The SQALE method: Meaningful insights into your Technical Debt

Jean-Louis Letouzey

2012
Table of Content

- Reminder on Quality and Technical Debt
- The SQALE method and the Technical Debt
- The SQALE method and the Business Perspective
- Managing your Technical Debt with SQALE
- A powerful paradigm shift
- How to deploy/use the SQALE method
- Demonstration
Quality

- Quality is: compliance to requirement
- Before measuring Quality, you first need to define it
  - At project level or
  - Organization level
Technical Debt

Ward Cunningham at OOPSLA 92: “Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite...

“Every minute spent on not-quite-right code counts as interest”
Technical Debt

- A strong communication tool
- Definition may differ from experts
- Product – Process
- Voluntary – involuntary
- Short term – Long term

- A specificity: Technical Debt disappears at the end of the Software’s life
- What really matters is the nature and duration of the interests

- One reference book:
Technical Debt and Agility

- Agile considers the source code as one major deliverable (not design model, not documentation)
- Agile promotes transparency

- The definition of « right code » should be considered for the Definition(s) of Done « DoD »
  - List of source code quality requirements
  - Type and acceptable level of Technical Debt

- Technical Debt shall be identified and monitored
- Project shall plan and prioritize activities for repaying and limiting this debt
Table of Content

- Reminder on Quality and Technical Debt
- The SQALE method and the Technical Debt
- The SQALE method and the Business Perspective
- Managing your Technical Debt with SQALE
- A powerful paradigm shift
- How to deploy/use the SQALE method
- Demonstration
The SQALE method

- Public under an open source licence
- Generic
  - Tool independant
  - Language independant
  - Development Process independant
- Widely used
The SQALE method is simple

1. Quality Model
2. Analysis Model
3. Indices
4. Indicators

Source Code Analysis Tools

Source code related requirements

Findings Table

Remediation costs Table

Remediation functions

Technical Debt

d. / h. / $ ....

« Right Code »
The SQALE method is simple

1. Quality Model
2. Analysis Model
3. Indices
4. Indicators

Source Code Analysis tools

« Right Code »

<table>
<thead>
<tr>
<th>Remediaion functions</th>
<th>Remediation costs Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source code related requirements</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Findings Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quality Model</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Analysis Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQI</td>
</tr>
<tr>
<td>STI</td>
</tr>
<tr>
<td>SRI</td>
</tr>
<tr>
<td>SQID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>d. / h. / $ ....</td>
</tr>
<tr>
<td>d. / h. / $ ....</td>
</tr>
</tbody>
</table>

© inspearit
Ordered Requirements

- Upon their relative importance (low, ..., blocking)
- Upon Life Cycle activities

« Right Code »

Requirements, appear only once within the Quality Model, when they are first needed.
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.
The SQALE Pyramid: 2 points of view

An **analytic view** provided by orthogonal characteristics. One understands impact of each Non Conformity and improvement on quality characteristic and life cycle issues.

An **external view** that represents the perceived quality evaluated by consolidation of the hierarchy of characteristics.
The SQALE rating

- A synthetic indicator dedicated to management dashboards
- Based on the ratio between the Technical Debt and the development cost
Table of Content

- Reminder on Quality and Technical Debt
- The SQALE method and the Technical Debt
- The SQALE method and the Business Perspective
- Managing your Technical Debt with SQALE
- A powerful paradigm shift
- How to deploy/use the SQALE method
- Demonstration
The Non-remediation Cost

- To fix non « right code » has an impact on the technical activities and an associated remediation cost: The Technical Debt
- To keep and deliver non « right code » has an impact on the business activities and an associated non-remediation cost: The Business Impact
- In SQALE this Business impact is estimated/quantified to help the analysis and the prioritization of remediation activities
The Non-remediation functions

A Non-remediation function is associated to each requirement to represent its relative « importance »

1. Quality Model
   - Source code related requirements

2. Analysis Model
   - Non-remediation functions

3. Indices

4. Indicators

Business Impact

d. / h. / $ ....

Findings Table

1
5
6
1

Non-remediation costs Table

100
1

« Right Code »
The SQALE Debt Map

- An analysis indicator valid at all artefact level

- File
- Component
- Application
Managing upon agile principles

Priorities are established upon the « Value/Price » ratio
Table of Content

- Reminder on Quality and Technical Debt
- The SQALE method and the Technical Debt
- The SQALE method and the Business Perspective
- Managing your Technical Debt with SQALE
- A powerful paradigm shift
- How to deploy/use the SQALE method
- Demonstration
Managing your Technical Debt

Identify your Technical Debt
- Define what produce debt
- Define your debt estimation model
- Calculate your debt

Analyze your debt (upon its origin, impact, age...)

Pay your debt back with an optimized plan
- For new agile project
- For agile project with a « legacy » part
The SQALE pyramid defines a logical priority for remediation actions.
Analyze your Technique Debt

The SQALE Pyramid provides a technical perspective: Impact on the project’s activities

<table>
<thead>
<tr>
<th>SQALE Pyramid</th>
<th>Cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>0.8</td>
<td>1,314.4</td>
</tr>
<tr>
<td>Maintainability</td>
<td>353.7</td>
<td>1,313.6</td>
</tr>
<tr>
<td>Security</td>
<td>32.1</td>
<td>959.9</td>
</tr>
<tr>
<td>Efficiency</td>
<td>39.1</td>
<td>927.8</td>
</tr>
<tr>
<td>Changeability</td>
<td>665.7</td>
<td>888.7</td>
</tr>
<tr>
<td>Reliability</td>
<td>146.0</td>
<td>223.0</td>
</tr>
<tr>
<td>Testability</td>
<td>77.0</td>
<td>77.0</td>
</tr>
</tbody>
</table>

*Agility*
Which code is more « agile »?

<table>
<thead>
<tr>
<th>SQALE Pyramid</th>
<th>Cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>0.8</td>
<td>1,314.4</td>
</tr>
<tr>
<td>Maintainability</td>
<td>353.7</td>
<td>1,313.6</td>
</tr>
<tr>
<td>Security</td>
<td>32.1</td>
<td>959.9</td>
</tr>
<tr>
<td>Efficiency</td>
<td>39.1</td>
<td>927.8</td>
</tr>
<tr>
<td>Changeability</td>
<td>665.7</td>
<td>888.7</td>
</tr>
<tr>
<td>Reliability</td>
<td>146.0</td>
<td>223.0</td>
</tr>
<tr>
<td>Testability</td>
<td>77.0</td>
<td>77.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SQALE Pyramid</th>
<th>Cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>17.2</td>
<td>1,303.9</td>
</tr>
<tr>
<td>Maintainability</td>
<td>158.2</td>
<td>1,286.7</td>
</tr>
<tr>
<td>Security</td>
<td>19.5</td>
<td>1,128.5</td>
</tr>
<tr>
<td>Efficiency</td>
<td>24.0</td>
<td>1,109.0</td>
</tr>
<tr>
<td>Changeability</td>
<td>239.1</td>
<td>1,085.0</td>
</tr>
<tr>
<td>Reliability</td>
<td>766.0</td>
<td>845.9</td>
</tr>
<tr>
<td>Testability</td>
<td>79.9</td>
<td>79.9</td>
</tr>
</tbody>
</table>
Deliver with a residual debt

**Technical Debt** is permanently monitored and analyzed for identifying and performing immediate remediations.
Deliver with a residual debt

Technical Debt is permanently monitored and analyzed for identifying and performing immediate remediations.

Optimization phase: One need to take into account the Business perspective in order to minimize the impact of the residual debt.
Analyze your Technical Debt

- The **business perspective** upon « Severity », « Importance »

### SQALE Remediation Costs to reduce risk

<table>
<thead>
<tr>
<th>Severity</th>
<th>Blocker</th>
<th>Critical</th>
<th>Major</th>
<th>Minor</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost</strong></td>
<td>40.3</td>
<td>409.4</td>
<td>20.4</td>
<td>20.5</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>40.3</td>
<td>449.7</td>
<td>470.0</td>
<td>490.6</td>
<td>490.6</td>
</tr>
</tbody>
</table>

### SQALE Remediation Costs to reduce risk

<table>
<thead>
<tr>
<th>Severity</th>
<th>Blocker</th>
<th>Critical</th>
<th>Major</th>
<th>Minor</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost</strong></td>
<td>0.0</td>
<td>0.0</td>
<td>503.0</td>
<td>0.0</td>
<td>22.1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>0.0</td>
<td>0.0</td>
<td>503.0</td>
<td>503.0</td>
<td>525.1</td>
</tr>
</tbody>
</table>
Optimize your remediation budget

Use the impact/cost ratio

Diagram showing a scatter plot with axes labeled "Business Impact" and "Non-remediation Cost" and "Remediation Cost." The plot includes a red dashed line indicating the impact/cost ratio.
Dealing with « Legacy » code

- Technical Debt is often too high to be resorbed
- Several strategies:
  - Limit the global growth of the debt (i.e.: <2%)
  - Resorb the debt related to blocking or critical (the violations associated to high Non-Remediation Costs)
  - « The boy scout rule »: we should leave the code cleaner than we found it.....

- Identify your strategic applications and plan specific budget and resources for cleaning up part of the debt and decrease the future « interest »
Table of Content

- Reminder on Quality and Technical Debt
- The SQALE method and the Technical Debt
- The SQALE method and the Business Perspective
- Managing your Technical Debt with SQALE
- A powerful paradigm shift
- How to deploy/use the SQALE method
- Demonstration
The need for a new paradigm

- Source code measurement project have difficulties when deployed on a large perimeter
- An SEI report states a high failure ration
  - 80% of projects are stopped before 24 months

Let’s try to identify some causes...
MAINTAINABILITY INDEX

\[ MI^4 = 171 - 3.42 \ln(\text{aveE}) - 0.23 \text{aveV}(g') - 16.2 \ln(\text{aveLOC}) + (50 \times \sin(\sqrt{2.46 \times \text{aveCM}})) \]

CLASS COHESION

\[ LCOM = LCOM = \frac{(1/a \times \Sigma A) - m}{(1 - m)} \]

Where \( a \) is the number of attributes of the class, \( \Sigma A \) is the sum across the set of attributes of the number of methods that access each attribute, and \( m \) is the number of methods of the class.
Aggregations
On the opposite, Technical Debt is simple
A simple but powerful paradigm

Technical Debt

- Simple calculation
- Simple aggregation
- Objectivity
- Wide public
- Few false positive
- Representativity
- Clear rating rule
- Comparisons
- Remediation priorities
- Analysis perspectives

SQALE method

New

Past

Balance

© inspearit-2012
Table of Content

- Reminder on Quality and Technical Debt
- The SQALE method and the Technical Debt
- The SQALE method and the Business Perspective
- Managing your Technical Debt with SQALE
- A powerful paradigm shift
- How to deploy/use the SQALE method
- Demonstration
SQALE deployment

1. Initialization
   - Planification
   - Stakeholders
   - Perimeter...

2. Tailoring
   - Method training
   - Tailoring the SQALE
   - basic models

3. Pilot
   - Model Validation
   - Building a ready to deploy solution

4. Deployment
   - Tools installation & integration
   - Awareness and Coaching sessions

- A simple technical part
  - Setting your own SQALE models
  - Tool implementation and validation

- A « Change management » part
  - Fighting against old ideas about code measurement
  - Addressing a large population
The SQALE method: Summary

- Easy to understand and to deploy
- A strong and powerful implementation of the Technical Debt concept
- Open source and tool independent
- Used worldwide
- Expert recognition

“In the domain of software quality evaluation, I find SQALE – Software Quality Assessment based on Life Cycle Expectations – a great tool for implementing my mantra. It interprets source code analysis in terms of what really matters in the specific client environment. In so doing, it transforms an overwhelming set of measurement data to actionable insights which are meaningful at multiple levels of the firm.”

Israel Gat, Cutter Consortium
Table of Content

- Reminder on Quality and Technical Debt
- The SQALE method and the Technical Debt
- The SQALE method and the Business Perspective
- Managing your Technical Debt with SQALE
- A powerful paradigm shift
- How to deploy/use the SQALE method
- Demonstration
Topics to cover during demo (15 mn)

- Sonar architecture and eco-system
- Setting a SQALE Quality Model (= Sonar profile)
- Setting the SQALE Sonar plugin
  - Mapping requirements to characteristics and subchars
  - Setting the Remediation Functions
  - Setting the Rating threshold
  - Setting Non-Remediation factors
- Using SQALE Dasboards and indicators
  - Building a dashboard
  - Analyzing the Technical Debt and Business Impact
  - Navigation down to the source code
- Other features
  - Technical debt history
  - Treemap for portfolio management