

The SQALE Method for Managing Technical Debt

Introduction

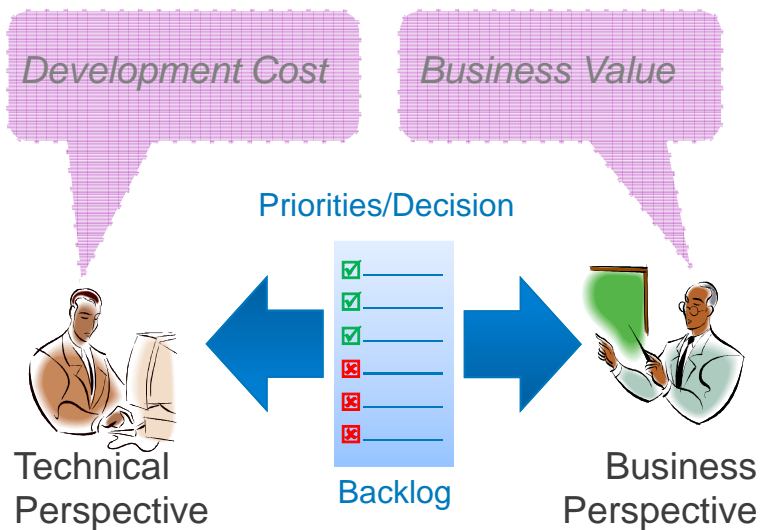
Jean-Louis Letouzey

The SQALE method

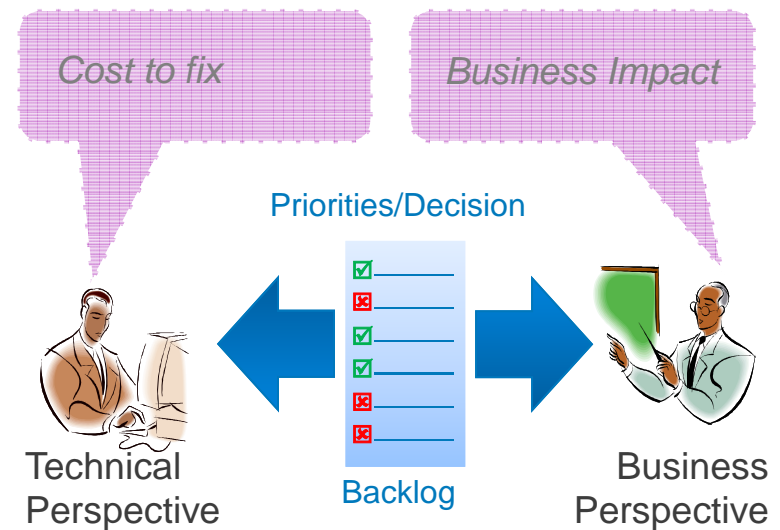
- A meta-method, a framework for managing Technical Debt
- **Open source** (www.sqale.org), no royalty
- Tool independent (as today: 4 tools)
- A very large number of users

How to prioritize?

“Factoring”



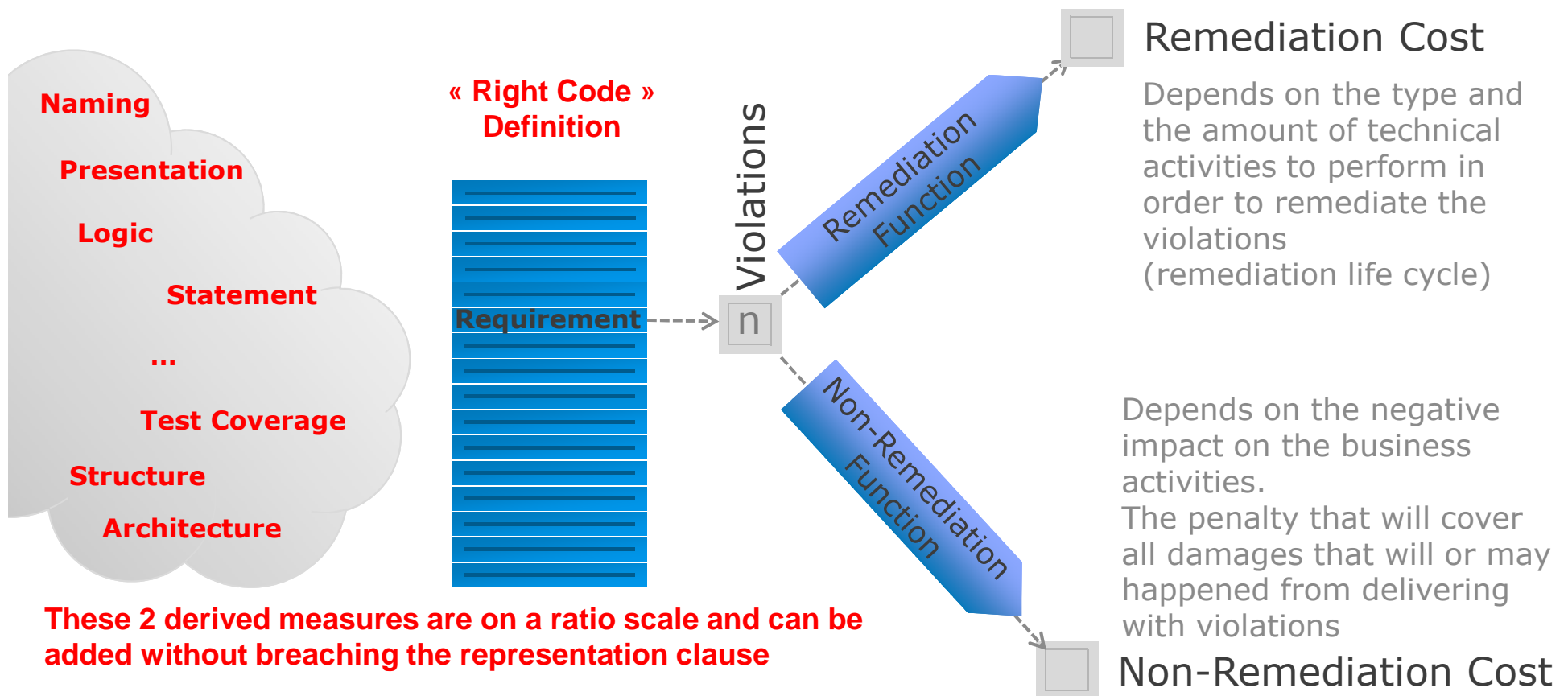
“Refactoring”



- Major aspects
 - Dev cost
 - Business Value
 - Dependencies

SQALE: 2 Estimation Models

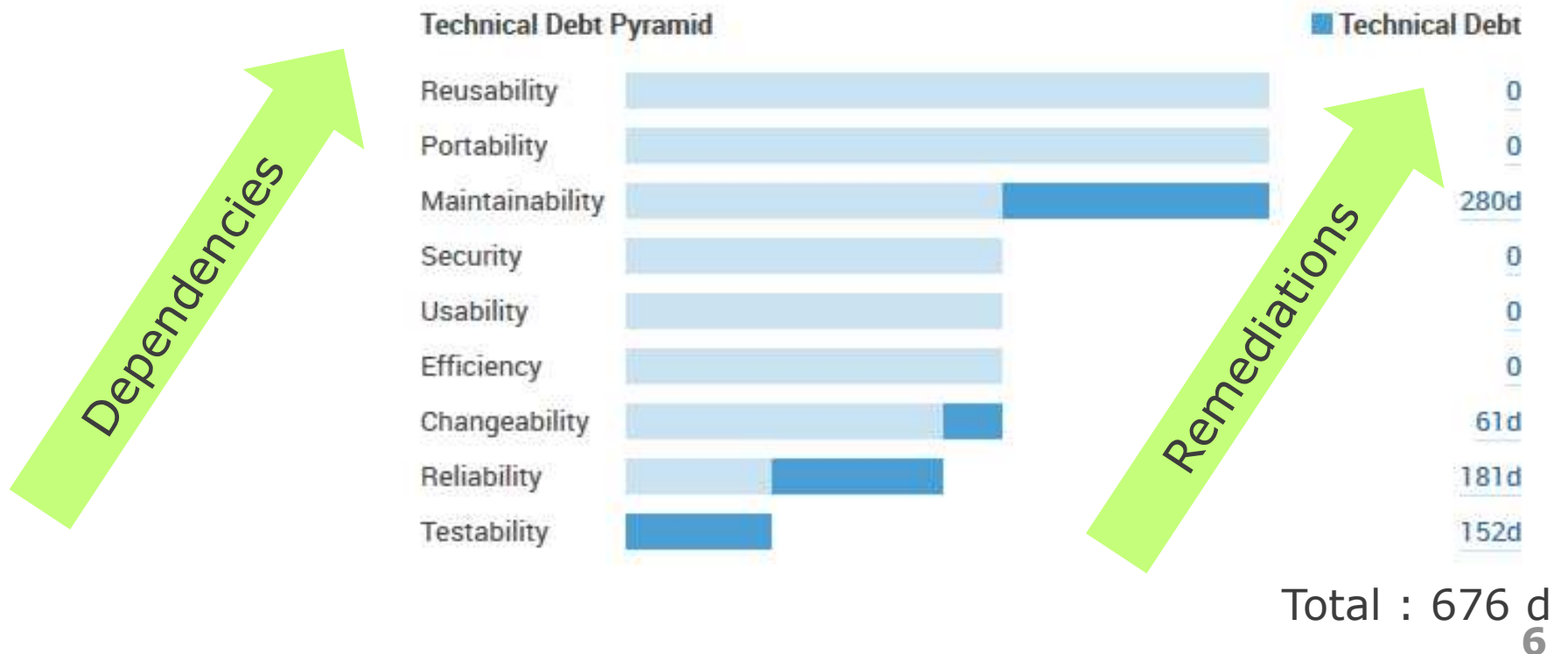
- Estimation models transform tool findings in costs
 - One for the Technical Perspective > Technical Debt (principal)
 - One for the Business Perspective > Business Impact (interest)



Paying back your Technique Debt - 1

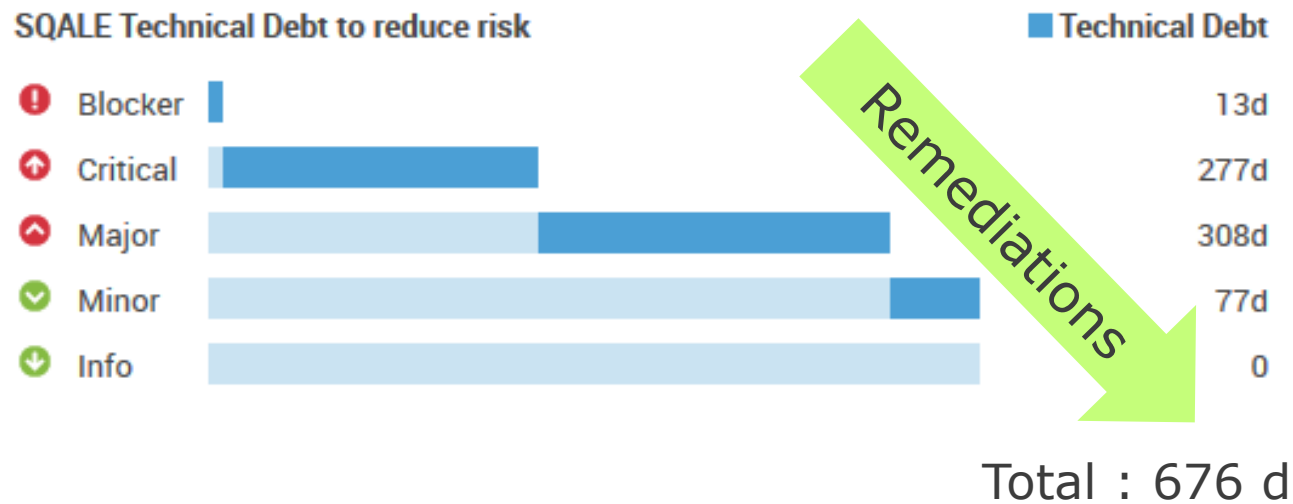
SQALE supports 3 strategies

- ✦ The SQALE pyramid defines a **logical** priority (technical dependencies) for remediation actions. This is a **technical perspective**



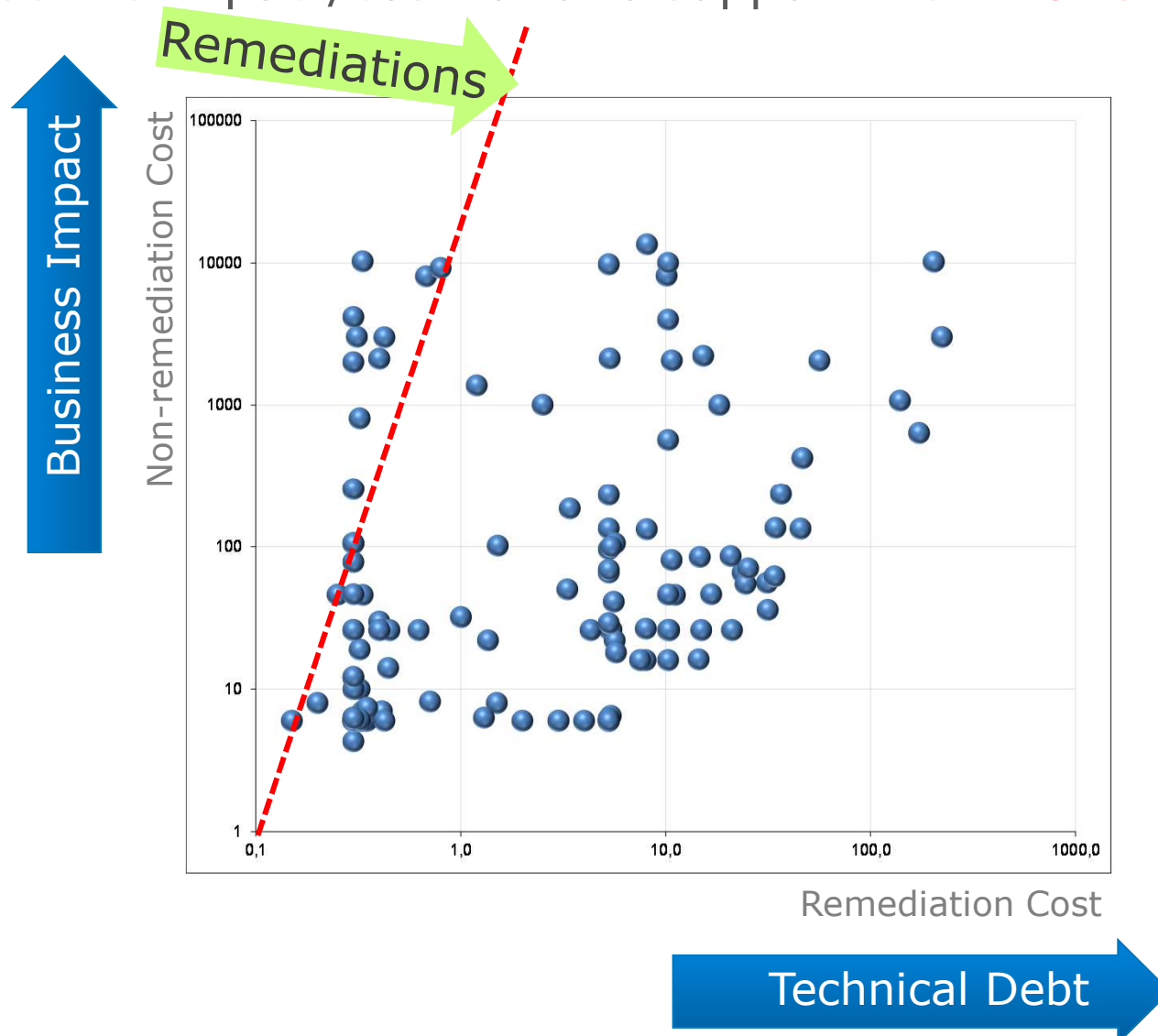
Paying back your Technique Debt - 2

- ✦ The **business perspective** upon « Severity », « Impact »



Paying back your Technique Debt - 3

- Use the impact/cost ratio to support the "ROI strategy"



Managing TD at Portfolio level

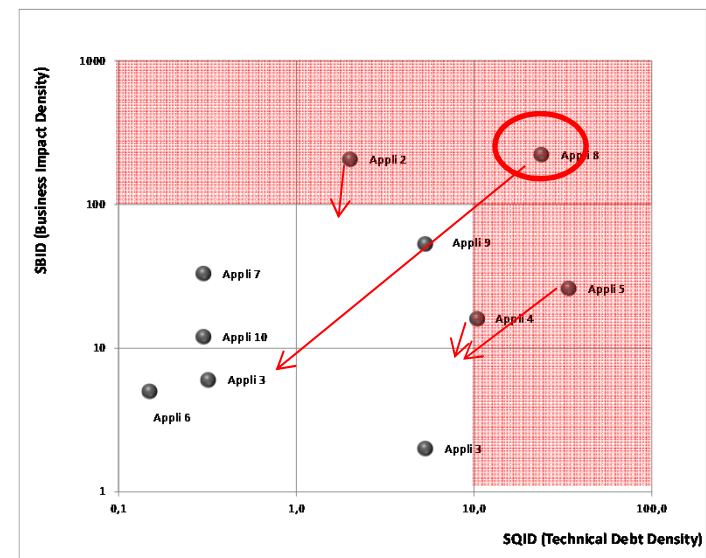
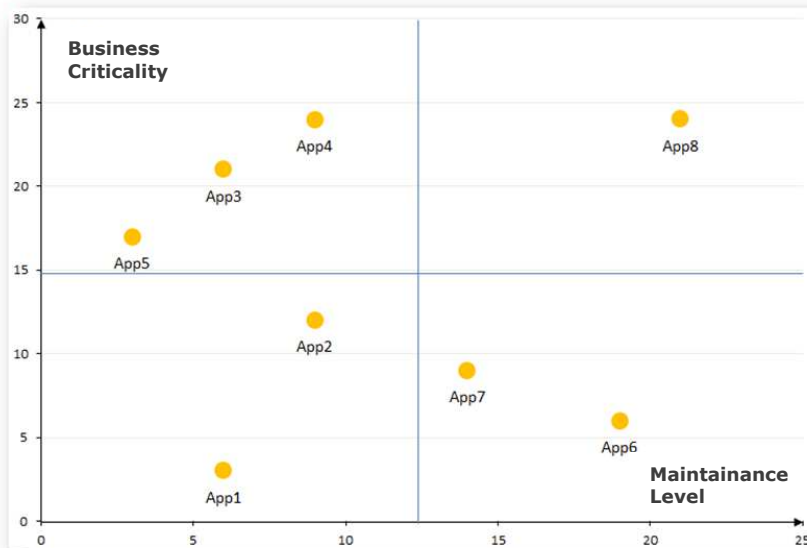
- Because of history, software entropy, the quality of your portfolio will never be as good as you may expect.
- You will never have all the time and budget to improve all your portfolio at "A" level. **It is a question of compromise**
- A 3 step Process/Strategy

Focus on major Gaps

Operational Analysis

VS

Code analysis results



1. Establish an inventory of your portfolio and identify expected quality level for each of your applications
2. Compare their amount and type of Technical Debt against your expectation
3. Focus energy on highest gaps

How to deploy?

- Estimation models:
 - No standard for that
 - Tools default values are not precise in all contexts
- Organize workshops with experts to build their models based on their local assumptions
- Configure the tool according to the organization model

[Duplicated blocks](#) Common SonarQube ▼ ▼ Default: Logic Default: Linear Default: 1h

Examples

- **Assumption example:**

- H4: Remediation Cost covers: coding, unit testing development and debug, and regression testing update.
- H5: Developers have coding experience. They are not novices, nor are they expert coders but they know how to use their development environment effectively.
- H7: Developers or the development team owns the code they work on. The developer understands what he/she is programming (or will get a quick support from other team members).

- **Impact levels example:**

Impact level	Typical Potential Business impact of non Remediation	Detail/ Examples	Relative Business impact
Very High	May result in bug in production if the line is executed	Potential division by zero, Dangerous cast, NPE, missing logic without commented justification	5000
High	Major additional workload for reproducing, locating and fixing a bug. Efficiency decrease	Untestable code. Unprecise error handling. Non optimal statement, Query.	500
Medium	Minor additional workload for implementing a new feature. No reliability impact on production. Impact is on ownership cost	Ununderstandable code. Coupled code	50
Low	No significant additional workload incurred by delivering a non remediation. Can just be fixed later	Bad indentation. Naming rule violation	1

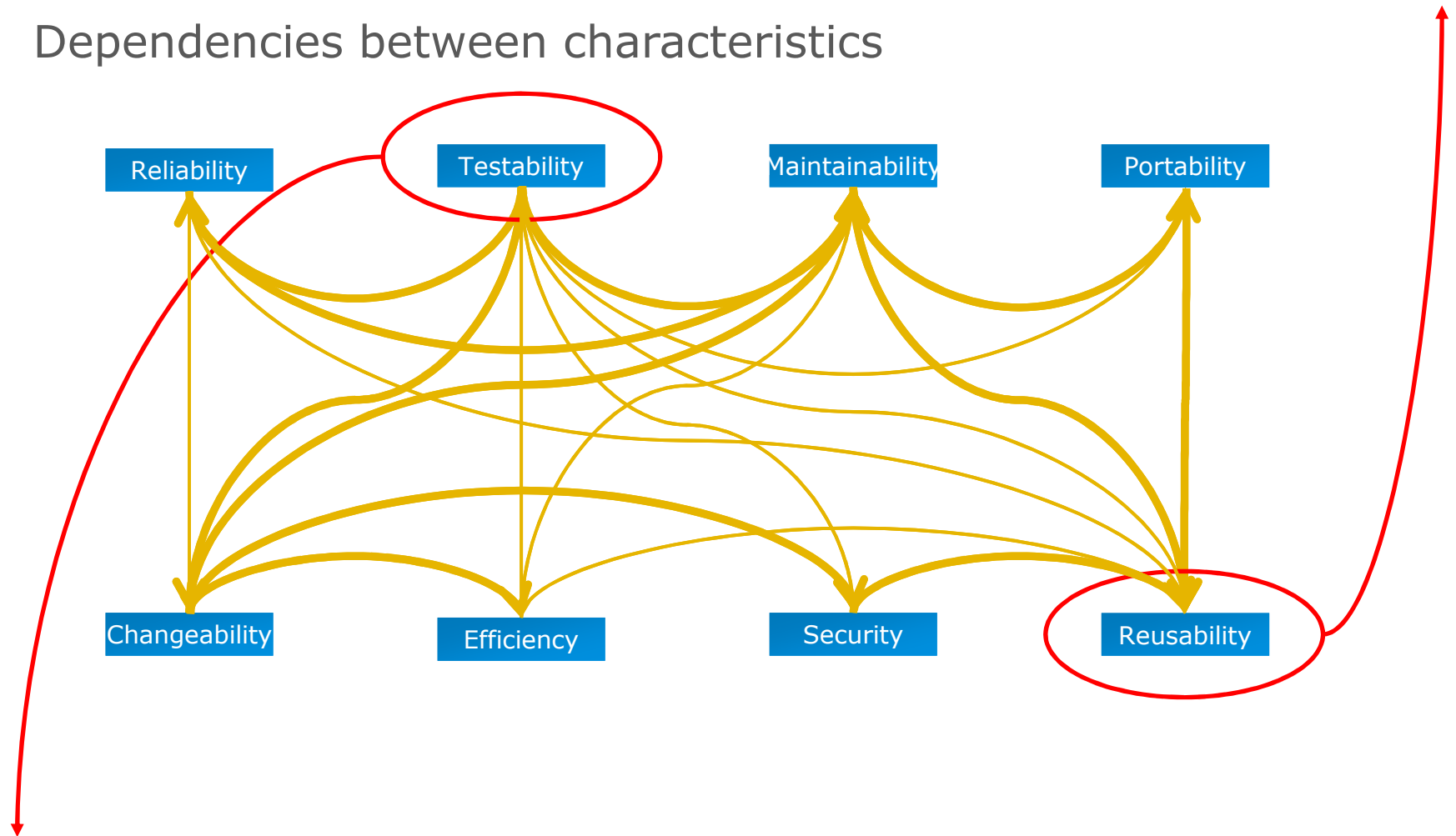
THANK YOU

QUESTIONS

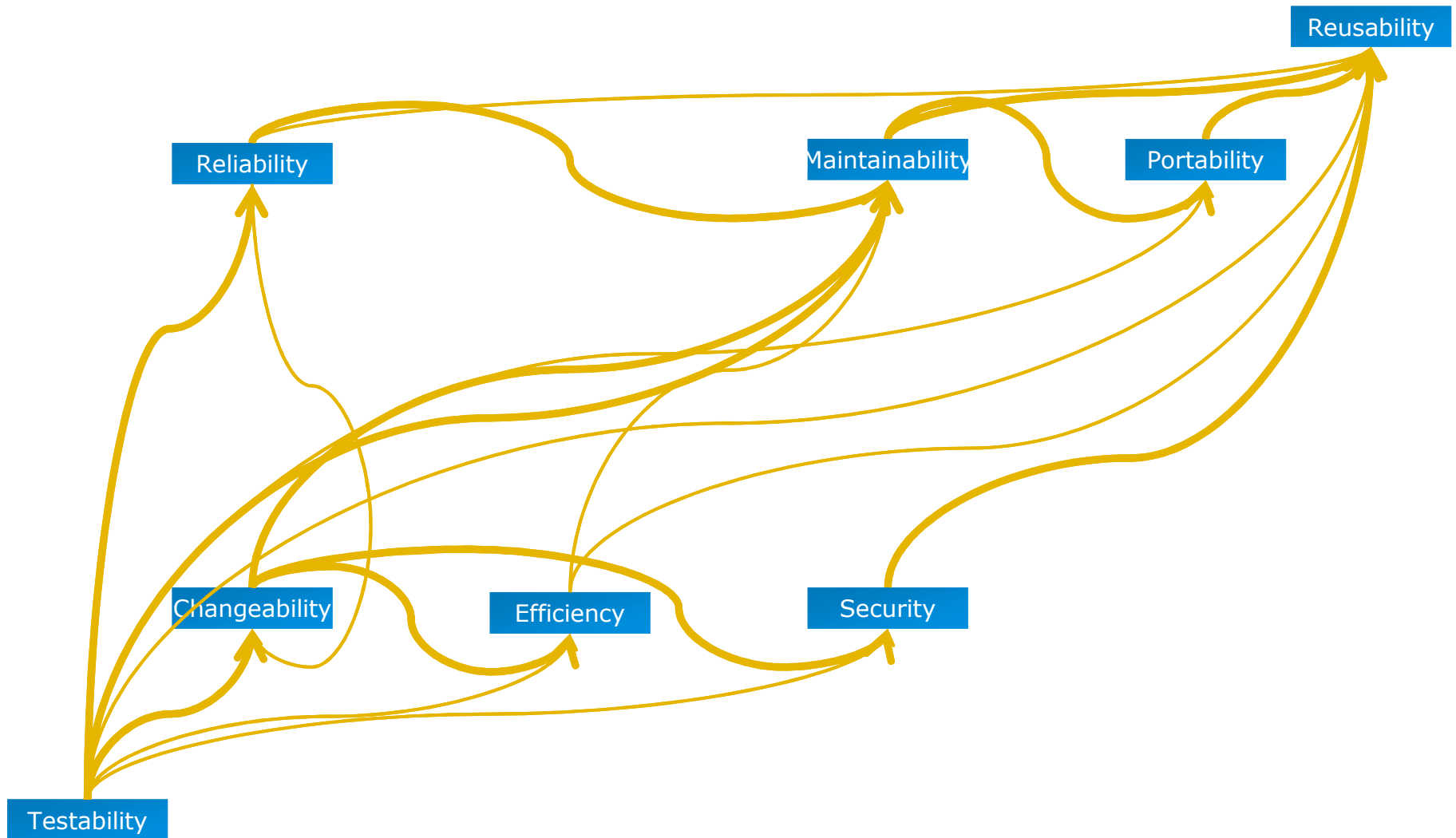
Addendum

SQALE Quality Model: Order of the Characteristics

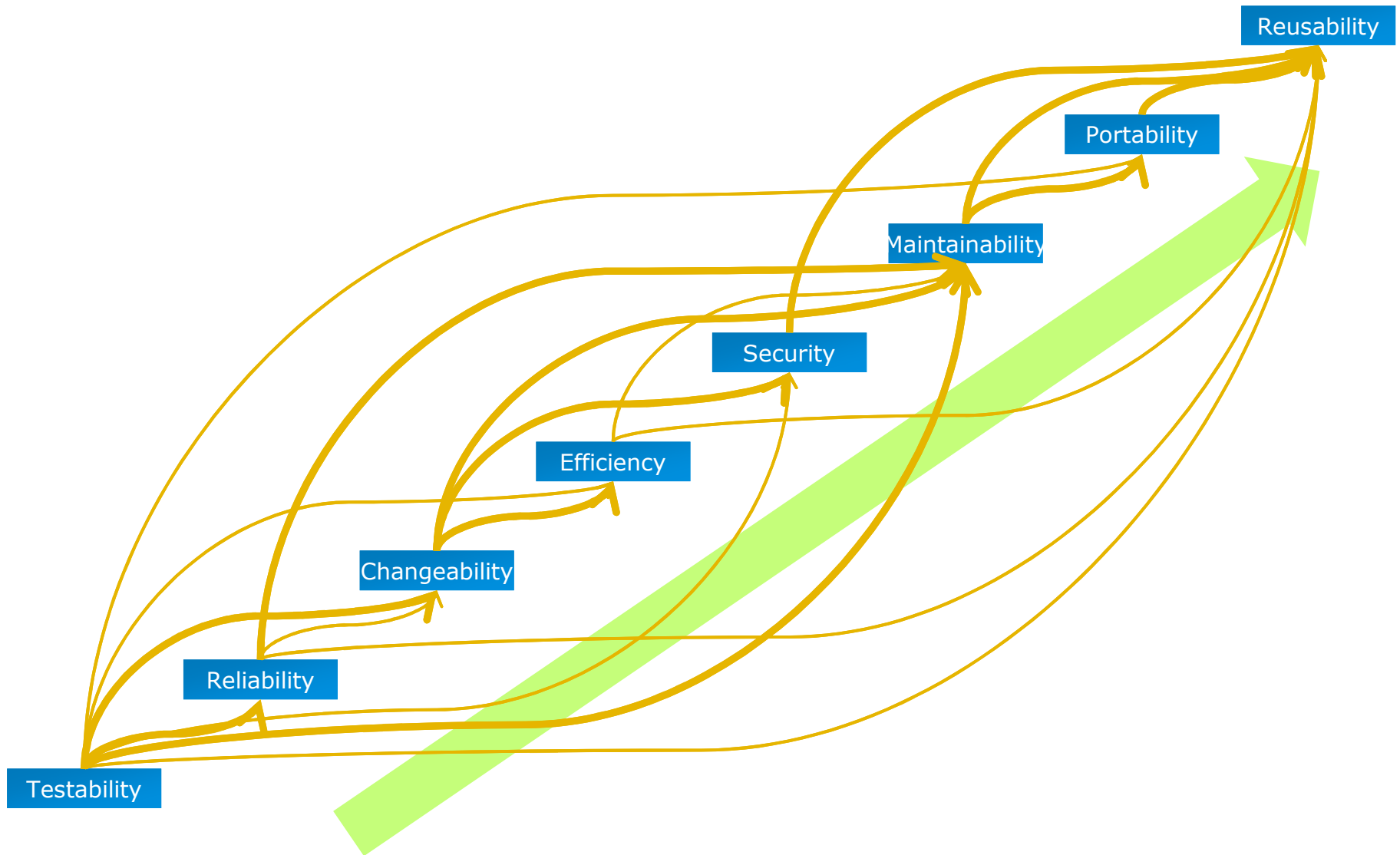
- Dependencies between characteristics



SQALE Quality Model: Order of the Characteristics

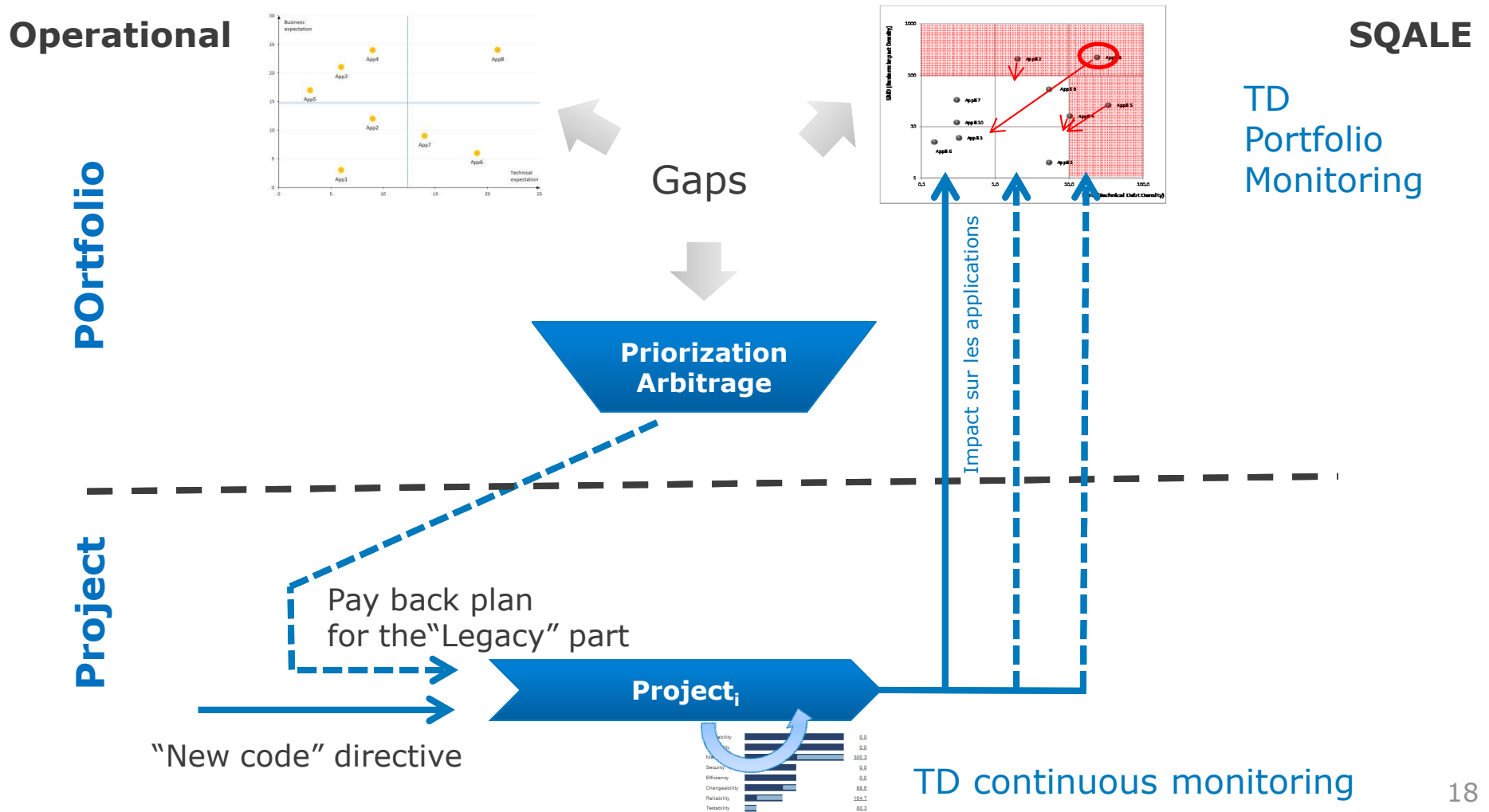


SQALE Quality Model: Order of the Characteristics



Portfolio management with SQALE

- A 2 levels and 2 speeds system



Screenshots -1

Navigation: [Dashboards](#) [Projects](#) [Measures](#) [Issues](#) [Quality Profiles](#) front

inspearit SQALE Audit Center [Manage dashboards](#)

Projects

A	Name	Version	LOCs	SQALE Rating	Last Analysis	Blocker issues <small>Δ last</small>	Business value	Maintainance level	Techno
★	Bear	2.1	3,624		05 May 2014	0	7,000.0	8.0	C#
★	Camel	3.7	8,134		05 May 2014	0	80.0	4.0	Javascript
★	Dolphin	4.0.0	14,728		05 May 2014	0	300.0	15.0	C#
★	Eagle	15.3	108,818		05 May 2014	-129	1,000.0	27.0	Java
★	Panther	1.0.3	30,571		05 May 2014	0	1,800.0	12.0	Java
★	Seagull	4.1.0	22,254		05 May 2014	0	800.0	7.0	C#
★	Tiger	2.4	5,867		05 May 2014	-12	10,000.0	32.0	Java
★	Whale	2.3	156,184		05 May 2014	-14	6,000.0	38.0	Java/Javascript

8 results

sonarqube

SonarQube™ technology is powered by [SonarSource SA](#)
Version 4.2 - [Community](#) - [Documentation](#) - [Get Support](#) - [Plugins](#)

Screenshots - 2



SonarQube™ technology is powered by [SonarSource SA](#)
Version 4.2 - [Community](#) - [Documentation](#) - [Get Support](#) - [Plugins](#)

Screenshots - 3

Dashboards Projects Measures Issues Quality Profiles front

Bear

SQALE

SQALE Détails-1

SQALE Détails-2

Dashboard

Hotspots

Issues

Time Machine

TOOLS

Components

Issues Drilldown

Design

Libraries

Clouds

Compare

insparit SQALE Audit Center

Version 2.1 - 05 May 2014 14:45 Time changes...

SQALE Rating

A

Technical Debt

1.1 days

Lines of Code

3K

Useless Duplicated Code Percentage

0.55 %

Business Impact

3,505

Business Impact Density

967.16 per KLoc

SQALE History

Apr 07, 2014

Total: 1.1

- Testability: 0
- Reliability: 0.2
- Changeability: 0.1

Technical Debt Pyramid

	Technical Debt	Total
Reusability	0.0	1.1
Portability	0.0	1.1
Maintainability	0.8	1.1
Security	0.0	0.3
Efficiency	0.0	0.3
Changeability	0.1	0.3
Reliability	0.2	0.2
Testability	0.0	0.0

SQALE Remediation Costs to reduce risk

	Cost	Total
! Blocker	0.0	0.0
+ Critical	0.2	0.2
^ Major	0.1	0.3
✓ Minor	0.8	1.1
+ Info	0.0	1.1

File Distribution by SQALE Rating

SQALE Sunburst

Depth: 3

Screenshots - 4



SonarQube™ technology is powered by SonarSource SA
Version 4.2 - [Community](#) - [Documentation](#) - [Get Support](#) - [Plugins](#)