The SQALE method: Meaningful insights into your Technical Debt*

Jean-Louis Letouzey

August 2012

* The technical debt visible in your source code
Presentation

Jean-Louis Letouzey
Expert consultant at inspearit France

- Product Audit, Technical Debt assessment
- Author of the SQALE method
- Preaching, Teaching SQALE
- SQALE tailoring and deployment in large organizations
Table of Contents

- Reminder on Quality and Technical Debt
- Technical Debt: a powerful Paradigm Shift
- The SQALE method: Structure
- Analyzing Technical Debt with SQALE
- Paying back Technical Debt with SQALE
- Summary and Conclusion
- Demonstration
Quality

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

If you don’t define it, you won’t get it!
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
- Understandable by managers (it fits well in Excel sheets)
- Many blogs, definitions from experts
  - Product – Process
  - Voluntary – involuntary
  - ...
- One reference book:
Technical Debt

In a recent (2012) interview* Ward Cunningham provided some clarifications:

❖ “We can say that the **CODE** is of high quality when productivity remains high in the presence of change in **TEAM** and **GOALS.**”

Technical debt = Work to be done = Principal Impact = Lost of productivity = Interest

* www.techdebt.org
Technical Debt and Agility

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

- Agile promotes transparency
- Technical Debt shall be identified and monitored
- Project shall plan and prioritize activities for repaying and limiting this debt
Technical Debt and Agile Projects

"DoD" Including Definition of « Right Code » Acceptable type and level of debt

Part of debt to be repaid within the sprint

« Continuous inspection » Monitoring TD

© inspearit-2012
What Managing Technical Debt Means?

To Manage TD means at least:

1. Define what creates TD
2. Define how to calculate TD
3. Set Goals at organizational or project level
4. Monitor the TD against the goals
5. Compare TD across versions, projects, subcontractors...
6. Analyze TD (age, location and impact)
7. Set Pay down goals
8. Set Pay down plan/priorities
9. ...
Table of Content

- Reminder on Quality and Technical Debt
- **Technical Debt: a powerful Paradigm Shift**
- The SQALE method: Structure
- Analyzing Technical Debt with SQALE
- Paying back Technical Debt with SQALE
- Summary and Conclusion
- Demonstration
Since the 70’s, the Software community is trying to measure « Quality » and Quality of source code

Technical Debt makes the difference on 4 points

1st point: Technical debt measures “Non Quality”!

“Good Code” Measures 8

“Bad Code” Measures 0

“Bad Code”

“Right Code”
2nd point: “Right code”

- Technical Debt refers to “Right code”, not “perfect code”

- As soon as one attribute is in the unacceptable area (i.e. potentially decreasing productivity), the code is “not right”
3rd point: Formulas: 5 days training!

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

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

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.
4th point: Aggregations: no averages!
Benefits from the Paradigm Shift

<table>
<thead>
<tr>
<th>TD Specificities</th>
<th>Benefits at measure level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation by addition</td>
<td>No masking, no false positive</td>
</tr>
<tr>
<td>Inverse direction</td>
<td>Objective</td>
</tr>
<tr>
<td>« Right » vs « Perfect »</td>
<td>Precise</td>
</tr>
<tr>
<td>Ease of understanding</td>
<td>Short training</td>
</tr>
</tbody>
</table>

Technical debt is a much more powerful system compared to everything used previously to measure source code
Table of Content

- Reminder on Quality and Technical Debt
- Technical Debt: a powerful Paradigm Shift
- The SQALE method: Structure
- Analyzing the Technical Debt with SQALE
- Paying back Technical Debt with SQALE
- Summary and Conclusion
- Demonstration
The SQALE method: Context

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

- Based on TD
- Open source
- Generic (a framework)
- Tool independent
- Widely used

Definition document, articles, tools list are on sqale.org

© inspearit-2012 18
SQALE Structure

1 Quality Model

Source Code related requirements

Analysis tools

2 Analysis Models

Findings Table

Estimation models

Costs Tables

3 Indices

SQI
STI
SRI
SQID

4 Indicators

Technical Debt

d. / h. / $ ....

« Right Code » Definition

© inspearit-2012
The SQALE Quality Model: Source Code Requirements

SQALE asks you to associate each of your expectations (requirements) with a quality characteristic.

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

Depends on the type and the amount of technical activities to perform in order to remediate the violations (remediation life cycle)

Non-Remediation Cost

Depends on the negative impact on the business activities. The penalty that will cover all damages that will or may happened from delivering with violations

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

© inspearit-2012
### Table of Content

- Reminder on Quality and Technical Debt
- Technical Debt: a powerful Paradigm Shift
- The SQALE method: Structure
- **Analyzing Technical Debt with SQALE**
- Paying back Technical Debt with SQALE
- Summary and Conclusion
- Demonstration
SQALE Structure

1 Quality Model
   - Source Code related requirements

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

3 Indices
   - SQI
   - Σ
   - Costs
   - d. / h. / $ ....
   - Technical Debt

« Right Code » Definition

© inspearit-2012
What Managing Technical Debt Means?

To **Manage** TD means at least:

1. Define what creates TD
2. Define how you calculate TD
3. Set Goals at organizational or project level
4. Monitor the TD against the goals
5. Compare TD across versions, projects, subcontractors...
6. Analyze TD (age, location and impact)
7. Set Pay down goals
8. Set Pay down plan/priorities
9. ...

**SQALE:**

Quality Model Remediation Function
Index: Technical Debt Density

- SQALE defines Density indicators
- You divide your TD by the size of your artifact
- Size may be measured in function points, KSLOC...

SQID (SQALE Quality Index Density = Technical Debt Density)

- Allows to compare projects, versions, subcontractors...
- Allows to set goals without knowing the size in advance
Indicator: The SQALE rating

- A synthetic indicator dedicated to management dashboards
- Based on the ratio between the Technical Debt and the Development Cost
What Managing Technical Debt Means?

To **Manage** TD means at least:

1. Define what creates TD
2. Define how you calculate TD
3. Set Goals at organizational or project level
4. Monitor the TD against the goals
5. Compare TD across versions, projects, subcontractors...
6. Analyze TD (age, location and impact)
7. Set Pay down goals
8. Set Pay down plan/priorities
9. ...

**SQALE:**
- Quality Model
- Remediation Function
- Index Densities
- Rating
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.
Analyze your Technical 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>

© inspearit-2012
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>
What Managing Technical Debt Means?

To **Manage** TD means at least:

1. Define what creates TD
2. Define how you calculate TD
3. Set Goals at organizational or project level
4. Monitor the TD against the goals
5. Compare TD across versions, projects, subcontractors...
6. Analyze TD (age, location and impact)
7. Set Pay down goals
8. Set Pay down plan/priorities
9. ...

**SQALE:**

- Quality Model
- Remediation Function
- Index Densities
- Rating
- Pyramid
Table of Content

- Reminder on Quality and Technical Debt
- Technical Debt: a powerful Paradigm Shift
- The SQALE method: Structure
- Analyzing Technical Debt with SQALE
- Paying back Technical Debt with SQALE
- Summary and Conclusion
- Demonstration
The ways to prioritize features

- When budget or time is limited, features must be prioritized. 2 perspectives are commonly considered to prioritize features.
- In other cases (?), a technical logic is applied.
- The same logic is applicable to Non-Conformities.

### Features

<table>
<thead>
<tr>
<th>Technical Perspective</th>
<th>Business Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Cost</td>
<td>Business Value</td>
</tr>
</tbody>
</table>

#### Priorities/Decision

- Features List
- Business Perspective

### Non-Conformities

<table>
<thead>
<tr>
<th>Technical Perspective</th>
<th>Business Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical Debt</td>
<td>Business Impact</td>
</tr>
</tbody>
</table>

#### Priorities/Decision

- N.C. List
- Business Perspective
Paying back your Technique Debt

SQALE supports 3 strategies upon 3 perspectives

The SQALE pyramid defines a **logical** priority for remediation actions. This is a **technical perspective**
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.

Acceptable Level

Residual Technical Debt

Time

Impact
Analyze your Technical Debt

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

---

**SQALE Remediation Costs to reduce risk**

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

**SQALE Remediation Costs to reduce risk**

<table>
<thead>
<tr>
<th>Category</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>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Major</td>
<td>503.0</td>
<td>503.0</td>
</tr>
<tr>
<td>Minor</td>
<td>0.0</td>
<td>503.0</td>
</tr>
<tr>
<td>Info</td>
<td>22.1</td>
<td>525.1</td>
</tr>
</tbody>
</table>

© inspearit-2012
The SQALE Debt Map

- An analysis indicator valid at all artifact level. Use the business impact to support the business perspective

- File
- Component
- Application
Optimize your remediation budget

- Use the impact/cost ratio to support the combined perspective
Table of Content

- Reminder on Quality and Technical Debt
- Technical Debt: a powerful Paradigm Shift
- The SQALE method: Structure
- Analyzing Technical Debt with SQALE
- Paying back Technical Debt with SQALE
- Summary and Conclusion
- Demonstration
A simple but powerful paradigm

- Simple calculation
- Simple aggregation
- Objectivity
- Wide public
- Few false positives
- Representativeness
- Clear rating rule
- Comparisons
- Remediation priorities
- Analysis perspectives
Paying back the TD with SQALE

SQALE supports 3 different strategies:

1. **1st strategy**: Follow the technical Logic (avoid useless rework)
   - Use the SQALE pyramid

2. **2nd strategy**: decrease the business impact by starting fixing the Non-conformities with the highest business impact
   - Use the SQALE Business Impact (SBI)
   - Use the Debt Map

3. **3rd strategy**: optimize your ROI by starting fixing the non-conformities with the highest ROI
   - Use the Business Impact/Remediation cost ratio
   - Use the Debt Map

Depending on your context, choose one of the single perspectives or the combined perspective
SQALE deployment

1. Initialization
   - Planning Stakeholders Scope...

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
- Build in pay-back support
- 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

© inspeartit-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
just sqale it

Thanks
Questions?

www.inspearit.com

http: /www.sqale.org

Follow #sqale on twitter

jean-louis.letouzey@inspearit.com