Benchmark-based Software Product Quality

Tiago L. Alves
Background

About me
- Degree + MSc Informatics and System Engineering, University of Minho, Braga, PT
- 2006: Young Graduate Trainee at ESOC, Darmstadt, Germany
- Currently: PhD Researcher at University of Minho hosted at the Software Improvement Group, Amsterdam, Netherlands (finishing thesis)

Research interests
- Source code analysis techniques
- Software metrics and quality models to assess quality
- Industrial applications
Who are we?

- Highly specialized research company for quality of software, founded in 2000 as a spin-off of the Centre for Mathematics and Information Technology
- Independent and therefore able to give objective advice
- Decorated with the Innovator Award 2007 and ICT Regie Award 2008

What do we do?

- Fact-based consultancy supported by our automated toolset for source code analysis
- Assessment across technologies by use of technology-independent methods

Our mission: We give you control over your software.
Services

Software Risk Assessment
• In-depth investigation of software quality and associated business risks
• Answers to specific research questions

Software Monitoring
• Continuous measurement, feedback, and development consultancy
• Guard quality from start to finish

Software Product Certification
• Five levels of technical quality (maintainability)
• Evaluation by SIG, certification by TÜV Informationstechnik
Who is using our services?

Financial and Insurance companies
- ABN AMRO
- PGGM
- ING
- Rabobank
- Friesland Bank
- LeasePlan
- Interpolis
- Bank Mendes Gans
- globalcollect
- Allianz

Government
- Rijksoverheid
- Kadaster
- Zorg en Zekerheid
- Delta Lloyd
- Eurobank EFG
- ZwitserLeven
- KAS BANK
- SNS Bank
- Volvo OtoFinans

Logistical
- DHL
- Getronics PinkRocade
- KLM
- Centric
- Capgemini
- TNT
- Exact Software
- Euromax Terminal
- Norfolk Line
- IBM
- NXP
- ProRail
- Port of Rotterdam
- Logica

IT
- Essent
- ENECO
- SWISSLEX
- KPMG
- Gasunie
- THCO
- Alcatel-Lucent
- Electrabel

Other
Bermuda triangle of software quality assurance

Process (organizational)
- ISO 9001
- SPICE (ISO 15504)
- CMMI

Product
- ISO 9126
- ISO 25010

People (individual)
- OCP (Oracle)
- MCP (Microsoft)

Project (individual)
- Prince2
- PMBOK / PMI
- RUP (IBM)
- Scrum
SIG Quality Model for Maintainability (operationalization of the ISO 9126)

ISO/IEC 9126
- characteristics
  - Functionality
  - Reliability
  - Maintainability
  - Usability
  - Efficiency
  - Portability

- sub-characteristics
  - Analysability
  - Changeability
  - Stability
  - Testability

SIG Quality Model
- product properties
  - Volume
  - Duplication
  - Unit complexity
  - Unit size
  - Unit interfacing
  - Testing

- source code measurements
  - Functional + unit testing / coverage

- metrics
Benchmarking metrics to ratings

Risk profiles thresholds derivation

Risk profile

Cumulative rating thresholds

Rating | Moderate | High | Very-high
--- | --- | --- | ---
★ ★ ★ ★ | 17.9% | 9.9% | 3.3%
★ ★ ★ ★ ★ | 23.4% | 16.9% | 6.7%
★ ★ ★ ★ ★ ★ | 31.3% | 23.8% | 10.6%
★ ★ ★ ★ ★ ★ ★ | 39.1% | 29.8% | 16.7%
★ ★ ★ ★ ★ ★ ★ ★ | - | - | -

Rating: ★ ★ ★ ★ ★ ★ ★ (2.68)
Part I

Derivation of risk thresholds
How to derive thresholds?
Life sciences vs. software sciences

Cholesterol levels

Typical values $\mu$

Not so good $\mu \pm \delta$

Malnutrition - anxiety, depression, suicide
Heart attack

Complexity

Tiago L. Alves, Invited PEM Talk, SEN1, CWI, Amsterdam 2011-06-22 © Software Improvement Group
Derivation of risk thresholds: requirements

Requirements

1. Respect the statistical properties of the metric (scale and distribution)
2. Based on data analysis from a representative set of systems (benchmark)
3. Repeatable, transparent, and of straightforward execution.
4. Enable traceability of results
## Experimental benchmark

<table>
<thead>
<tr>
<th>Technology</th>
<th>License</th>
<th>n</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>Proprietary OSS</td>
<td>60</td>
<td>8,435K</td>
</tr>
<tr>
<td></td>
<td>OSS</td>
<td>22</td>
<td>2,756K</td>
</tr>
<tr>
<td>C#</td>
<td>Proprietary OSS</td>
<td>17</td>
<td>794K</td>
</tr>
<tr>
<td></td>
<td>OSS</td>
<td>1</td>
<td>10K</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>100</td>
<td>11,996K</td>
</tr>
</tbody>
</table>
Derivation of risk thresholds: methodology

1. metrics extraction
   System → (Entity → Metric × Weight)

2. weight ratio calculation
   System → (Entity → Metric × WeightRatio)

3. entity aggregation
   System → (Metric → WeightRatio)

4. system aggregation
   Metric → WeightRatio

5. weight ratio aggregation
   WeightRatio → Metric

6. thresholds derivation
   Metric, Metric, Metric

Legend
- map relation (one-to-many relationship)
- product (pair of columns or elements)

System
Repsents individual systems (e.g. Vuze)

Entity
Repsents a measurable entity (e.g java method)

Metric
Repsents a metric value (e.g. McCabe of 5)

Weight
Repsents the weight value (e.g. LOC of 10)

WeightRatio
Repsents the weight percentage inside of the system (e.g. entity LOC divided by system LOC)
Derivation of risk thresholds: background

Histogram vs. Quantile plots
Derivation of risk thresholds: Weight by size
Derivation of risk thresholds
Summarizing a metric distribution
Derivation of risk thresholds
Relative weighting
Derivation of risk thresholds: values
Risk Thresholds for the SIG Quality Model

<table>
<thead>
<tr>
<th>Metric / Quantiles</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit complexity</td>
<td>6</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>Unit size</td>
<td>30</td>
<td>44</td>
<td>74</td>
</tr>
<tr>
<td>Module inward coupling</td>
<td>10</td>
<td>22</td>
<td>56</td>
</tr>
<tr>
<td>Module interface size</td>
<td>29</td>
<td>42</td>
<td>73</td>
</tr>
<tr>
<td>Metric / Quantiles</td>
<td>80%</td>
<td>90%</td>
<td>95%</td>
</tr>
<tr>
<td>Unit interfacing</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Analysis of risk thresholds

- Risk Category: Low, Medium, High, Very High
- Percentage: 80%, 70%, 10%

[Diagram showing box plots for different risk categories with percentage thresholds.]
Derivation of risk thresholds
Concluding remarks

Novel methodology to derive metric thresholds
• Solid methodology based on benchmark (depart from expert opinion)
• Agrees with expert opinion (thresholds are sensible)

Plans for the future
• Validate with external characteristics

“The” Lessons
• We can attribute meaning to thresholds
• Benchmarks are of extreme importance
Benchmarking metrics to ratings

Risk profiles thresholds derivation

Risk thresholds
- Low risk: [0,6]
- Moderate risk: [6,8]
- High risk: [8,14]
- Very-high risk: [14, ∞]

Risk profile

Low risk: 74.2%
Moderate risk: 7.1%
High risk: 8.8%
Very-high risk: 9.9%

Cumulative rating thresholds

<table>
<thead>
<tr>
<th>Rating</th>
<th>Moderate</th>
<th>High</th>
<th>Very-high</th>
</tr>
</thead>
<tbody>
<tr>
<td>★★★★☆</td>
<td>17.9%</td>
<td>9.9%</td>
<td>3.3%</td>
</tr>
<tr>
<td>★★★★☆</td>
<td>23.4%</td>
<td>16.9%</td>
<td>6.7%</td>
</tr>
<tr>
<td>★★★☆☆</td>
<td>31.3%</td>
<td>23.8%</td>
<td>10.6%</td>
</tr>
<tr>
<td>★★★☆☆</td>
<td>39.1%</td>
<td>29.8%</td>
<td>16.7%</td>
</tr>
<tr>
<td>★★★★★</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Rating

★☆☆☆☆☆ (2.68)
Part II

Calibration of rating thresholds
Benchmark partitioning
The problem

![Graph showing McCabe values vs quantiles (% of LOC)]

Tiago L. Alves, Invited PEM Talk, SEN1, CWI, Amsterdam 2011-06-22 © Software Improvement Group
Benchmark partitioning
The problem #2

Unit complexity risk profiles for 20 random systems

- Low risk
- Moderate risk
- High risk
- Very-high risk
Benchmark partitioning
Order by Very-high risk category

Unit complexity risk profiles for 20 random systems

Other risk categories do not follow same order
Ratings calibration algorithm

Require: riskprofiles : (Moderate x High x VeryHigh)*, partition\textsuperscript{N-1}

1: thresholds ← ()
2: ordered[Moderate] ← order(riskprofiles.Moderate)
3: ordered[High] ← order(riskprofiles.High)
4: ordered[VeryHigh] ← order(riskprofiles.VeryHigh)

5: for rating = 1 to (N - 1) do
6:   i ← 0
7:   repeat
8:     i ← i + 1
9:     thresholds[rating][Moderate] ← ordered[Moderate][i]
10:    thresholds[rating][High] ← ordered[High][i]
11:   thresholds[rating][VeryHigh] ← ordered[VeryHigh][i]
12: until distribution(riskprofiles, thresholds[rating]) ≤ partition[rating] and i < length(riskprofiles)
13: index ← i
14: for all risk in (Moderate, High, VeryHigh) do
15:   i ← index
16:   done ← False
17:   while i > 0 and not done do
18:     thresholds.old ← thresholds
19:     i ← i - 1
20:     thresholds[risk][rating] ← ordered[risk][i]
21: if distribution(riskprofiles, thresholds[rating]) < partition[rating] then
22:   thresholds ← thresholds.old
23:   done ← True
24: end if
25: end while
26: end for
27: end for
28: return thresholds
Unit complexity risk profiles for 20 random systems
Calibrated rating thresholds
Unit complexity (McCabe)

Cumulative rating thresholds calibrated from 100 system benchmark

<table>
<thead>
<tr>
<th>Rating</th>
<th>Low</th>
<th>Moderate</th>
<th>High</th>
<th>Very-High</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[0,6]</td>
<td>[6,8]</td>
<td>[8,14]</td>
<td>&gt; 14</td>
</tr>
<tr>
<td>★★★★★</td>
<td>-</td>
<td>17.9%</td>
<td>9.9%</td>
<td>3.3%</td>
</tr>
<tr>
<td>★★★★</td>
<td>-</td>
<td>23.4%</td>
<td>16.9%</td>
<td>6.7%</td>
</tr>
<tr>
<td>★★★</td>
<td>-</td>
<td>31.3%</td>
<td>23.8%</td>
<td>10.6%</td>
</tr>
<tr>
<td>★★</td>
<td>-</td>
<td>39.1%</td>
<td>29.8%</td>
<td>16.7%</td>
</tr>
<tr>
<td>★</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Calibration of rating thresholds
Concluding remarks

Novel methodology to aggregate metric to ratings
• Support of N-point scale (used 5-point star rating)
• Solid methodology based on benchmark
• Enables traceability using thresholds and risk profiles

Plans for the future
• Validate with external characteristics
• Step for building quality models

“The” Lessons
• We can attribute meaning to ratings
• Ratings can be used to rank and evaluate software systems
Benchmarking metrics to ratings

**Risk profiles thresholds derivation**

Risk thresholds:
- Low risk: [0,6]
- Moderate risk: [6,8]
- High risk: [8,14]
- Very-high risk: ]14, ∞[

**Risk profile**

- Low risk: 74.2%
- Moderate risk: 7.1%
- High risk: 8.8%
- Very-high risk: 9.9%

**Cumulative rating thresholds**

<table>
<thead>
<tr>
<th>Rating</th>
<th>Moderate</th>
<th>High</th>
<th>Very-high</th>
</tr>
</thead>
<tbody>
<tr>
<td>★★★★★</td>
<td>17.9%</td>
<td>9.9%</td>
<td>3.3%</td>
</tr>
<tr>
<td>★★★★★</td>
<td>23.4%</td>
<td>16.9%</td>
<td>6.7%</td>
</tr>
<tr>
<td>★★★★☆☆</td>
<td>31.3%</td>
<td>23.8%</td>
<td>10.6%</td>
</tr>
<tr>
<td>★★★☆☆☆</td>
<td>39.1%</td>
<td>29.8%</td>
<td>16.7%</td>
</tr>
<tr>
<td>★★☆☆☆☆</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Rating** ⭐⭐⭐⭐⭐ (2.68)

1st-level aggregation

2nd level aggregation
Part III
Using ratings for quality evaluation
Simulators for Space Domain

**EuroSim**
- Commercial simulator
- Consortium: Dutch Space, NLR, TASK24
- Supports hardware-in-the-loop and man-in-the-loop
- Hard real-time
- Has non-space applications (e.g. aircraft simulation)

**ESA/ESOC SimSat**
- ESA owned (free)
- Mainly used for spacecraft telecommunication simulation
- Real-time
Technical Analysis on EuroSim
Research Motivation

Comparison of the technical quality between systems of the same domain

- Dutch Space EuroSim mk4.1.5
- ESA SimSat v4.0.1 issue 2

Research question

- How does the technical quality of EuroSim compare to SimSat?
  (how to use ratings to evaluate and compare quality)
Scope of the analysis

**Analyzed programming languages**
- C/C++
- Java

**Considerations**
- Production and test code analyzed separately
- Excluded documentation code and examples
- Excluded generated code (Icon files generated by XMP)
- Excluded open-source libraries
- Excluded drivers supplied by hardware suppliers
  - device drivers developed by EuroSim Consortium were included in the analysis
- Excluded code not in use: PerflGS, LibCadesee
Unit Complexity definition

- Cyclomatic Complexity = Number of decision points per unit (method/function)

- Widely accepted measurement for code complexity
- Should be as small as possible

McCabe: 4
Unit Complexity comparison

Discussion

• Both EuroSim and SimSat rank two stars
• Very-high complexity was found: 4.2% for EuroSim and 2.6% SimSat
• For EuroSim, very-high complexity is localized in 16 methods.
• Taking the last three risk categories, EuroSim has almost twice the risk of SimSat
• For EuroSim, comparing versions reveals slow decrease of the quality
## ISO 9126 Maintainability Comparison between EuroSim and SimSat

<table>
<thead>
<tr>
<th></th>
<th>Volume</th>
<th>Duplication</th>
<th>Unit size</th>
<th>Complexity</th>
<th>Unit interfacing</th>
<th>Test quality</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dutch Space</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EuroSim</strong></td>
<td>★★★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★★</td>
</tr>
<tr>
<td>Analysability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changeability</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stability</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>
</tr>
<tr>
<td><strong>ESA SimSat</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score</td>
<td>★★★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★★</td>
</tr>
<tr>
<td>Analysability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changeability</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stability</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>
</tr>
</tbody>
</table>

**Dutch Space EuroSim Maintainability:** ★★★

**ESA SimSat Maintainability:** ★★
Conclusion

Contributions
• Generic methodology to use metrics for quality evaluation
• Demonstration of the methodology using space-domain software

Benchmark-based approach benefits
• Meaningful (relation with industrial systems)
• Operational (thresholds can be obtained automatically)

Future work
• Use the methodology to validate metrics external characteristics
• Finalize PhD thesis!!!
Shameless Commercial Alert
• Internships (currently hosting 3 PhD + 4 MSc students)
• Jobs
• Research cooperation

• Very interesting clients
• You get to see lots and lots of source code from all over the world
• Software Engineering or Consultancy skills (or both)

• See www.sig.eu
More info? Feel free to contact...

Tiago L. Alves

E: t.alves@sig.eu
W: www.sig.eu
T: +31 20 3140950