such as call, use, and create

- Visitor interface: mij.ast.TreeVisitor
- Element interface: mij.ast.Node
- Concrete elements of the pattern: m(rc3-7)
- Concrete visitor role: Calculator

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Step 2: Automated Architecture Analysis

Overview

Step 2.1 Measure and Monitor
Compare, Contrast, and Monitor

Step 2.2: Pinpoint Hotspots
Detection and Visualization

Step 2.3: Quantify Design Debt
Costs and benefits

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Step 2.1 Measure and Monitor

A new metric suite

- **Decoupling Level (DL)** [Ran et al. 2016]:
  an options-based metric, measuring the system’s ability to generate options

- **Propagation Cost (PC)** [MacCormack 2006]:
  a DSM-based metric, measuring how tightly coupled a system is
The Concept of "Metrics"

What is a "real" metric?

Software "metrics":
Lines of Code (LOC), CK metrics, McCabe complexity

Real metrics
Meter, Pound, Kilogram
The Properties of a Real Metric

**Toward a real metric**

- **Support cross-project comparison**
  - Precisely tell maintainability differences
  - Independent of project domains, sizes, ages, etc.

- **Support Evolution Monitoring**
  - Remain stable for non-refactored versions
  - Non-trivial changes should reflect architecture variation
  - Architectural debt thermometer

- **Form a Maintainability Chart**
  - Similar to real health chart
  - An industrial benchmark

---

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Decoupling Level (DL): Towards a Real Metric

Rationale

💡 Module: A true module (with high option value) should be
  • Small
  • Independent

💡 System: A highly modularized system should
  • Have large numbers of true modules...
  • connected by design rules

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Decoupling Level (DL)

Algorithm

Upper Layer modules:
- The fewer dependents, the higher the value
- The smaller the module, the higher the value

True modules:
- The smaller a true module, the higher the value
- The more true modules, the higher the value

The more files are clustered into true module, the higher the value

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Propagation Cost (PC)

Rationale

Decoupling Level (DL) and Propagation Cost (PC)

From 129 open source projects: 108 open source, 21 industrial
Decoupling Level (DL) and Propagation Cost (PC)  

Cross project comparison

<table>
<thead>
<tr>
<th></th>
<th>Open Source(%)</th>
<th>Commercial(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DL</td>
<td>PC</td>
</tr>
<tr>
<td>Avg</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>Median</td>
<td>58</td>
<td>18</td>
</tr>
<tr>
<td>Max</td>
<td>92</td>
<td>72</td>
</tr>
<tr>
<td>Min</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>20th Pt</td>
<td>47</td>
<td>8</td>
</tr>
<tr>
<td>40th Pt</td>
<td>55</td>
<td>14</td>
</tr>
<tr>
<td>60th Pt</td>
<td>66</td>
<td>21</td>
</tr>
<tr>
<td>80th Pt</td>
<td>75</td>
<td>34</td>
</tr>
</tbody>
</table>

Pt: Percentile

- Best DL (93%) is from industrial
- Worst DL (14%) is from open source
Decoupling Level (DL) and Propagation Cost (PC)

Towards a Health Chart

An industrial project:

DL: 29%, 10th percentile: Confirmed to have severe maintenance difficulty

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Non-trivial variation in DL indicate major architecture variation

- An industrial project: Evolved for 6 year, 29 releases, 1082-1852 files
Evolutionary Monitoring

Non-trivial variation in DL faithfully indicate major architecture variation

1. **DL increases from 45% to 74%:**
   Transforming from a prototype to a real product
Evolutionary Monitoring

Non-trivial variation in DL faithfully indicate major architecture variation

DL decreases from 78% to 68%:
New features were added and Technical debt accumulates
Evolutionary Monitoring

Non-trivial variation in DL faithfully indicate major architecture variation

DL decreases from 65% to 48%:
Unsuccessful refactoring

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Evolutionary Monitoring

Non-trivial variation in DL faithfully indicate major architecture variation

DL increases from 48% to 62%:
“Cleaning up” technical debt
Step 2.2 Pinpoint Hotspots: (1) Flaw Detection

Flaws responsible for high-maintenance

- We automatically identify 6 types of design flaws
  1. Unstable interface
  2. Modularity violation
  3. Crossing
  4. Improper inheritance
  5. Cliques among files
  6. Package cycles

- These flaws are highly correlated with bugs, changes, and churn
Flaw 1: Unstable Interface

Sample Flaw
### Flaw 2: Crossing

#### Sample Flaw

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>path1.File1_h</td>
<td>(1)</td>
<td>2</td>
<td>6</td>
<td>11</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>path1.File2_h</td>
<td>d</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>path1.File3_h</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>path2.File1_h</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>path2.File2_h</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>path2.File3_h</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>path3.File1_h</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>path3.File2_c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>path3.File3_c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>path4.File1_c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>path4.File2_c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>path4.File3_c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>path5.File1_c</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*DISTRIBUTION STATEMENT A* Approved for public release and unlimited distribution.
Flaw 3: Modularity Violation

Sample Flaw
Step 2.2 Pinpoint Hotspots: (2) Root Detection

Design Spaces that Grow and Propagate Errors

Architecture Roots

DRSpaces propagate errors among multiple files through flawed connections.
The more error-prone files are, the more likely they are architecturally connected.

Observations about Roots

File in the DRSpaces led by an error prone file is also error-prone.
A few (~5) DRSpaces capture more than half of the most error-prone files.
Error-prone DRSpaces often have architectural flaws.
Sample Root from JBoss

Each root has multiple design problems

Jboss JDBCCMRFIELDBridge DRSpace
The Impact of Roots in Open Source Projects

The observation is consistent with other projects

Consider JBoss: 5 Root spaces captured 52% of all files with more than 5 bugs
Step 2.3 Quantification

*Calculate the Costs and Benefits*

- Calculate the costs of each root, each flaw and each types of flaws
- Calculate ROI (Return on Investment)
The Costs of Each Flaw and Each Type of Flaw

Sample output from DV8

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Clique</td>
<td>26</td>
<td>322</td>
<td>21%</td>
<td>1,790</td>
<td>28%</td>
<td>26,294</td>
<td>41%</td>
<td>643</td>
<td>34%</td>
<td>16,557</td>
<td>45%</td>
</tr>
<tr>
<td>Crossing</td>
<td>91</td>
<td>368</td>
<td>24%</td>
<td>3,146</td>
<td>50%</td>
<td>40,247</td>
<td>63%</td>
<td>1,051</td>
<td>55%</td>
<td>25,177</td>
<td>68%</td>
</tr>
<tr>
<td>ModularityViolation</td>
<td>667</td>
<td>588</td>
<td>38%</td>
<td>4,538</td>
<td>72%</td>
<td>46,224</td>
<td>72%</td>
<td>1,438</td>
<td>75%</td>
<td>27,648</td>
<td>74%</td>
</tr>
<tr>
<td>PackageCycle</td>
<td>175</td>
<td>499</td>
<td>32%</td>
<td>2,417</td>
<td>38%</td>
<td>29,906</td>
<td>47%</td>
<td>778</td>
<td>41%</td>
<td>18,889</td>
<td>51%</td>
</tr>
<tr>
<td>UnstableInterface</td>
<td>6</td>
<td>316</td>
<td>21%</td>
<td>1,669</td>
<td>26%</td>
<td>19,898</td>
<td>31%</td>
<td>388</td>
<td>20%</td>
<td>11,457</td>
<td>31%</td>
</tr>
<tr>
<td>UnhealthInheritance</td>
<td>72</td>
<td>257</td>
<td>17%</td>
<td>1,528</td>
<td>24%</td>
<td>22,007</td>
<td>34%</td>
<td>480</td>
<td>25%</td>
<td>13,481</td>
<td>36%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance Name</th>
<th>Size</th>
<th>Tot. CF</th>
<th>Tot. CC</th>
<th>Tot. BF</th>
<th>Tot. BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clique1</td>
<td>99</td>
<td>226</td>
<td>7,847</td>
<td>112</td>
<td>4,673</td>
</tr>
<tr>
<td>Clique2</td>
<td>78</td>
<td>181</td>
<td>431</td>
<td>7</td>
<td>212</td>
</tr>
<tr>
<td>Clique3</td>
<td>28</td>
<td>181</td>
<td>1,686</td>
<td>49</td>
<td>897</td>
</tr>
<tr>
<td>Clique4</td>
<td>18</td>
<td>246</td>
<td>3,130</td>
<td>39</td>
<td>1,427</td>
</tr>
<tr>
<td>Clique5</td>
<td>16</td>
<td>168</td>
<td>3,553</td>
<td>89</td>
<td>2,662</td>
</tr>
</tbody>
</table>
## Return on Investment Calculation

**Experience with SoftServe** [Kazman et al. 2015]

<table>
<thead>
<tr>
<th>DRSpace Leading File</th>
<th>DRSpace Size</th>
<th>Norm Size</th>
<th>Current Defects/Yr</th>
<th>Norm Defects</th>
<th>Current Changes/Yr</th>
<th>Norm Changes/Yr</th>
<th>Tot LOC Changed</th>
<th>Norm LOC Changed</th>
<th>Refactor Cost (PM)</th>
<th>Cost/Change*</th>
<th>Refactor Cost</th>
<th>Expected Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pear.java</td>
<td>139</td>
<td>119.33</td>
<td>166</td>
<td>142.5</td>
<td>1058</td>
<td>839.2</td>
<td>49,171</td>
<td>42,213</td>
<td>5.5</td>
<td>39</td>
<td>346</td>
<td>20,281</td>
</tr>
<tr>
<td>Apple.java</td>
<td>158</td>
<td>133.83</td>
<td>63</td>
<td>53.4</td>
<td>607</td>
<td>451.7</td>
<td>25,603</td>
<td>21,686</td>
<td>7</td>
<td>44</td>
<td>388</td>
<td>22,745</td>
</tr>
<tr>
<td>Bean.java</td>
<td>65</td>
<td>37.83</td>
<td>72</td>
<td>41.9</td>
<td>429</td>
<td>207.2</td>
<td>17,807</td>
<td>10,364</td>
<td>1.5</td>
<td>12</td>
<td>110</td>
<td>6,429</td>
</tr>
</tbody>
</table>

| DRSpace Total        | 290.99       | 237.8     | 1498               | 74,263       | 14                 | 96.0            | 843.871         | 49,455          |                  |              |              |                  |

| Savings              | 797          | 265       | 2332               | 135,453      |                  | 142             | 654             | 24,808          |                  |              |              |                  |

| Base defect rates    | 0.33         |          |                    |              |                  |                 |                 |                 |                  |              |              |                  |
| Base change rates    | 2.9          |          |                    |              |                  |                 |                 |                 |                  |              |              |                  |
| Base LOC/file        | 169.95       |          |                    |              |                  |                 |                 |                 |                  |              |              |                  |
| LOC/PM               | 600          |          |                    |              |                  |                 |                 |                 |                  |              |              |                  |

Result: ~300% ROI in the first year alone!
Analyzing 8 ABB Projects

Recent and Representative Experiences

✓ Using 3 complementary techniques:
  • Architecture-level maintainability metrics
  • Architecture flaw analysis and root detection
  • Cost and benefit analysis

✓ 8 projects developed at multiple locations (India, USA, and Switzerland) and differ in their ages, domains, and sizes.

✓ We reported the results back to each project and collected feedback

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Research Questions

Collecting feedbacks of practitioners

RQ1: does the tool suite help to close the gap between management and development? That is, does it help them to decide *if*, *when*, and *where* to refactor?

RQ2: does the tool suite help practitioners understand the maintainability of their systems *relative to other projects* internal to the company, and relative to a more broad-based benchmark suite?

RQ3: does the tool suite help developers pinpoint the *hotspots* of their systems, i.e., the groups of files with severe design flaws?
# Metrics Scores and Rankings

## Measure and Monitor

<table>
<thead>
<tr>
<th></th>
<th>DL</th>
<th>Percentile</th>
<th>PC</th>
<th>Percentile</th>
<th>#Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proj_EO</td>
<td>78%</td>
<td>85th</td>
<td>6%</td>
<td>85th</td>
<td>144</td>
</tr>
<tr>
<td>Proj_BM</td>
<td>77%</td>
<td>85th</td>
<td>2%</td>
<td>98th</td>
<td>371</td>
</tr>
<tr>
<td>Proj_CH</td>
<td>76%</td>
<td>81st</td>
<td>16%</td>
<td>54th</td>
<td>6,948</td>
</tr>
<tr>
<td>Proj_EP</td>
<td>72%</td>
<td>74th</td>
<td>7%</td>
<td>83th</td>
<td>1,541</td>
</tr>
<tr>
<td>Proj_SS</td>
<td>57%</td>
<td>49th</td>
<td>20%</td>
<td>45th</td>
<td>15,333</td>
</tr>
<tr>
<td>Proj.OP</td>
<td>57%</td>
<td>49th</td>
<td>21%</td>
<td>41th</td>
<td>7,754</td>
</tr>
<tr>
<td>Proj_CO</td>
<td>55%</td>
<td>43rd</td>
<td>17%</td>
<td>52th</td>
<td>491</td>
</tr>
<tr>
<td>Proj_EC</td>
<td>28%</td>
<td>5th</td>
<td>62%</td>
<td>2nd</td>
<td>4,125</td>
</tr>
</tbody>
</table>
Hotspots Detected

Pinpoint Hotspots

- Unstable Interface
- File Cliques
- Modularity Violations
- Most error-prone files

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
## Costs and Benefits of Architecture Debts

### Costs and benefits

<table>
<thead>
<tr>
<th>Root</th>
<th>Size (%)</th>
<th>Rt. CF (%)</th>
<th>Rt. CC (%)</th>
<th>Rt. BF (%)</th>
<th>Rt. BC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>root1</td>
<td>147 (10%)</td>
<td>1,109 (18%)</td>
<td>13,487 (21%)</td>
<td>414 (22%)</td>
<td>9,347 (25%)</td>
</tr>
<tr>
<td>root2</td>
<td>93 (6%)</td>
<td>1,050 (17%)</td>
<td>11,486 (18%)</td>
<td>452 (24%)</td>
<td>6,696 (18%)</td>
</tr>
<tr>
<td>root3</td>
<td>79 (5%)</td>
<td>601 (10%)</td>
<td>5,453 (9%)</td>
<td>183 (10%)</td>
<td>3,821 (10%)</td>
</tr>
<tr>
<td>root4</td>
<td>104 (7%)</td>
<td>486 (8%)</td>
<td>10,794 (17%)</td>
<td>166 (9%)</td>
<td>6,236 (17%)</td>
</tr>
</tbody>
</table>

### Debt Penalty

### Expected Benefits

<table>
<thead>
<tr>
<th>Root</th>
<th>Size</th>
<th>% Size</th>
<th>Change</th>
<th>Bug</th>
</tr>
</thead>
<tbody>
<tr>
<td>root1</td>
<td>147</td>
<td>10%</td>
<td>24%</td>
<td>29%</td>
</tr>
<tr>
<td>root2</td>
<td>222</td>
<td>14%</td>
<td>38%</td>
<td>52%</td>
</tr>
<tr>
<td>root3</td>
<td>263</td>
<td>17%</td>
<td>47%</td>
<td>57%</td>
</tr>
<tr>
<td>root4</td>
<td>364</td>
<td>24%</td>
<td>55%</td>
<td>65%</td>
</tr>
</tbody>
</table>

### Penalty of Architecture Roots

<table>
<thead>
<tr>
<th>Root</th>
<th>Extra CF</th>
<th>Extra CC</th>
<th>Extra BF</th>
<th>Extra BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>root1</td>
<td>612</td>
<td>8,450</td>
<td>263</td>
<td>6,418</td>
</tr>
<tr>
<td>root2</td>
<td>1,332</td>
<td>16,601</td>
<td>615</td>
<td>11,175</td>
</tr>
<tr>
<td>root3</td>
<td>1,687</td>
<td>19,570</td>
<td>724</td>
<td>13,552</td>
</tr>
<tr>
<td>root4</td>
<td>1,754</td>
<td>26,110</td>
<td>763</td>
<td>17,314</td>
</tr>
</tbody>
</table>

| Percentage | 28% | 41% | 40% | 47% |
Step 3: Collecting User feedback

Answering research questions

- Surveys
- Interviews
Post-Mortem

Survey with practitioners

Q1: What did the report reveal that you didn’t know about your software?
Q2: Are the metrics useful for reflecting the architecture of your software?
Q3: What did the architecture design flaws reveal about your software?
Q4: What actions have you planned as a result of the architecture design flaws report?
Q5: What did the architecture roots reveal about your software?
Q6: What actions do you plan to take to address architecture roots?
Results

RQ1: does the tool suite help to close the gap between management and development? That is, does it help them to decide if, when, and where to refactor?

5 Participants of all 8 projects verified that the information provided was useful in closing the understanding gap with management. They have begun the refactoring process.
**Results**

**RQ review**

RQ2: does the tool suite help practitioners understand the maintainability of their systems relative to other projects internal to the company, and relative to a more broad-based benchmark suite?

All participants said the report gave them quantifiable results with which to judge their project. The comparison with industrial benchmarks made it clear that maintenance difficulty caused by degrading architecture is common.
Results

RQ review

RQ3: does the tool suite help developers pinpoint the *hotspots* of their systems—that is, the groups of files with severe design flaws?

✔ Six of the eight projects planned to or already started refactoring to address the detected flaws. The project with the lowest DL score is undergoing a major rewrite.
It is possible to automatically and objectively assess and quantify architecture quality.

These results were enthusiastically received by the projects.

6 of the 8 projects are now embarking on major refactorings.

ABB is incorporating DV8 into its development process
Conclusion

RQ review

You can't manage it if you don't measure it. DV8 tool suite make measurement of design complexity, design flaws and design debt less labor-intensive and easily repeatable.

Incorporating these techniques into the build process ensures rapid feedback with supporting data.

This measurement, detection, and quantification practice leads to improved architectures.
Questions?

Office hour:
Thursday
1:30-2:30
Room E

www.archdia.net (Under Constraction)
[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
References


References


References


References


References

[Xiao et al. 2016] Lu Xiao, Yuanfang Cai, Rick Kazman, Ran Mo, Qiong Feng, "Identifying and quantifying architectural debt", ICSE 2016: 488-498
