Are you Afraid of Change?

Metrics for Software Evolvability

Arie van Deursen, Delft University of Technology
Joint work with Eric Bouwers and Joost Visser (SIG)
UC Irvine, March 15, 2013
@avandeuersen
View on Delft
Johannes Vermeer
1662
• 2 mile tunnel + station
• 4 train tracks
• Parking for 100 cars

• 1200 new apartments
• 24,000 m² park
• Parking for 4000 bikes

How would you manage this 15 year 650M Euro project?
The TU Delft
Software Engineering Research Group

Education
• Programming, software engineering
• MSc, BSc projects

Research
• Software testing
• Software architecture
• Repository mining
• Collaboration
• End-user programming
• Reactive programming
• Language workbenches
Collect detailed *technical findings* about software-intensive systems

Translate into *actionable information* for *high-level management*

Using methods from academic and self-funded *research*
Today’s Programme

• **Goal:** Can we measure software quality?

• **Approach:** How can we evaluate metrics?

• **Research:** Can we measure encapsulation?

• **Outlook:** What are the implications?
Context: *Software Risk Assessments*

- **System Supplier** provides for the system.
- **System Client** uses the system.
- **SRA Client** receives the system.
- **Sessions** participate.
- **Final Report** writes.
- **SRA Consultant** contributes to.
- **SRA Analyst** analyzes.

*ICSM 2009*
Early versus Late Evaluations

• Today’s topic: “Late” evaluations.
  – Actually implemented systems
  – In need of change

• Out of scope today:
  – “Early” evaluation (e.g., ATAM)
  – Software process (improvement)

L. Dobrica and E. Niemela. A survey on software architecture analysis methods. TSE 2002
ISO Software Quality Characteristics

- Functional Suitability
- Performance Efficiency
- Compatibility
- Usability
- Security
- Maintainability
- Reliability
- Portability

ISO 25010
Software Metric Pitfalls

Reflections on decade of metric usage

Pitfall 1: Treating the Metric

Metric values are symptoms:
It’s the root cause that should be addressed
Pitfall 2: Metric in a Bubble

To interpret a metric, a context is needed

Temporal / Trend

Peers / Norms
Pitfall 3: Metrics Galore

Not everything that can be measured needs to be measured.
Trade-offs in design require multiple metrics

In carefully crafted metrics suite, negative side effects of optimizing one metric are counter-balanced by other ones
Putting Metrics in Context

• Establish benchmark
  – Range of industrial systems with metric values
• Determine thresholds based on quantiles.
  – E.g.: 70%, 80%, 90% of systems
  – No normal distribution

Example: McCabe. 90% of systems have average unit complexity that is below 15.

Assessments 2003--2008

• ISO 9126 quality model
• ~50 assessments
• Code/module level metrics

• Architecture analysis always included
  – No architectural metrics used.

"Architectures allow or preclude nearly all of a system’s quality attributes."
-- Clements et al, 2005

Heitlager, Kuipers, Visser. A Practical Model for Measuring Maintainability. QUATIC 2007

Van Deursen, Kuipers. Source-Based Software Risk Assessments, ICSM 2003
2009: Re-thinking Architectural Analysis

Qualitative study of 40 risk assessments

Which architectural properties?

Outcome: Metrics refinement wanted

<table>
<thead>
<tr>
<th></th>
<th>High Level Design</th>
<th>Modularization</th>
<th>Separation of Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Functional Duplication</td>
<td>2</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>Layering</td>
<td>28</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Libraries / Frameworks</td>
<td>22</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Logic in Database</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Module Dependencies</td>
<td>7</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Module Functionality</td>
<td>4</td>
<td>32</td>
<td>13</td>
</tr>
<tr>
<td>Module Inconsistency</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Module Size</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Relation Documentation / Implementation</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Source Grouping</td>
<td>0</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>Technology Age</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Technology Usage</td>
<td>7</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Technology Combination</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Textual Duplication</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

Eric Bouwers, Joost Visser, Arie van Deursen: Criteria for the evaluation of implemented architectures. ICSM 2009
ISO 25010 Maintainability

“Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers”

Five sub-characteristics:

- Analyzability, Modifiability,
- Testability, Reusability
- Modularity
Modularity

ISO 25010 maintainability sub characteristic:

“Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components”
Information Hiding

Things that change at the same rate belong together.

Things that change quickly should be insulated from things that change slowly.

Measuring Encapsulation?

Can we find software architecture metrics that can serve as indicators for the success of encapsulation of an implemented software architecture?

Eric Bouwers, Arie van Deursen, and Joost Visser. 
Quantifying the Encapsulation of Implemented Software Architectures 
Metric Criteria in an Assessment Context

1. Potential to measure the level of encapsulation within a system
2. Is defined at (or can be lifted to) the system level
3. Is easy to compute and implement
4. Is as independent of technology as possible
5. Allows for root-cause analysis
6. Is not influenced by the volume of the system under evaluation
What is an Architecture?

Architectural Meta-Model

System

Component

Module

Unit

Dependency

Kind: Enum
Cardinality: Int

To

From

Architectural Element

Name: String
Size: Int

Architectural Meta-Model
Searching the Literature

- Identified over 40 candidate metrics
- Survey by Koziolek starting point
- 11 metrics meet criteria

Our own Proposal:
Dependency Profiles

Module types:
1. Internal
2. Inbound
3. Outbound
4. Transit

Eric Bouwers, Arie van Deursen, Joost Visser.
Dependency Profiles (2)

• Look at relative size of different module types

• Dependency profile is quadruple:
  <%internal, %inbound, %outbound, %transfer>

• <40, 30, 20, 10> versus <60, 20, 10, 0>

• Summary of componentization at the system level
Profiles in benchmark of ~100 systems
# Literature Study: Candidate Metrics

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbr.</th>
<th>Src.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of Cohesive Interactions</td>
<td>RCI</td>
<td>[Briand et al. 1993]</td>
</tr>
<tr>
<td>Cumulative Component Dependency</td>
<td>CCD</td>
<td>[Lakos 1996]</td>
</tr>
<tr>
<td>Average CCD</td>
<td>ACD</td>
<td>[Lakos 1996]</td>
</tr>
<tr>
<td>Normalized CCD</td>
<td>NCD</td>
<td>[Lakos 1996]</td>
</tr>
<tr>
<td>Cyclic Dependency Index</td>
<td>CDI</td>
<td>[Sarkar et al. 2007]</td>
</tr>
<tr>
<td>Inbound code</td>
<td>IBC</td>
<td>[Bouwers et al. 2011b]</td>
</tr>
<tr>
<td>Outbound code</td>
<td>OBC</td>
<td>[Bouwers et al. 2011b]</td>
</tr>
<tr>
<td>Internal code</td>
<td>IC</td>
<td>[Bouwers et al. 2011b]</td>
</tr>
<tr>
<td>Number of Binary Dependencies</td>
<td>NBD</td>
<td></td>
</tr>
<tr>
<td>Component Balance</td>
<td>CB</td>
<td>[Bouwers et al. 2011a]</td>
</tr>
<tr>
<td>Module Size Uniformity Index</td>
<td>MSUI</td>
<td>[Sarkar et al. 2007]</td>
</tr>
<tr>
<td>Number of components</td>
<td>NC</td>
<td></td>
</tr>
</tbody>
</table>
Metrics Evaluation

1. Quantitative approach
   - Which metric is the best predictor of good encapsulation?
   - Compare to change sets (repository mining)

2. Qualitative approach:
   - Is the selected metric useful in a late architecture evaluation context?
Commit in version repository results in change set
Change set I: modules \{ A, C, Z \}
Affects components C1 and C3
Change set II: modules \{ B, D, E \}
Affects components C1 only
Change set III: modules \{ Q, R, U \} 
Affects components C2 only
Change set IV: modules \{ S, T, Z \}
Affects components C2 and C3
Observation 1:  
Local Change-Sets are Good

• Combine change sets into *series*

• *The more local changes in a series, the better the encapsulation worked out.*
Observation 2: Metrics may change too

• A change may affect the value of the metrics.

• Cut large set of change sets into sequence of stable change-set series.
Change set I: modules \{ A, C, Z \}
Affects components C1 and C3
Change set I: modules \{ A, C, Z \}

*The Change Set may affect metric outcomes!!*
Solution: *Stable Period Identification*
Experimental Setup

• Identify 10 long running open source systems
• Determine metrics on monthly snapshots
• Determine stable periods per metric:
  – Metric value
  – Ratio of local change in this period
• Compute (Spearman) correlations [0, .30, .50, 1]
• Assess significance (p < 0.01)
• [ Assess project impact ]
• Interpret results
## Systems Under Study

<table>
<thead>
<tr>
<th>Name</th>
<th>Period</th>
<th>Size (KLOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Start</td>
<td>End</td>
</tr>
<tr>
<td>Ant</td>
<td>2000-02</td>
<td>2011-05</td>
</tr>
<tr>
<td>Argouml</td>
<td>2008-03</td>
<td>2011-07</td>
</tr>
<tr>
<td>Beehive</td>
<td>2004-08</td>
<td>2008-10</td>
</tr>
<tr>
<td>Crawljax</td>
<td>2010-01</td>
<td>2011-07</td>
</tr>
<tr>
<td>Findbugs</td>
<td>2003-04</td>
<td>2011-07</td>
</tr>
<tr>
<td>Jasperreports</td>
<td>2004-01</td>
<td>2011-08</td>
</tr>
<tr>
<td>Jedit</td>
<td>2001-10</td>
<td>2011-08</td>
</tr>
<tr>
<td>Jhotdraw</td>
<td>2001-03</td>
<td>2005-05</td>
</tr>
<tr>
<td>Lucene</td>
<td>2001-10</td>
<td>2011-08</td>
</tr>
<tr>
<td>Struts2</td>
<td>2006-06</td>
<td>2011-07</td>
</tr>
</tbody>
</table>
# Stable Periods

<table>
<thead>
<tr>
<th>Metric</th>
<th>periods</th>
<th>Min</th>
<th>Med.</th>
<th>Max</th>
<th>covered</th>
<th>Min</th>
<th>Med.</th>
<th>Max</th>
<th>total</th>
<th>&gt; 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCI</td>
<td>94</td>
<td>1</td>
<td>4.0</td>
<td>38</td>
<td>80.9 %</td>
<td>3</td>
<td>113.0</td>
<td>968</td>
<td>17760</td>
<td>93.6 %</td>
</tr>
<tr>
<td>CCD</td>
<td>71</td>
<td>1</td>
<td>6.0</td>
<td>40</td>
<td>85.9 %</td>
<td>3</td>
<td>222.0</td>
<td>1178</td>
<td>19011</td>
<td>97.2 %</td>
</tr>
<tr>
<td>ACD</td>
<td>111</td>
<td>1</td>
<td>3.0</td>
<td>38</td>
<td>75.6 %</td>
<td>1</td>
<td>92.0</td>
<td>954</td>
<td>16564</td>
<td>91.9 %</td>
</tr>
<tr>
<td>NCD</td>
<td>74</td>
<td>1</td>
<td>4.5</td>
<td>40</td>
<td>83.6 %</td>
<td>3</td>
<td>192.5</td>
<td>1174</td>
<td>17922</td>
<td>95.9 %</td>
</tr>
<tr>
<td>CDI</td>
<td>65</td>
<td>1</td>
<td>6.0</td>
<td>50</td>
<td>88.3 %</td>
<td>1</td>
<td>224.0</td>
<td>2334</td>
<td>20526</td>
<td>95.4 %</td>
</tr>
<tr>
<td>IBC</td>
<td>122</td>
<td>1</td>
<td>3.0</td>
<td>35</td>
<td>68.1 %</td>
<td>3</td>
<td>67.5</td>
<td>715</td>
<td>13811</td>
<td>95.9 %</td>
</tr>
<tr>
<td>OBC</td>
<td>111</td>
<td>1</td>
<td>3.0</td>
<td>42</td>
<td>71.8 %</td>
<td>3</td>
<td>68.0</td>
<td>1337</td>
<td>15346</td>
<td>94.6 %</td>
</tr>
<tr>
<td>IC</td>
<td>119</td>
<td>1</td>
<td>2.0</td>
<td>41</td>
<td>71.2 %</td>
<td>2</td>
<td>50.0</td>
<td>1257</td>
<td>14759</td>
<td>91.6 %</td>
</tr>
<tr>
<td>NBD</td>
<td>108</td>
<td>1</td>
<td>3.0</td>
<td>38</td>
<td>75.8 %</td>
<td>3</td>
<td>88.5</td>
<td>846</td>
<td>15436</td>
<td>94.4 %</td>
</tr>
<tr>
<td>CB</td>
<td>82</td>
<td>1</td>
<td>3.0</td>
<td>77</td>
<td>80.6 %</td>
<td>3</td>
<td>76.5</td>
<td>5147</td>
<td>19345</td>
<td>91.5 %</td>
</tr>
<tr>
<td>MSUI</td>
<td>99</td>
<td>1</td>
<td>3.0</td>
<td>35</td>
<td>77.1 %</td>
<td>1</td>
<td>91.0</td>
<td>1176</td>
<td>18028</td>
<td>93.9 %</td>
</tr>
<tr>
<td>NC</td>
<td>59</td>
<td>1</td>
<td>6.0</td>
<td>53</td>
<td>90.8 %</td>
<td>7</td>
<td>262.0</td>
<td>1805</td>
<td>21428</td>
<td>96.6 %</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Metric</th>
<th>Correlation</th>
<th>Corrected p-value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCI</td>
<td>0.16</td>
<td>11.3</td>
<td>0.94</td>
</tr>
<tr>
<td>CCD</td>
<td>-0.27</td>
<td>0.13</td>
<td>0.01</td>
</tr>
<tr>
<td>ACD</td>
<td>-0.26</td>
<td>0.04</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>NCD</td>
<td>-0.19</td>
<td>0.59</td>
<td>0.05</td>
</tr>
<tr>
<td>CDI</td>
<td>0.32</td>
<td>11.94</td>
<td>1.00</td>
</tr>
<tr>
<td>IBC</td>
<td>-0.30</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>OBC</td>
<td>-0.31</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>IC</td>
<td>0.47</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>NBD</td>
<td>-0.22</td>
<td>0.14</td>
<td>0.01</td>
</tr>
<tr>
<td>CB</td>
<td>0.29</td>
<td>0.05</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>MSUI</td>
<td>-0.08</td>
<td>2.42</td>
<td>0.20</td>
</tr>
<tr>
<td>NC</td>
<td>-0.26</td>
<td>0.27</td>
<td>0.02</td>
</tr>
</tbody>
</table>
Best Indicator for Encapsulation: *Percentage of Internal Code*

Module types:
1. Internal
2. Inbound
3. Outbound
4. Transit
Threats to Validity

Construct validity
- Encapsulation == local change?
- Commit == coherent?
- Commit size?
- Architectural model?

Reliability
- Open source systems
- All data available

Internal validity
- Stable periods: Length, nr, volume
- Monthly snapshots
- Project factors

External validity
- Open source, Java
- IC behaves same on other technologies
Shifting paradigms

• Statistical hypothesis testing:

  Percentage of internal change is valid indicator for encapsulation

• But is it of any use?
• Can people work with?
• *Shift to pragmatic knowledge paradigm*
Software Risk Assessments

- System Supplier
  - Provides
  - Participates

- System Client
  - Uses
  - Receives

- SRA Client
  - Participates

- Sessions
  - Participates

- Final Report
  - Writes
  - Contributes to
  - Analyzes

- SRA Consultant
  - Contributes to

- SRA Analyst
Experimental Design

Goal:
• Understand the usefulness of dependency profiles
• From the point of view of external quality assessors
• In the context of external assessments of implemented architectures

Embedding

• January 2012: New metrics in SIG models
  – 50 risk assessments during 6 months
  – Monitors for over 500 systems
  – “Component Independence”

• System characteristics:
  – C#, Java, ASP, SQL, Cobol, Tandem, ...
  – 1000s to several millions of lines of code
  – Banking, government, insurance, logistics, ...
Data Gathering: Observations

- February-August 2012
- Observer collects stories of actual usage
- Written down in short memos.

- 17 different consultants involved
- 49 memos collected.
- 11 different customers and suppliers
Data Gathering: Interviews

• 30 minute interviews with 11 assessors
• Open discussion:
  – “How do you use the new component independence metric”?
  – Findings in 1 page summaries
• Scale 1-5 answer:
  – How useful do you find the metric?
  – Does it make your job easier?
Resulting Coding System

Decision making
- Targeted improvements
- Start of discussions
- Communication device

Intuition
- Small systems
- Older technologies
- Influence of nr of components

Component Definition
- File-system versus mental model
- System Scope
- Subjectivity

Application
- Model introduction
- Definition of actions
- Effect prediction
- Steering
- Context
- Effort prediction

Implementation
- Component Balance
- Component Independence

Michaela Greiler, Arie van Deursen, Margaret-Anne D. Storey: Test confessions: A study of testing practices for plug-in systems. ICSE 2012: 244-253
Motivating Refactorings

• Two substantial refactorings mentioned:
  1. Code with semi-deprecated part
  2. Code with wrong top-level decomposition.

• Developers were aware of need for refactoring. With metrics, they could:
  – Explain need to stakeholders
  – Explain progress made to stakeholders
What is a Component?

Different “architectures” exist:
1. In the minds of the developers
2. As-is on the file system
3. As used to compute the metrics

- Easiest if 1=2=3
- Regard as different *views*
- Different view per developer?
Concerns

• Do size or age affect information hiding?

• No components in Pascal, Cobol, ...
  – Naming conventions, folders, mental, ...
  – Pick best fitting mental view

• # top level components independent of size
  – Metric distribution also not size dependent

Not Easy-to-Use. But Useful.
Dependency Profiles: Conclusions

Lessons Learned
Need for
• Strict component definition guidelines
• Body of knowledge
  – Value patterns
  – With recommendations
  – Effort estimation
• Improved dependency resolution

Threats to Validity
• High realism
• Data confidential
• Range of different systems and technologies

Wanted: replication in open source (Java / Sonar) context
A Summary in Seven Slides
Accountability and Explainability

- Accountability in software architecture?
  - Not very popular

- Stakeholders are entitled to an explanation

- Metrics are a necessary ingredient
Metrics Need Context

Temporal / Trend

Peers / Norms
Metrics Research Needs Datasets

Two recent Delft data sets:

- **Github Torrent:**
  - Years of github history in relational database.
  - Georgios Gousios

- **Maven Dependency Dataset**
  - Versioned call-level dependencies in full Maven Central.
  - Steven Raemaekers
Metrics Research needs Qualitative Methods

• Evaluate based upon the *possibilities of action*

• Calls for rigorous studies capturing reality in *rich narratives*

• Case studies, interviews, surveys, ethnography, grounded theory, ...
Encapsulation Can be Measured

Module types:
1. Internal
2. Inbound
3. Outbound
4. Transit

And doing so, leads to meaningful discussions.
Should we be Afraid of Change?

Metrics for Software Evolvability

Arie van Deursen, Delft University of Technology
Joint work with Eric Bouwers & Joost Visser (SIG)
@avandeursen