Software: Good, Bad, or Just a Matter of Taste?

Arie van Deursen
Delft University of Technology, The Netherlands
SBES 2012, Natal, Brazil, 27 September 2012
Acknowledgements

• Marcio Delamaro & CBSoft organizers

• Joint work with Eric Bouwers & Joost Visser Software Improvement Group

• Co-workers TU Delft Software Engineering Research Group
Collect detailed *technical findings* about software-intensive systems.

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

Using methods from academic and self-funded *research*.

Focus on Software Architecture

The organizational structure of a software system including components, connectors, constraints, and rationale

-- Kogut & Clemens, 1994

“Architectures allow or preclude nearly all of a system’s quality attributes.”

-- Clements et al, 2005
Early versus Late Evaluations

- **Early:** Making design decisions
  - Design space exploration
  - What-if analysis

- **Late:** Analyzing *as-implemented* architecture
  - Drift & erosion
  - Actual suitability

---

van Deursen, Hofmeister, Koschke, Moonen, Riva: 
Symphony: View-Driven Software Architecture Reconstruction. WICSA 2004

L. Dobrica and E. Niemela. A survey on software architecture analysis methods. 
The Software Risk Assessment Process
Lord Kelvin

Meten = Weten

Medicação = Conhecimento

To measure = To know

“when you cannot measure it, (...) your knowledge is of a meagre and unsatisfactory kind”
Country Size
Brazil - NL
Population
Brazil - NL
Gross Domestic Product
GDP Per Capita
Brazil – NL
Football Wins
4-3
IT Labor Force?
IT Profitability?
IT Quality??
Software metrics for project management?

Beware!

E. Bouwers, J. Visser, and A. van Deursen.

Getting what you Measure

Communications of the ACM, May 2012
Pittfall 1:

Metric in a bubble:
No context, no goal

Getting What You Measure

ARE SOFTWARE METRICS helpful tools or a waste of time? For every developer who treasures these mathematical abstractions of software systems, there is a developer who thinks software metrics are invented just to keep project managers busy. Software metrics can be very powerful tools that help achieve your goals but it is important to use them correctly, as they also have the power to demotivate project teams and steer development in the wrong direction.

For the past 11 years, the Software Improvement Group has advised hundreds of organizations concerning software development and risk management on their strategy. We have used software metrics in investigations in which we have studied the behavior of a system. Additionally, we have tracked the ongoing development of 400 systems. While we have learned some pitfalls of using metrics in project management, the article addresses them.

Software Metrics Steer People
“You get what you measure.” This phrase definitely applies to software project teams. No matter what you define as a metric, as soon as it is used to evaluate a team, the value of the metric moves toward the desired value. Thus, to reach a particular goal, you can continuously measure properties of the desired goal and plot these measurements in a place visible to the team. Ideally, the desired goal is plotted alongside the current measurement to indicate the distance to the goal.

Imagine a project in which the runtime performance of a particular use...
Pittfall 2:

Metrics galore: Measuring everything

Getting What You Measure

Are software metrics helpful tools or a waste of time? For every developer who treasures these mathematical abstractions of software systems, there is a developer who thinks software metrics are invented just to keep project managers busy. Software metrics can be very powerful tools that help achieve your goals but it is important to use them correctly, as they also have the power to demotivate project teams and steer development in the wrong direction.

For the past 11 years, the Software Improvement Group has advised hundreds of organizations concerning software development and risk management on the strategic use of software metrics. We have used software metrics to improve performance and investigate why projects go over budget or exceed their deadlines. Additionally, we use software metrics to track the ongoing development of 400 systems. While we have learned some pitfalls, this article addresses the benefits of software metrics in project management.

Software Metrics Steer People

“you get what you measure.” This phrase definitely applies to software project teams. No matter what you define as a metric, as soon as it is used to evaluate a team, the value of the metric moves toward the desired value. Thus, to reach a particular goal, you can continuously measure properties of the desired goal and plot these measurements in a place visible to the team. Ideally, the desired goal is plotted alongside the current measurement to indicate the distance to the goal.

Imagine a project in which the runtime performance of a particular use
Pittfall 3:

One-track metric: Narrow-minded
Pittfall 4:

Treating the metric: Fighting symptoms

Software Metrics Steer People
“you get what you measure.” This phrase definitely applies to software project teams. No matter what you define as a metric, as soon as it is used to evaluate a team, the value of the metric moves toward the desired value. Thus, to reach a particular goal, you can continuously measure properties of the desired goal and plot these measurements in a place visible to the team. Ideally, the desired goal is plotted alongside the current measurement to indicate the distance to the goal.
ISO25010 -- The Many Faces of Software Quality
## What to measure?

<table>
<thead>
<tr>
<th>Maintainability sub-characteristic</th>
<th>Volume</th>
<th>Duplication</th>
<th>Unit size</th>
<th>Unit complexity</th>
<th>Unit interfacing</th>
<th>Module coupling</th>
<th>Component balance</th>
<th>Component independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzability</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td>❌</td>
<td>✗</td>
</tr>
<tr>
<td>Modifiability</td>
<td></td>
<td>✗</td>
<td>✗</td>
<td>❌</td>
<td></td>
<td>❌</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Testability</td>
<td>✗</td>
<td></td>
<td></td>
<td>❌</td>
<td>❌</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Modularity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Reusability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
</tbody>
</table>

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

Metric Value: Good or Bad?

• **Benchmark** of many industrial systems
  – All metrics of interest collected

• Determine **thresholds** based on percentage of systems with given value.
  – E.g.: 90% of systems have McCabe <= 15
  – McCabe > 15 suggests *high risk*.

---

Tiago L. Alves, Christiaan Ypma, Joost Visser.
Deriving metric thresholds from benchmark data. *ICSM* 2010.
2009: Re-assessing Architectural Properties

Study of 40 risk assessments

Rethinking architectural properties

Outcome: Metrics refinement wanted

<table>
<thead>
<tr>
<th>Metric</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
Information Hiding

Things that change at the same rate belong together

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

Encapsulation Metrics?

Which software architecture metrics 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 
Approach

• Identification of candidate metrics

• Quantitative approach (repository mining):
  – *Which metric is the best predictor of good encapsulation?*

• Qualitative approach:
  – *Is the selected metric useful in a late architecture evaluation context?*
## 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>
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-Set Series 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*
Approach

• 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>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Start</td>
<td>End</td>
<td>Start</td>
<td>End</td>
</tr>
<tr>
<td>Ant</td>
<td>2000-02</td>
<td>2011-05</td>
<td>3</td>
<td>97</td>
</tr>
<tr>
<td>Argouml</td>
<td>2008-03</td>
<td>2011-07</td>
<td>113</td>
<td>108</td>
</tr>
<tr>
<td>Beehive</td>
<td>2004-08</td>
<td>2008-10</td>
<td>45</td>
<td>86</td>
</tr>
<tr>
<td>Crawljax</td>
<td>2010-01</td>
<td>2011-07</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Findbugs</td>
<td>2003-04</td>
<td>2011-07</td>
<td>7</td>
<td>97</td>
</tr>
<tr>
<td>Jasperreports</td>
<td>2004-01</td>
<td>2011-08</td>
<td>28</td>
<td>171</td>
</tr>
<tr>
<td>Jedit</td>
<td>2001-10</td>
<td>2011-08</td>
<td>35</td>
<td>79</td>
</tr>
<tr>
<td>Jhotdraw</td>
<td>2001-03</td>
<td>2005-05</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>Lucene</td>
<td>2001-10</td>
<td>2011-08</td>
<td>6</td>
<td>67</td>
</tr>
<tr>
<td>Struts2</td>
<td>2006-06</td>
<td>2011-07</td>
<td>25</td>
<td>22</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

Eric Bouwers, Arie van Deursen, Joost Visser.
Threats to Validity

**Construct validity**
- Encapsulation == local change?
- Commit == coherent?
- Commits too small?
- Commits too large?
- Architectural (meta) model appropriate?

**External**
- Open source, Java
- IC behaves same on other technologies

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

**Reliability**
- Open source systems
- All data available
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 to
- **System Client** uses
- **Final Report** receives
- **Sessions** participate
- **SRA Client** participates
- **SRA Consultant** writes and contributes to
- **SRA Analyst** analyzes

**Diagram Notes:**
- Sessions participate in the process.
- Final Report is a key output.
- The system is analyzed by the SRA Analyst.
Using “Percentage of Internal Change” For Evaluation Purposes

Setting

• SIG has included %IC in its revised maintainability model

• Used in risk assessments and monitors of many industrial systems

• Frequent discussion between clients, consultants, suppliers, and analysts.
Evaluation Approach

• SIG consultants and analysts report any IC-related experience to investigator

• Collect “stories” of actual use in (anonymized) memos.
  – 6 months, 50 memos so far

• Analyze using grounded theory

Michaela Greiler, Arie van Deursen, Margaret-Anne D. Storey: Test confessions: A study of testing practices for plug-in systems. ICSE 2012: 244-25
Preliminary Findings (1)

[ Still collecting, analyzing, and digesting ]

• Metrics added *require* componentization / architectural view.
  – Documenting it is a useful process
  – Low %IC values trigger discussion

• Important to see metric in full suite (avoid “one-track method”)
Preliminary Findings (2)

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

• *These should be the same!*
Preliminary Findings (3)

• Is low %IC fixable?
• Requires (substantial) refactoring:
  – Smaller components and interfaces,
  – Fewer dependencies.
• Consultant’s wish list: “what-if analysis”:
  – See effects of changing architecture
• Major refactoring for one of the systems monitored planned based on %IC insights
Encapsulation Can be Measured!

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

And doing so, leads to meaningful discussions.
Software: Good, Bad, or Just a Matter of Taste?

Ultimately a Matter of Engineering Tradeoffs

Yet there is a difference between right and wrong
Software Metrics Matter

• A key tool for informed decision making

• Benchmarking, monitoring, and interpretation are key

• Pitfalls are plentiful, in research as in practice
(Open Source) Repositories: A Research Gold Mine

• Today: Metrics validation.

• GitHub! GitHub! GitHub!

• Relentless identification of threats to validity essential: “Ameaças à validade”!
Paradigm: *Pragmatism*

- Evaluate based upon *the possibilities of action*
- Calls for rigorous qualitative studies capturing reality in *rich narratives*
- Rich palette of empirical methods needed
The Final: July 13, 2014
The Final:
July 13, 2014

Brazil vs
Netherlands