The SQALE Method (V1.0) for Managing Technical Debt

A 15 mn pres

June 2012

Jean-Louis Letouzey
The SQALE method: Context

**SQALE**: Software Quality Assessment based on Lifecycle Expectations

- Based on TD
- Generic

© inspearit 2012
SQALE Structure

1. Quality Model
   - Source Code
   - Analysis tools

2. Analysis Models
   - Estimation models
   - Findings Table
   - Costs Tables

3. Indices
   - SQI
   - STI
   - SRI
   - SQID
   - ... Technical Debt

4. Indicators
   - SQI
   - STI
   - SRI
   - SQID
   - ...
The SQALE Quality Model: Source Code Requirements

- SQALE ask you to organize your set of expectations (requirements) based on your lifecycle needs.

Requirements, appear only once within the Quality Model, when they are first needed. >>> orthogonal model

© inspearit- 2012
SQALE: 2 Estimation Models

- Estimation models transform findings in costs
- One for the Technical Perspective > Technical Debt
- One for the Business Perspective > Business Impact

Remediation Cost

- Depend on the type and the amount of technical activities to perform in order to remediate the NC (remediation life cycle)

Non-Remediation Cost

- Depend on the negative impact on the business activities. The penalty that will cover all damages that will or may happen from delivering with NC

These 2 derived measures are on a ratio scale and can be added without breaching the representation clause

© inspearit- 2012
SQALE indicators: 1 – Info & Analysis

- **Rating:** Synthesis, reporting
  - Based on the ratio: Tech.Debt/Dev.Cost

- **Pyramid:** Technical Perspective of your TD:

  ![SQALE Pyramid](image)

  Tool Vendors may offer additional indicators (historical, by dev., by age....)
SQALE indicators: 2 - Priorities

**Enough time/budget**
- Follow the logic from the **Pyramid**, start from the bottom

<table>
<thead>
<tr>
<th>SQALE Pyramid</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>0.0</td>
</tr>
<tr>
<td>Maintainability</td>
<td>27.1</td>
</tr>
<tr>
<td>Security</td>
<td>0.0</td>
</tr>
<tr>
<td>Efficiency</td>
<td>2.1</td>
</tr>
<tr>
<td>Changeability</td>
<td>268.7</td>
</tr>
<tr>
<td>Reliability</td>
<td>55.9</td>
</tr>
<tr>
<td>Testability</td>
<td>289.9</td>
</tr>
</tbody>
</table>

- Relevant strategy for projects starting from scratch or the new part of the debt on a maintenance project

**Limited time/budget**
- Look for the best Impact/Cost ratio. Use the SQALE **Debt Map**

- Relevant strategy for improving the quality of legacy apps

© inspearit- 2012
Managing Technical Debt with SQALE

1. Define what create TD
2. Define how you calculate TD
3. Set Goals
4. Monitor the TD
5. Compare TD
6. Analyse TD (origin, location, and impact)
7. Set Pay down goals
8. Set Pay down plan/priorities

SQALE Support

- Quality Model
- Analysis Model
- Index density, Rating, Kiviat
- SQI index
- SQI Density
- Pyramid, SQI, SBII
- Indices density, Rating, Kiviat
- The 2 Perspectives, Pyramid, Debt Map

© inspearit- 2012
SQALE Status

- Public, open source, royalty free
- Language, Dev. process, Tool editor independant
- Largely deployed and used
- The SQALE website
  - Method Definition Document
  - Link to Thesis, Presentations, articles, tools
  - Blog

- **Inspearit assets**
  - 1 day Method Training
  - Inspearit calibrated Quality and Estimation Models
  - Evaluation process
  - Evaluation Framework
  - Measurement database

© inspearit- 2012
just sqale it

Thanks

Questions?

http://www.sqale.org

jean-louis.letouzey@inspearit.com
Addendum; Additional material

- Remediation functions
- « Right Code »
- Does it generates TD?
- Managing upon agile principles
- More details about the SQALE Quality Model
- More details about the SQALE indicators
- Some SQALE tool screenshots
Remediation functions

Remediation workload (and remediation functions) depend on the « remediation lifecycle »

Developers use a limited number of « remediation lifecycle »

- Check out
- Fix auto.
- Check In

- Check out
- Fix manually
- Check In

- Check out
- Fix
- Run exist. tests
- Check In

- Check out
- Fix
- Dev. & run new tests
- Run exist. tests
- Check In

© inspearit- 2012
What is « right code »

« Right code » is not « perfect code »

As soon as one attribute is in the unacceptable area, the code is « not right »
Does it generates technical debt?

- Debt could come without a commitment or a promise?
- My portfolio is made of 60% Java and 40% C#. This is not optimized compared to 100% in the same language. Is that TD? *(same question with 60% Oracle and 40% DB2)*
- My application has been developed some time ago with Oracle 9. Since then, Oracle moved to V11. That’s due to technology obsolescence. Is that TD?
- My application works fine for our current 500 users. It won’t support 5,000 users without a complete redesign. Is that TD?
- Some people in my team don’t have the expected skills and should attend dedicated training on xyz. Is that TD?
Managing upon agile principles

- Priorities are established upon the « Value/Price » ratio

Features

- Development Cost
- Business Value

Non-Conformities

- Technical Debt
- Business Impact

Priorities/Decision

Features List

Business Perspective

N.C. List

Technical Perspective

Business Perspective
SQALE is based on Lifecycle Expectations

ISO 25010

- Maintainability
- Portability
- Efficiency
- Reliability
- Security

Reuse
- Reusability

Port
- Portability

Maintain
- Maintainability
- Security
- Efficiency
- Changeability
- Reliability

Change
- Testability

Test
- Code

Deliver
- Efficiency

Iso 25010 Quality characteristics are mapped to the file life cycle activities

© inspearit- 2012
Any type of requirements

- Any type of code related requirements are accepted, provided they are justified and verifiable

- It is possible to remove characteristics

- Generally, depending on project or organization’s context, a SQALE Quality model contains between 40 and 100 requirements

© inspearit- 2012
The SQALE rating

- A synthetic indicator for management dashboards
- Depend directly on the TD/DEVCost ratio
The SQALE technical perspective

- Distribution of the debt upon links to characteristic/activity

<table>
<thead>
<tr>
<th>SQALE Pyramid</th>
<th>Cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>0.0</td>
<td>643.7</td>
</tr>
<tr>
<td>Maintainability</td>
<td>27.1</td>
<td>643.7</td>
</tr>
<tr>
<td>Security</td>
<td>0.0</td>
<td>616.6</td>
</tr>
<tr>
<td>Efficiency</td>
<td>2.1</td>
<td>616.6</td>
</tr>
<tr>
<td>Changeability</td>
<td>268.7</td>
<td>614.5</td>
</tr>
<tr>
<td>Reliability</td>
<td>55.9</td>
<td>345.8</td>
</tr>
<tr>
<td>Testability</td>
<td>289.9</td>
<td>289.9</td>
</tr>
</tbody>
</table>

© inspearit- 2012
The SQALE business perspective

- Distribution of the Debt upon the criticity

<table>
<thead>
<tr>
<th>SQALE Remediation Costs to reduce risk</th>
<th>Cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocker</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Critical</td>
<td>8.9</td>
<td>8.9</td>
</tr>
<tr>
<td>Major</td>
<td>598.4</td>
<td>607.3</td>
</tr>
<tr>
<td>Minor</td>
<td>35.2</td>
<td>642.5</td>
</tr>
<tr>
<td>Info</td>
<td>1.2</td>
<td>643.7</td>
</tr>
</tbody>
</table>

© inspearit- 2012
The SQALE Debt Map

- An analysis indicator valid at all artefact level

Priority

- File
- Component
- Application

© inspearit- 2012
SQALE Dashboard sample

Focus on technical debt and its distribution
SQALE Implementation: Tools

Example: Metrixware
SQALE implementation: Tools

Example: Squoring