Software Product Quality and its Effects

Joost Visser

November 2011
Software Improvement Group

Background

• Spin-off from CWI in 2000, self-owned, independent
• Strong academic background, innovative, award-winning, profitable

Activity

• Management advisory, fact-based
• Accredited software analysis lab employs analysis tools and models
• Experienced staff transforms analysis data into advice

Track record

• Finance, government, logistics, telecom, manufacturing, energy, …
• We analyze over 100 systems annually
Who is using our services?

Financials & insurance companies

Public

Retail/Logistics

Technology

Utilities/Telco
Selected services

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

Software Monitoring
- Continuous measurement, feedback, and decision support
- Guard quality from start to finish

Software Product Certification
- Five levels of technical quality
- Evaluation by SIG, certification by TÜV Informationstechnik
Software with high technical quality can evolve with low cost and risk to keep meeting functional and non-functional requirements.
ISO/IEC 9126
Software engineering -- Product quality
  1. Quality model
  2. External metrics
  3. Internal metrics
  4. Quality in use metrics

ISO/IEC 14598
Information technology -- Software product evaluation
  1. General overview
  2. Planning and management
  3. Process for developers
  4. Process for acquirers
  5. Process for evaluators
  6. Documentation of evaluation modules
ISO/IEC 9126, Part 1
Quality perspectives

software product

internal quality

external quality

quality in use

effect of software product

phase

metrics

build 9126, Part 3

test 9126, Part 2

deploy 9126, Part 4
ISO/IEC 9126, Part 1
Software product quality characteristics

ISO/IEC 9126
Software Product Quality

- functionality
- reliability
- usability
- maintainability (analysability, changeability, stability, testability)
- efficiency
- portability
ISO/IEC 9126, Part 1
Maintainability

ISO/IEC 9126: Software Engineering - Product Quality

Maintainability =

- **Analyzability**: easy to understand where and how to modify?
- **Changeability**: easy to perform modification?
- **Stability**: easy to keep coherent when modifying?
- **Testability**: easy to test after modification?
External metrics, e.g.:
- Changeability: “change implementation elapse time”, time between diagnosis and correction
- Testability: “re-test efficiency”, time between correction and conclusion of test

Internal metrics, e.g.:
- Analysability: “activity recording”, ratio between actual and required number of logged data items
- Changeability: “change impact”, number of modifications and problems introduced by them

Critique
- Not pure product measures, rather product in its environment
- Measure after the fact
A Challenge

Use source code metrics to measure technical quality?

Plenty of metrics defined in literature
- LOC, cyclomatic complexity, fan in/out, coupling, cohesion, …
- Halstead, Chidamber-Kemener, Shepperd, …

Plenty of tools available
- Variations on Lint, PMD, FindBugs, …
- Coverity, FxCop, Fortify, QA-C, Understand, …
- Integrated into IDEs

But:
- Do they measure technical quality of a system?
Source code metrics
Cyclomatic complexity

- Accepted in the software community
- Academic: number of independent paths per method
- Intuitive: number of decisions made in a method
- Really, the number of if statements (and while, for, ...)
- Software Engineering Institute:

### Table 4: Cyclomatic Complexity

<table>
<thead>
<tr>
<th>Cyclomatic Complexity</th>
<th>Risk Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>a simple program, without much risk</td>
</tr>
<tr>
<td>11-20</td>
<td>more complex, moderate risk</td>
</tr>
<tr>
<td>21-50</td>
<td>complex, high risk program</td>
</tr>
<tr>
<td>greater than 50</td>
<td>untangleable program (very high risk)</td>
</tr>
</tbody>
</table>

IF FPFI-R21015-6667677V = 'ARC'
  IF FPFI-R21015-6667677V = 'J'
    IF THFI-R21015-6667677V = 'ARC'
      MOVE FPFI-R21015-6667677V TO W-012-V1 END-IF
    IF THFI-R21015-6667677V = 'AJP'
      MOVE FPFI-R21015-6667677V TO W-012-V2 END-IF
    IF THFI-R21015-6667677V = 'ALB'
      MOVE FPFI-R21015-6667677V TO W-012-V3 END-IF
    IF THFI-R21015-6667677V = 'CSK'
      MOVE FPFI-R21015-6667677V TO W-012-V4 END-IF
    IF THFI-R21015-6667677V = 'RHD'
      MOVE FPFI-R21015-6667677V TO W-012-V5 END-IF
    IF THFI-R21015-6667677V = 'SPE'
      MOVE FPFI-R21015-6667677V TO W-012-V6 END-IF
    IF THFI-R21015-6667677V = 'TM' OR
      THFI-R21015-6667677V = 'TV'
      MOVE FPFI-R21015-6667677V TO W-012-V7 END-IF
    IF THFI-R21015-6667677V = 'VR' OR
      THFI-R21015-6667677V = 'VE'
      MOVE FPFI-R21015-6667677V TO W-012-V8 END-IF
    IF THFI-R21015-6667677V = 'FA' OR
      THFI-R21015-6667677V = 'FA'
      MOVE FPFI-R21015-6667677V TO W-012-V9 END-IF
    IF THFI-R21015-6667677V = 'EX' OR
      THFI-R21015-6667677V = 'EX'
      MOVE FPFI-R21015-6667677V TO W-012-V10 END-IF
  ELSE
    IF FPFI-R21015-6667677V = 'ARC'
      MOVE FPFI-R21015-6667677V TO W-012-V1 END-IF
    IF FPFI-R21015-6667677V = 'AJP'
      MOVE FPFI-R21015-6667677V TO W-012-V2 END-IF
    IF FPFI-R21015-6667677V = 'ALB'
      MOVE FPFI-R21015-6667677V TO W-012-V3 END-IF
    IF FPFI-R21015-6667677V = 'CSK'
      MOVE FPFI-R21015-6667677V TO W-012-V4 END-IF
    IF FPFI-R21015-6667677V = 'RHD'
      MOVE FPFI-R21015-6667677V TO W-012-V5 END-IF
    IF FPFI-R21015-6667677V = 'SPE'
      MOVE FPFI-R21015-6667677V TO W-012-V6 END-IF
    IF FPFI-R21015-6667677V = 'TM' OR
      FPFI-R21015-6667677V = 'TV'
      MOVE FPFI-R21015-6667677V TO W-012-V7 END-IF
    IF FPFI-R21015-6667677V = 'VR' OR
      FPFI-R21015-6667677V = 'VE'
      MOVE FPFI-R21015-6667677V TO W-012-V8 END-IF
    IF FPFI-R21015-6667677V = 'FA' OR
      FPFI-R21015-6667677V = 'FA'
      MOVE FPFI-R21015-6667677V TO W-012-V9 END-IF
    IF FPFI-R21015-6667677V = 'EX' OR
      FPFI-R21015-6667677V = 'EX'
      MOVE FPFI-R21015-6667677V TO W-012-V10 END-IF
  END-IF
END-IF
**Source code metrics**

**Coupling**

- **Efferent Coupling (Ce)**
  - How many classes do I depend on?
- **Afferent Coupling (Ca)**
  - How many classes depend on me?
- **Instability = Ce/(Ca+Ce) ∈ [0,1]**
  - Ratio of efferent *versus* total coupling
  - 0 = very stable = hard to change
  - 1 = very instable = easy to change

![Figure 1. Coupling graph](image)

<table>
<thead>
<tr>
<th>Class to Compile</th>
<th>Other Classes Compiled</th>
<th>Afferent Couplings</th>
<th>Efferent Couplings</th>
<th>Instability Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B,C,D,E</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>C,D,E</td>
<td>1</td>
<td>3</td>
<td>0.75</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>3</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>3</td>
<td>1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

*Software Product Quality and Its Effects © 2011 Software Improvement Group*
## Code duplication measurement

<table>
<thead>
<tr>
<th>0: abc</th>
<th>34: xxxxx</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: def</td>
<td>35: def</td>
</tr>
<tr>
<td>2: ghi</td>
<td>36: ghi</td>
</tr>
<tr>
<td>3: jkl</td>
<td>37: jkl</td>
</tr>
<tr>
<td>4: mno</td>
<td>38: mno</td>
</tr>
<tr>
<td>5: pqr</td>
<td>39: pqr</td>
</tr>
<tr>
<td>6: stu</td>
<td>40: stu</td>
</tr>
<tr>
<td>7: vwx</td>
<td>41: vwx</td>
</tr>
<tr>
<td>8: yz</td>
<td>42: xxxxx</td>
</tr>
</tbody>
</table>

Number of duplicated lines: 14
Code duplication

[Bar chart showing code duplication percentages for A, B, C, and D]
How to measure?

Software metrics crisis

Plethora of software metrics
• Ample definitions in literature
• Ample tools that calculate

Measurement yields data, not information
• How to aggregate individual measurement values?
• How to map aggregated values onto quality attributes?
• How to set thresholds?
• How to act on results?

SIG quality model handles these issues in a pragmatic way
Measurable software attributes

**Volume**
- How big? How much invested effort?

**Duplication**
- How lean or bloated? How repetitive?

**Modularity**
- How well organized / subdivided into parts?

**Complexity**
- How much logic / knowledge / decision points?

**Coupling**
- How many interconnections? How intricately weaved together?
The statistical nature of software metrics

Go where the variation is

Observe for all:

- Systems are similar in low percentiles. Systems differ in higher percentiles.
- Interesting differences occur mostly above the 70% percentile

Deriving Metric Thresholds from Benchmark Data
by T. Alves, C. Ypma, J. Visser in ICSM 2010
SIG Quality Model
Quality profiles

1. Measure source code metrics per method / file / module
2. Summarize distribution of measurement values in “Quality Profiles”

<table>
<thead>
<tr>
<th>Cyclomatic complexity</th>
<th>Risk category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 10</td>
<td>Low</td>
</tr>
<tr>
<td>11 - 20</td>
<td>Moderate</td>
</tr>
<tr>
<td>21 - 50</td>
<td>High</td>
</tr>
<tr>
<td>&gt; 50</td>
<td>Very high</td>
</tr>
</tbody>
</table>

Lines of code per risk category

<table>
<thead>
<tr>
<th>Risk category</th>
<th>Low</th>
<th>Moderate</th>
<th>High</th>
<th>Very high</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>70%</td>
<td>12%</td>
<td>13%</td>
<td>5%</td>
</tr>
</tbody>
</table>
How do measurements lead to ratings?

A practical model for measuring maintainability
Heitlager, Kuipers, Visser in QUATIC 2007, IEEE Press

a. Aggregate measurements into “Quality Profiles”
b. Map measurements and quality profiles to ratings for system properties
c. Map ratings for system properties to ratings for ISO/IEC 9126 quality characteristics
d. Map to overall rating of technical quality
Maintainability Model

*Standard two-phase calibration process*

1. **Determine metric thresholds**
   - At level of metric (copybook fan-in)
   - Based on metric values per file
   => four risk categories

2. **Determine mapping to ratings**
   - At level of property rating (module coupling)
   - Based on risk profiles for each system
   => mapping of risk profiles to property ratings

**Data used**
- Selection of “modern systems” from curated warehouse of software analysis results.


Research

- Data: 16 open source systems (2.5 MLOC)
- Mining issues from issue trackers (50K issues)
- Analyzing source code (150 versions)

- Internal quality: maintainability of source code
- External quality: issue handling

1. Correlation analysis
2. Quantification of impact

- *The Influence of Software Maintainability on Issue Handling*
  MSc thesis, Technical University Delft

- *Indicators of Issue Handling Efficiency and their Relation to Software Maintainability*,
  MSc thesis, University of Amsterdam

- *Faster Defect Resolution with Higher Technical Quality of Software, SQM 2010*
Resolution time for defects and enhancements

- Faster issue resolution with higher quality
- Between 2 stars and 4 stars, resolution speed increases by factors 3.5 and 4.0
SIG Quality Model
Quantification

Productivity (resolved issues per developer per month)

- Higher productivity with higher quality
- Between 2 stars and 4 stars, productivity increases by factor 10
Certification

- Third party gives written assurance that a product conforms to specific requirements

Essential elements

- Criteria: based on ISO/IEC 9126
- Evaluation body: institute that examines the product
- Certification body: institute that confirms evaluation process and result

Results

- Evaluation report, including measurements
- Certificate and quality mark: “TÜViT Trusted Product Maintainability”
Evaluation Criteria

Calibrated against benchmark repository

Eligibility for quality mark

- High-level description: fulfill minimal requirements
- Quality ratings: 2 stars or more
- Overall rating: 3 stars or more

Calibration w.r.t. SIG Benchmark Repository

- At level of property ratings
- Against large set of systems
- Multiple technologies, multiple domains
- E.g. about 5% of all systems reach 5 stars for the complexity property

<table>
<thead>
<tr>
<th>Quality Ratings</th>
<th>Overall Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>★★★★★</td>
<td>★★★☆☆☆</td>
</tr>
<tr>
<td>★★★★★</td>
<td>★★★☆☆☆</td>
</tr>
<tr>
<td>★★★★☆☆</td>
<td>★★★☆☆☆</td>
</tr>
<tr>
<td>★★★☆☆☆</td>
<td>★★★☆☆☆</td>
</tr>
<tr>
<td>★★☆☆☆☆</td>
<td>★★☆☆☆☆</td>
</tr>
</tbody>
</table>

High-level description

- ✔ ✔ ✔
# Software Product Certification

**Who uses and how?**

<table>
<thead>
<tr>
<th>Company</th>
<th>Function</th>
<th>Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kas BANK</td>
<td>Tri-party collateral management</td>
<td>internal development</td>
</tr>
<tr>
<td>Rabobank</td>
<td>Bank-lobby console CRM</td>
<td>Ordina / Cognizant India</td>
</tr>
<tr>
<td>ProRail</td>
<td>On-board track visualization</td>
<td>Sogeti</td>
</tr>
<tr>
<td>KLM</td>
<td>Transfer kiosk</td>
<td>Accenture</td>
</tr>
<tr>
<td>SIDN</td>
<td>Domain registration</td>
<td>Provict</td>
</tr>
<tr>
<td>Agentschap BPR</td>
<td>Exchange of citizen information</td>
<td>internal development</td>
</tr>
<tr>
<td>GlobalCollect</td>
<td>Online payment</td>
<td>QuadroVision</td>
</tr>
<tr>
<td>Ordina</td>
<td>Insurance</td>
<td>internal development</td>
</tr>
<tr>
<td>MetaPress (USA)</td>
<td>Document management</td>
<td>SpringerLink</td>
</tr>
<tr>
<td>IT Mobile</td>
<td>Vehicle tracking, fleet management</td>
<td>internal development</td>
</tr>
<tr>
<td>RIPE NCC</td>
<td>Internet resource certification</td>
<td>internal development</td>
</tr>
<tr>
<td>Havenbedrijf Rotterdam</td>
<td>Harbour management</td>
<td>internal development</td>
</tr>
<tr>
<td>Rijkswaterstaat DICT</td>
<td>Infrastructure management</td>
<td>Logica</td>
</tr>
</tbody>
</table>

## Current applications of SIG/TÜViT evaluation criteria

- Meet criteria before **acceptance or deployment**
- Define improvement **roadmaps** towards certifiability
- Include criteria in RFPs, **contracts**, and **SLAs**
What should you remember from this lecture?

**Technical quality of software can be defined and measured**
- ISO/IEC 9126 provides definitions
- SIG quality model performs quantification and rating

**Measurement used to …**
- Set technical requirements
- Monitor quality and progress
- Certify products

**To help achieve …**
- Project success
- Reduction of test effort and rework
- Fast resolution of defects and other changes
- Adaptability under changing requirements