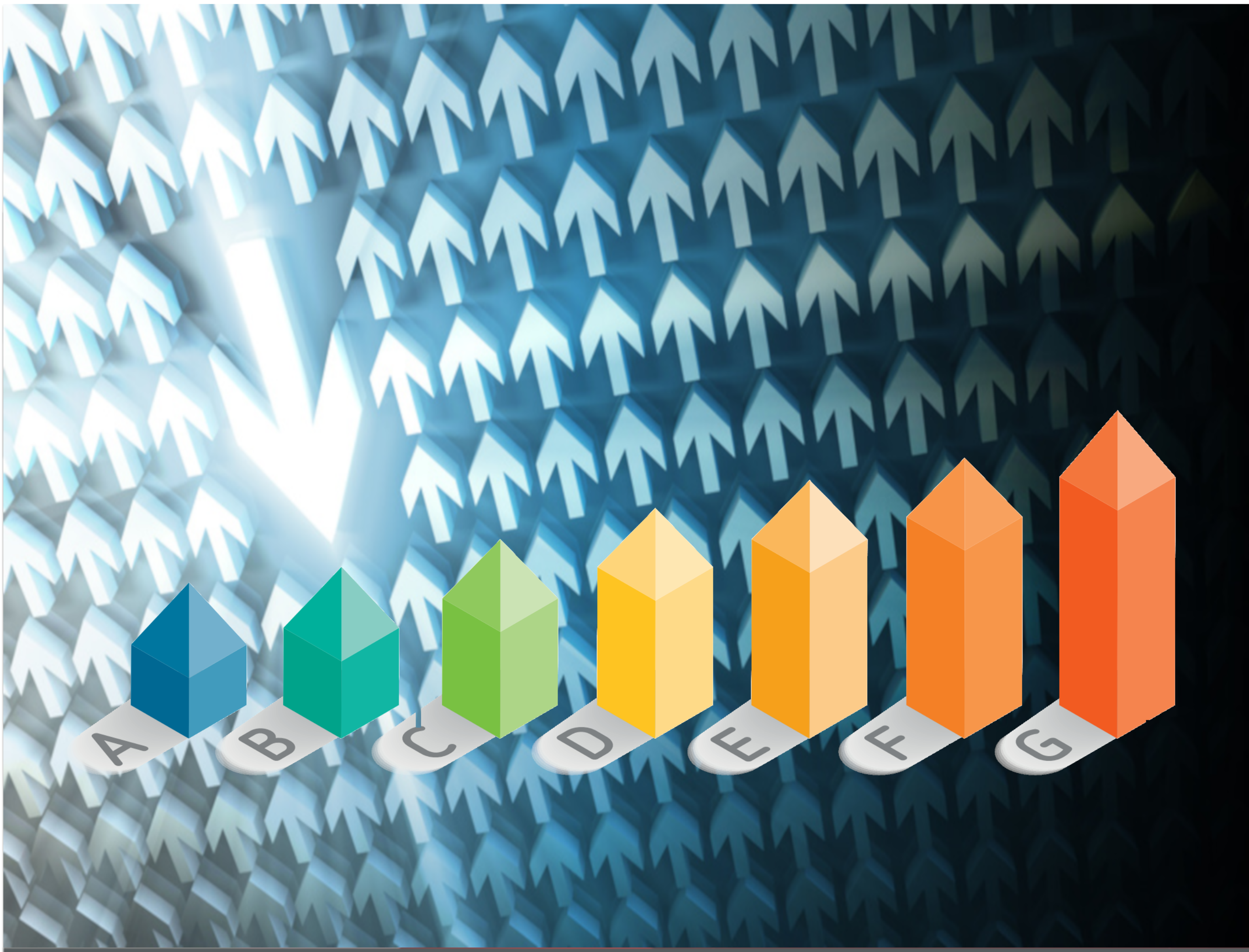


Getting Started Guide

Squore 20.0.12

Last updated 2021-07-08

Table of Contents

Preface

Foreword

This edition of the Getting Started Guide was released by Vector Informatik GmbH.

It is part of the user documentation of the Squore software product edited and distributed by Vector Informatik GmbH.

For information on how to use and configure Squore, the full suite of manuals includes:

User Manual	Target Audience
Squore Installation Checklist	New users before their first installation
Squore Installation and Administration Guide	IT personnel and Squore administrators
Squore Getting Started Guide	End users, new users wanting to discover Squore features
Squore Command Line Interface	Continuous Integration Managers
Squore Configuration Guide	Squore configuration maintainers, Quality Assurance personnel
Squore Eclipse Plugin Guide	Eclipse IDE users
Squore Reference Manual	End Users, Squore configuration maintainers
Squore API Guide	End Users, Continuous Integration Managers
Squore Software Analytics Handbook	End Users, Quality Assurance personnel

> You can also use the online help from any page when using the Squore web interface by clicking **? > Help**.

Licence

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of the copyright owner, Vector Informatik GmbH. Vector Informatik GmbH reserves the right to revise this publication and to make changes from time to time without obligation to notify authorised users of such changes. Consult Vector Informatik GmbH to determine whether any such changes have been made. The terms and conditions governing the licensing of Vector Informatik GmbH software consist solely of those set forth in the written contracts between Vector Informatik GmbH and its customers. All third-party products are trademarks or registered trademarks of their respective companies.

Warranty

Vector Informatik GmbH makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Vector Informatik GmbH shall not be liable for errors contained herein nor for incidental or consequential damages in connection with the furnishing, performance or use of this material.

This edition of the Getting Started Guide applies to Squore 20.0.12 and to all subsequent releases and modifications until otherwise indicated in new editions.

Responsabilities

Approval of this version of the document and any further updates are the responsibility of Vector Informatik GmbH.

Contacting Vector Informatik GmbH Product Support

If the information provided in this manual is erroneous or inaccurate, or if you encounter problems during your installation, contact Vector Informatik GmbH Product Support: https://portal.vector.com/

You will need a valid customer account to submit a support request. You can create an account on the support website if you do not have one already.

For any communication:

- **support@vector.com**
- **Vector Informatik GmbH Product Support**

  Vector Informatik GmbH - Holderäckerstr. 36 / 70499 Stuttgart - Germany

Getting the Latest Version of this Manual

The version of this manual included in your Squore installation may have been updated. If you would like to check for updated user guides, consult the Vector Informatik GmbH documentation site to consult or download the latest Squore manuals at https://support.squoring.com/documentation/latest. Manuals are constantly updated and published as soon as they are available.

Chapter 1. Introduction

This document is the Getting Started Guide for Squore.

It is indented as a follow up to the Squore Installation and Administration Guide and will help you understand how to use the Squore user interface to create and update projects. It is divided into several chapters, as detailed below:

- The Tools at Your Disposal provides details on where to find the sample Squore projects.
- Accessing Squore will guide you through your first access to Squore as a user.
- Creating Projects and Versions covers ways of creating new projects and versions.
- Understanding Analysis Results describes the user interface and functionality you will use in Squore on a daily basis.
- Managing Your To-Do List With Squore helps you integrate action items suggested by Squore into your workflow.
- Going Beyond Source Code shows how you can work with artefacts that are not source code.
- Track Your Favourite Indicators shows how you can track your favourite items and consult Squore results on mobile devices.
- Focus on Your Milestones guides you through the introduction and management of milestones and objectives in Squore.
- Communicating With Squore covers all reporting features of Squore.
- Keep it Tidy: Project Maintenance in Squore helps you maintain a Squore installation.
- Repository Connectors and Data Providers detail the various Repository Connectors and Data Providers you can use when launching analyses.

Chapter 2. The Tools at Your Disposal

Default Users and Sample Projects

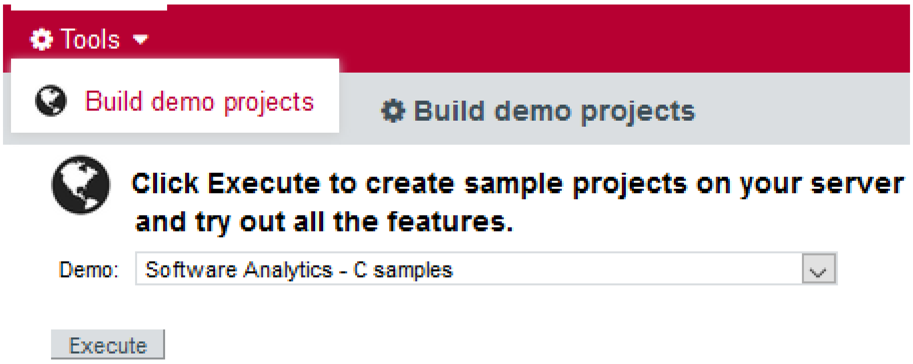
Squore ships with a collection of sample projects that we will refer to throughout this guide. Each project consists of one or several versions of the source code of an application. The code can be found in Squore Server and Squore CLI in the folder **<SQUORE_HOME>/samples**. If you do not have access to the sample projects, contact your Squore administrator to obtain a copy of the code.

Squore ships with a database that contains two sample users that you can use to familiarise yourself with all the functionality available:

- **admin/admin** is the default user that can manage the server installation, reload the server configuration after changes and perform access management tasks for the Squore installation.
- **demo/demo** is the default Squore power user that can create, review and manage projects, as well as give team members visibility or management privileges on the projects he himself manages.

You can use these two default users, but we recommend that you change their passwords after your first connection. The privileges and permissions assigned to these default users can be modified as needed. You can familiarise yourself with Squore permissions and privileges by referring to Understanding Roles.

You may choose to read this manual from beginning to end or jump straight to a specific topic. Logging in as the **demo** user gives you access to a **Tools** menu that allows reproducing the examples shown in this manual. Click **Tools > Build Demo Projects** and select the **Software Analytics - C samples** to get started.



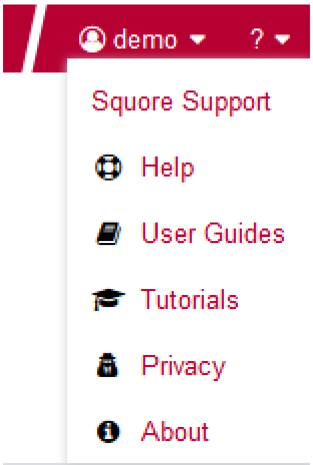
Click Execute to create sample projects on your server and try out all the features.

Demo: Software Analytics - C samples

The Tools menu allows you to create sample projects using the Software Analytics - C samples option

The menu is only accessible to the user **demo** or any user who belongs to a group called **demo**. Contact your Squore administrator if you do not have access to the Tools to create the sample projects.

Getting More Help



The Help menu

If at any moment you have doubts about how a feature works, Squore offers help in HTML and PDF formats. A Wiki and support site is also available.

Online Help

The Squore online help can be accessed from anywhere in Squore by clicking on the **? > Help** menu entry.

The online help is contextual and provides information in a popup window about the page that you are currently viewing in Squore.

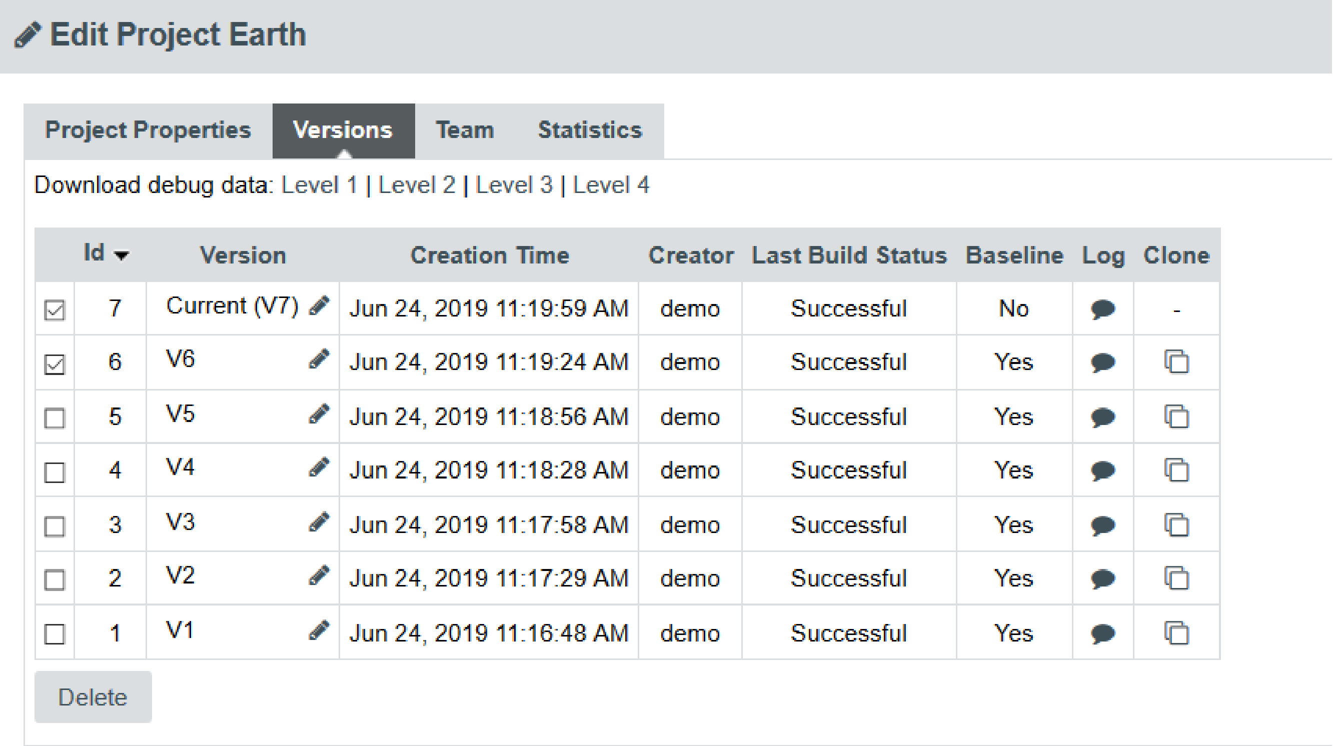
User Guides and Support Wiki

The Squore user guides are available in PDF and HTML format by clicking the **? > User Guides** menu entry in Squore. You can download a copy for offline use.

The Squore Support Wiki provides release notes, known issues and hints and tips for current and past Squore versions. Visit http://openwiki.squoring.com for more information.

Review Log Files and Download Debug Data

Every **owner** or **Project Manager** of a project can retrieve the analysis log files for their projects without the need to consult an administrator. This is done by accessing the **Manage** page for a particular project and viewing the **Versions** tab (**Projects page > Manage icon > Versions tab**) as shown below:



Download debug data: Level 1 | Level 2 | Level 3 | Level 4

	Id ▾	Version	Creation Time	Creator	Last Build Status	Baseline	Log	Clone
☑	7	Current (V7) ✎	Jun 24, 2019 11:19:59 AM	demo	Successful	No	💬	-
☑	6	V6 ✎	Jun 24, 2019 11:19:24 AM	demo	Successful	Yes	💬	⧉
☐	5	V5 ✎	Jun 24, 2019 11:18:56 AM	demo	Successful	Yes	💬	⧉
☐	4	V4 ✎	Jun 24, 2019 11:18:28 AM	demo	Successful	Yes	💬	⧉
☐	3	V3 ✎	Jun 24, 2019 11:17:58 AM	demo	Successful	Yes	💬	⧉
☐	2	V2 ✎	Jun 24, 2019 11:17:29 AM	demo	Successful	Yes	💬	⧉
☐	1	V1 ✎	Jun 24, 2019 11:16:48 AM	demo	Successful	Yes	💬	⧉

The Versions tab provides access to log files and allows to Download Debug Data

Clicking the **Log** icon opens a page showing the project's client and server logs for that analysis, as well as configuration and output files will open in a new browser tab.

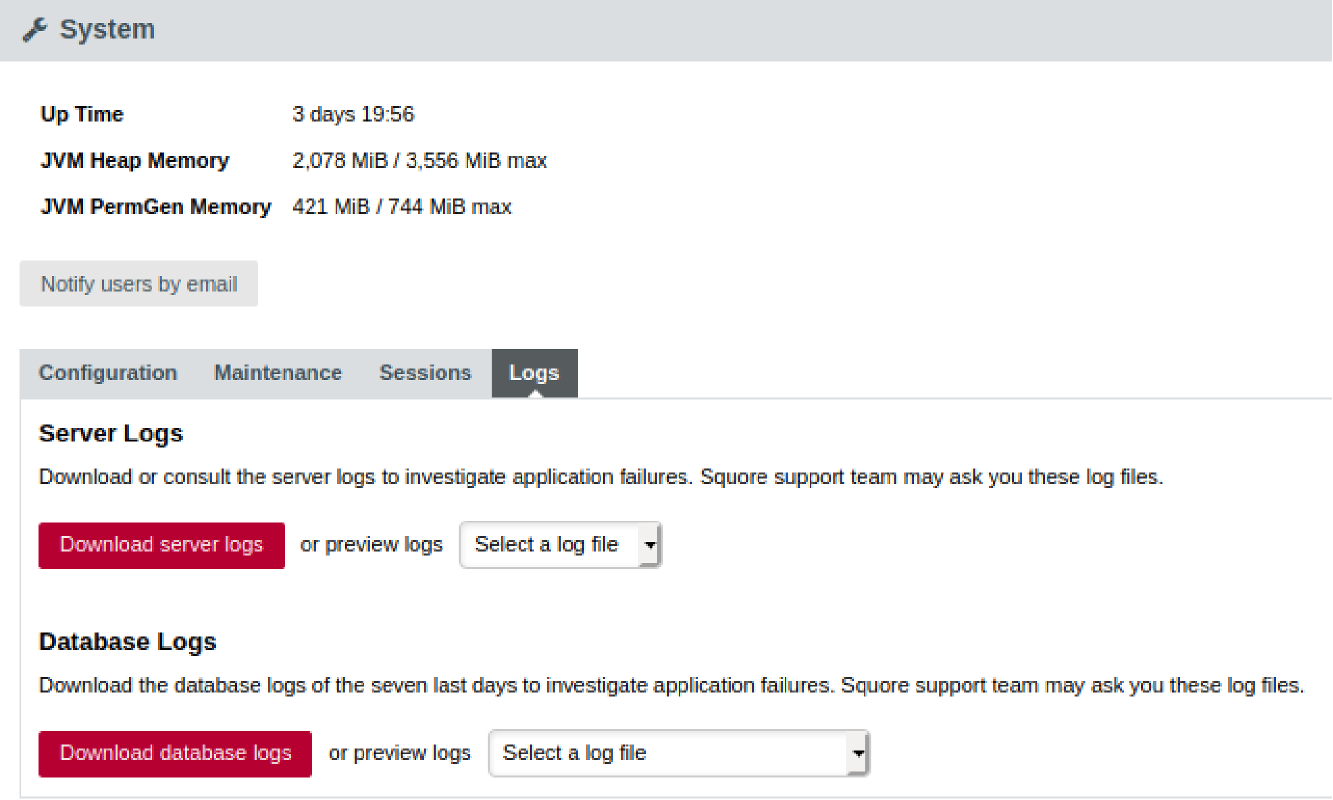
Clicking the one of the **Download Debug Data** links above the versions table downloads a zip file of the logs and project data that can be further analysed to understand problems during analysis. The lower levels include only data related to the latest analysis, while the higher levels include information related to the history of the entire project.

A debug data zip file contains a collection of logs, temporary and output files for each one or more versions of the project. Each version folder can contain the following items:

- A *DataProviders* folder containing the output files generated by each Data Provider run during the analysis.
- A *[DataProviderName].log* file for each Data Provider included in the analysis.
- A *conf.xml* file summarising the project parameters used for the analysis.
- A *output.xml* file containing the output information requested with the --filter parameter during the analysis.
- A *build.log* file containing the information relative to actions carried out on the server during the analysis.
- A *build_client.log* file containing the information relative to actions carried out on the client during the analysis.
- A *excluded.log* file containing the list of all files not included in the analysis and the reason for their exclusion. Note that this file is only generated if some files were excluded.
- A *table.md5* file containing state information about the analysed source code, if any.
- A *storage* folder containing information about the analysed source code, if any.

> If you do not want to download the entire debug package, note that the main log files can also be downloaded individually from the Projects page by clicking on the project status label.

In order to investigate application failures (rather than project analysis errors), Squore administrators have the possibility to extract and download Squore Logs file created by the application. You can access the Logs if you have administrator privileges by clicking **Administration > System > Logs tab** in the toolbar after logging in.



System

Up Time	3 days 19:56
JVM Heap Memory	2,078 MiB / 3,556 MiB max
JVM PermGen Memory	421 MiB / 744 MiB max

Server Logs

Download or consult the server logs to investigate application failures. Squore support team may ask you these log files.

Database Logs

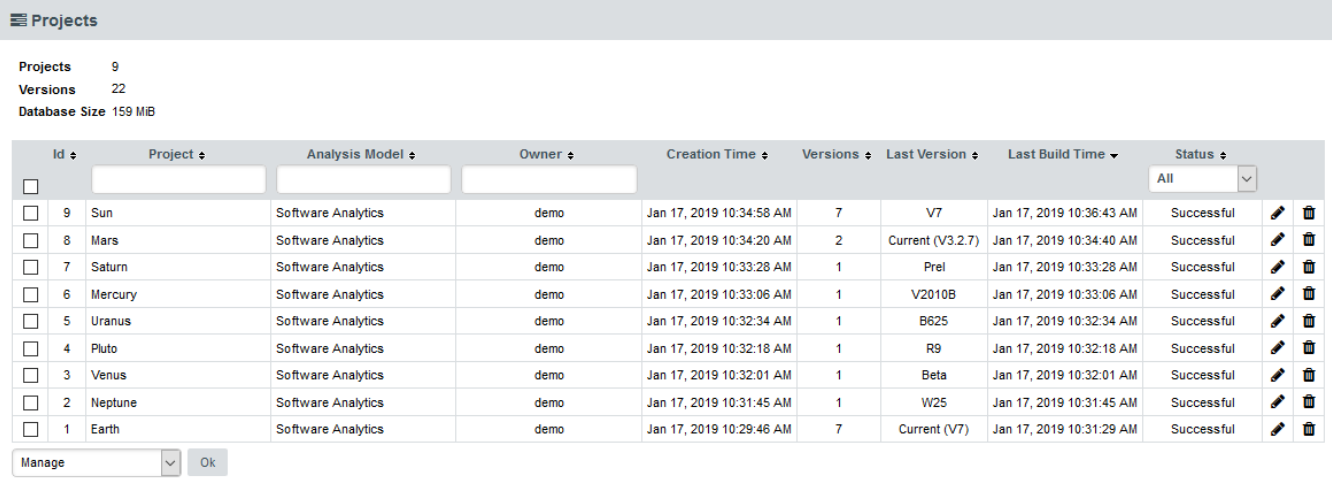
Download the database logs of the seven last days to investigate application failures. Squore support team may ask you these log files.

The administration system logs tab for administrators

In the Logs tab, administrators have two kind of logs available:

- **Server Logs**: extract and download into a .zip file the latest server log file created by the application. It is also possible to consult the server log in the UI.
- **Database Logs**: extract and download into a .zip file the last week database logs.

Administrators can also get debug information and manage any project created on the server by clicking **Administration > Projects**, which provides a detailed view of all projects created on Squore Server, on a summary page shown below.



Projects

Projects	9
Versions	22
Database Size	159 MiB

	Id	Project	Analysis Model	Owner	Creation Time	Versions	Last Version	Last Build Time	Status		
□									All		
□	9	Sun	Software Analytics	demo	Jan 17, 2019 10:34:58 AM	7	V7	Jan 17, 2019 10:36:43 AM	Successful	✎	🗑
□	8	Mars	Software Analytics	demo	Jan 17, 2019 10:34:20 AM	2	Current (V3.2.7)	Jan 17, 2019 10:34:40 AM	Successful	✎	🗑
□	7	Saturn	Software Analytics	demo	Jan 17, 2019 10:33:28 AM	1	Prel	Jan 17, 2019 10:33:28 AM	Successful	✎	🗑
□	6	Mercury	Software Analytics	demo	Jan 17, 2019 10:33:06 AM	1	V2010B	Jan 17, 2019 10:33:06 AM	Successful	✎	🗑
□	5	Uranus	Software Analytics	demo	Jan 17, 2019 10:32:34 AM	1	B625	Jan 17, 2019 10:32:34 AM	Successful	✎	🗑
□	4	Pluto	Software Analytics	demo	Jan 17, 2019 10:32:18 AM	1	R9	Jan 17, 2019 10:32:18 AM	Successful	✎	🗑
□	3	Venus	Software Analytics	demo	Jan 17, 2019 10:32:01 AM	1	Beta	Jan 17, 2019 10:32:01 AM	Successful	✎	🗑
□	2	Neptune	Software Analytics	demo	Jan 17, 2019 10:31:45 AM	1	W25	Jan 17, 2019 10:31:45 AM	Successful	✎	🗑
□	1	Earth	Software Analytics	demo	Jan 17, 2019 10:29:46 AM	7	Current (V7)	Jan 17, 2019 10:31:29 AM	Successful	✎	🗑

The project administration page for administrators

Chapter 3. Accessing Squore

This chapter walks you through your first access to Squore and covers the web interface and some ways to customise it to your liking.

Understanding Roles

Before you start working with Squore, it is essential to understand how access management works. The various permissions and privileges that can be assigned to Squore users are grouped in global roles and project roles respectively. A set of default global and project roles is available when you first start the server. You can edit them, or create more as needed.

Use this simple trick to remember the different between a global role and project role:

- A **global role** is a set of permissions granting access to certain Squore features to a user
- A **project role** is a set of privileges for a user within a Squore project.

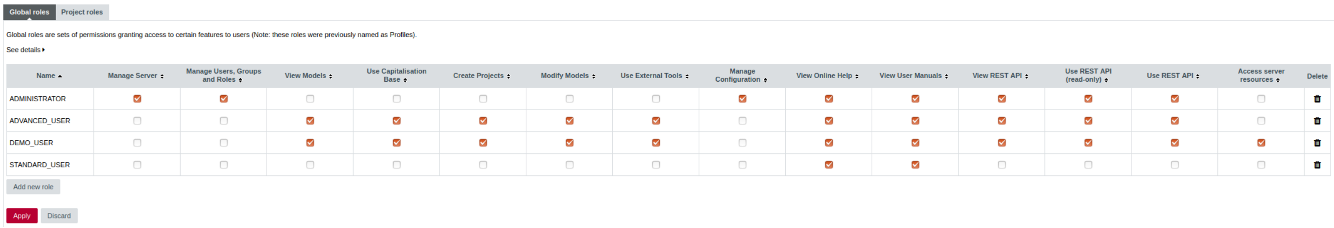
A Squore user with the global role, Administrator, can manage users as well as their global and project roles. A Squore user with the project role, Project Manager, can create a new version of this project or give access to another user to this project's analysis results.

User Global Roles

You can use global roles to grant or deny access to the following Squore features:

- **Manage Server:** Configure the server, access server logs, manage all projects.
- **Manage Users, Groups and Roles:** Complete access to user management on the server.
- **View Models:** Allows users to use the Viewer and the Validator.
- **Use Capitalisation Base:** Provides access to the Capitalisation Base feature to learn from past data in order to improve your model.
- **Create Projects:** Allows users to run analyses.
- **Modify Models:** Allows users to use the Dashboard Editor and the Analysis Model Editor, as well as view usage statistics for particular analysis models.
- **Use External Tools:** View and use external tools configured by your Squore Administrator. To learn more about this feature, consult the Configuration Guide.
- **Manage Configuration:** Allows users to reload the server configuration from disk.
- **View Online Help:** Allows users to consult the online help from the web interface.
- **View User Manuals:** Allows users to consult the product documentation from the web interface.
- **Access Server Resources:** Allows users to analyze files located on the server.

Four global roles are available by default, with permissions set as shown below:



Global roles are sets of permissions granting access to certain features to users (Note: these roles were previously named as Profiles).

See details ▸

Name ▴	Manage Server ⬍	Manage Users, Groups and Roles ⬍	View Models ⬍	Use Capitalisation Base ⬍	Create Projects ⬍	Modify Models ⬍	Use External Tools ⬍	Manage Configuration ⬍	View Online Help ⬍	View User Manuals ⬍	View REST API ⬍	Use REST API (read-only) ⬍	Use REST API ⬍	Access server resources ⬍	Delete
ADMINISTRATOR	☑	☑	□	□	□	□	□	☑	☑	☑	☑	☑	☑	□	🗑
ADVANCED_USER	□	□	☑	☑	☑	☑	☑	□	☑	☑	☑	☑	☑	□	🗑
DEMO_USER	□	□	☑	☑	☑	☑	☑	□	☑	☑	☑	☑	☑	☑	🗑
STANDARD_USER	□	□	□	□	□	□	□	□	☑	☑	□	□	□	□	🗑

Default global roles for administrators, advanced user, standard user and demo user

Note that a global role can be assigned to a user or a group of users. It is therefore possible for a user to be a member of more than one global role. In this case, the user's global role is the combination of all permissions from all the global roles they are a member of.

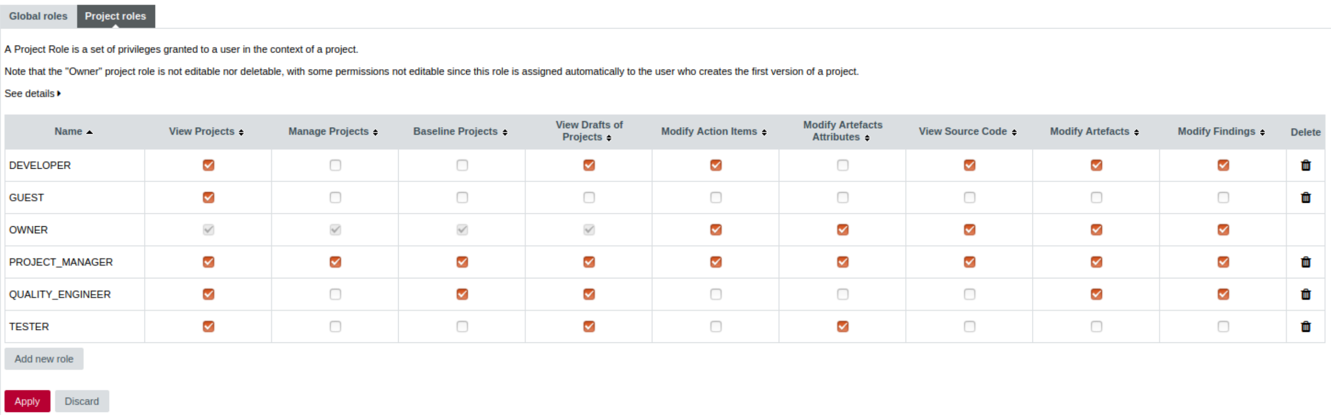
> For security purposes, the global role **DEMO_USER** should be deactivated on a production installation.

User Project Roles

A project role is the set of privileges that a user enjoys in the context of a project. You can use project roles to allow users to undertake these actions within the scope of a project:

- **View Projects:** Allows a user to see a project in their project list and to browse this project's analysis results.
- **Manage Projects:** Allows a user to manage a project: rename it, create or delete versions, access project creation log files and add other users to the project team.
- **Baseline Projects:** Allows a user to create a baseline version of a project that will not be overwritten by subsequent analysis. For more information about baselining, see Working with Draft and Baseline Versions.
- **View Drafts of Projects:** Allows a user to view the current draft version of a project. Without this privilege, only baseline versions of a project are visible in the project portfolio. For more information about baselining, see Working with Draft and Baseline Versions.
- **Modify Action Items:** Allows updating the status of Action Items from TODO to Relaxed for example. Without this privilege, the status is displayed as a read-only field.
- **Modify Artefacts Attributes:** Allows a user to modify the value of attributes displayed in the Forms tab of the Explorer. Without this privilege, attributes are read-only.
- **View Source Code:** Allows a user to click to view the source code of an artefact from any tab in the Explorer.
- **Modify Artefacts:** Allows a user to add, delete, relax, exclude artefacts from the artefact tree. Users without this privilege can still view artefacts created by others.
- **Modify Findings:** Allows user to change the status of violations on the Findings tab. Users without this privilege can view relaxed findings but cannot relax or unrelax them.

Six project roles are available by default, with privileges assigned as shown below:



A Project Role is a set of privileges granted to a user in the context of a project.

Note that the "Owner" project role is not editable nor deletable, with some permissions not editable since this role is assigned automatically to the user who creates the first version of a project.

See details ▸

Name ▲	View Projects ⬍	Manage Projects ⬍	Baseline Projects ⬍	View Drafts of Projects ⬍	Modify Action Items ⬍	Modify Artefacts Attributes ⬍	View Source Code ⬍	Modify Artefacts ⬍	Modify Findings ⬍	Delete
DEVELOPER	☑	□	□	☑	☑	□	☑	☑	☑	🗑
GUEST	☑	□	□	□	□	□	□	□	□	🗑
OWNER	☑	☑	☑	☑	☑	☑	☑	☑	☑	
PROJECT_MANAGER	☑	☑	☑	☑	☑	☑	☑	☑	☑	🗑
QUALITY_ENGINEER	☑	□	☑	☑	□	□	□	☑	☑	🗑
TESTER	☑	□	□	☑	□	☑	□	□	□	🗑

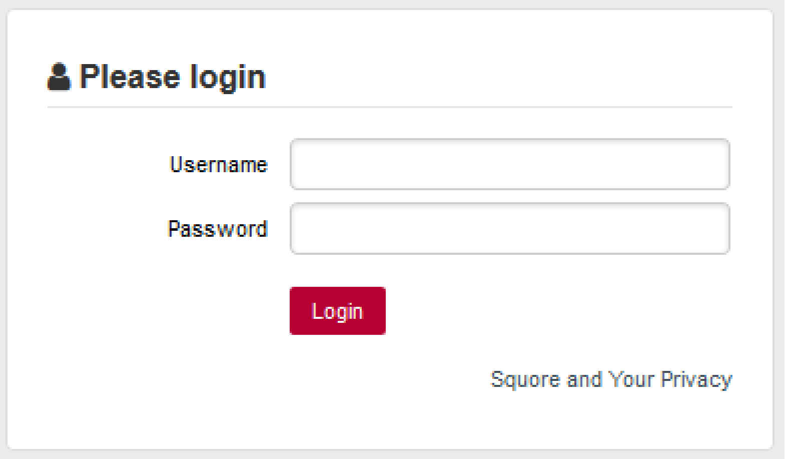
Default project roles available for users in Squore

Note that a user can have multiple project roles in a project. When you have multiple project roles in a project, you combine privileges from all the project roles that you are a member of.

The project role, **OWNER**, is assigned automatically to the user who creates the first version of a project. A project has only one owner, and you can control how much a project owner can see and do by modifying the permissions of the **OWNER** project role. An administrator can transfer ownership of a project to a new user if required.

How Do I log into Squore?

Your Squore installation runs on http://localhost:8180/SQuORE_Server by default. By accessing this page in your browser, you will be redirected to the Squore login page , as shown below:

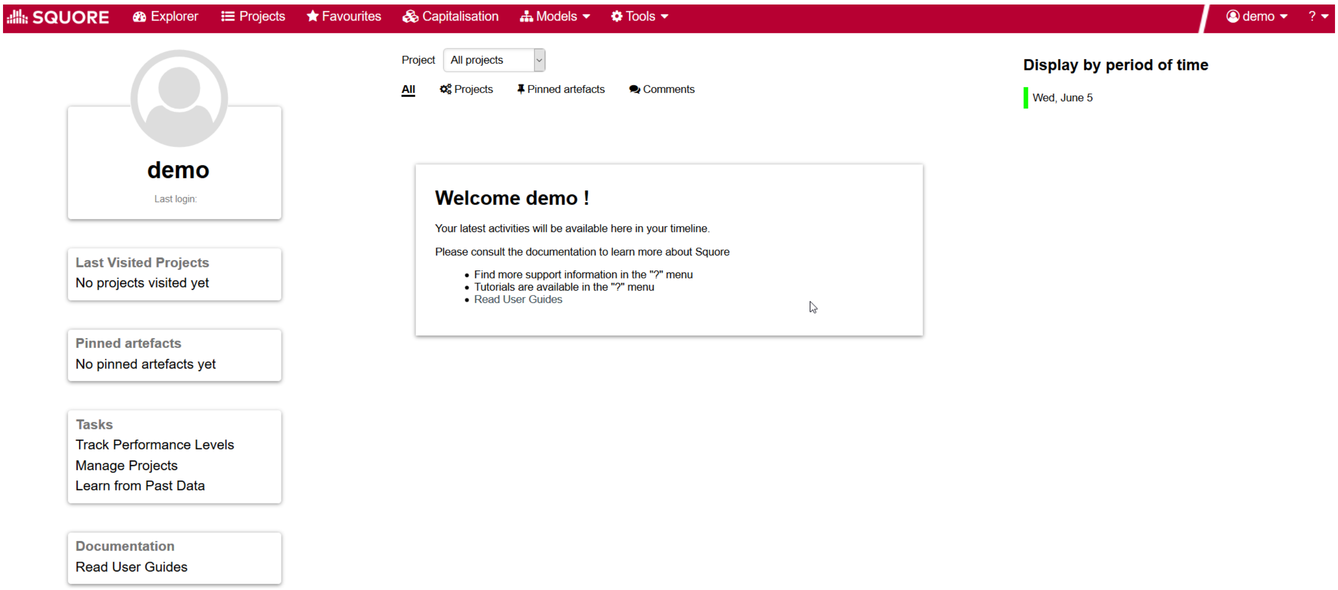


The Squore login form

Type in a username and a password and click **Login** to log in.

Where Do I Go From The Home Page?

To begin using Squore, log in as the demo user with *demo* as username and password on the login page. Click the **Login** button and wait for the Home page to open.



Welcome demo !

Your latest activities will be available here in your timeline.

Please consult the documentation to learn more about Squore

- Find more support information in the "?" menu
- Tutorials are available in the "?" menu
- Read User Guides

The Squore Home page

From the Home page, you can automatically return to the last projects or favourite artefacts that you had opened in the Explorer before logging out. A timeline of executed builds and news mechanism is available as well. You can also get links to the help and other features available for your account.

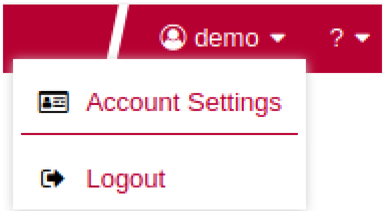
As the demo user, you are an advanced user of Squore and have access to the following functionality from the toolbar :

- **Explorer**, where you can review your analysis results.
- **Projects**, where you can launch new analyses and manage your projects.
- **Favourites**, where you can view and manage your favourite charts across projects.
- **Capitalisation**, where aggregated statistical data can be found.
- **Models**, under which you can examine all characteristics of your model and edit your dashboards.
- **Tools**, which contains shortcuts to scripts that recreate demo projects. Note that only the *demo* user and members of the *demo* group have access to this menu by default.
- **<username>**, where you can set your preferences and log out from Squore.
- **?**, where online help, user manuals and application information can be found.

> If you are looking for the administration tools, log in as an administrator of Squore using *admin* as the username and password. You will gain access to the **Administration** menu where you can configure access management and administer the server.

How Do I log out of Squore?

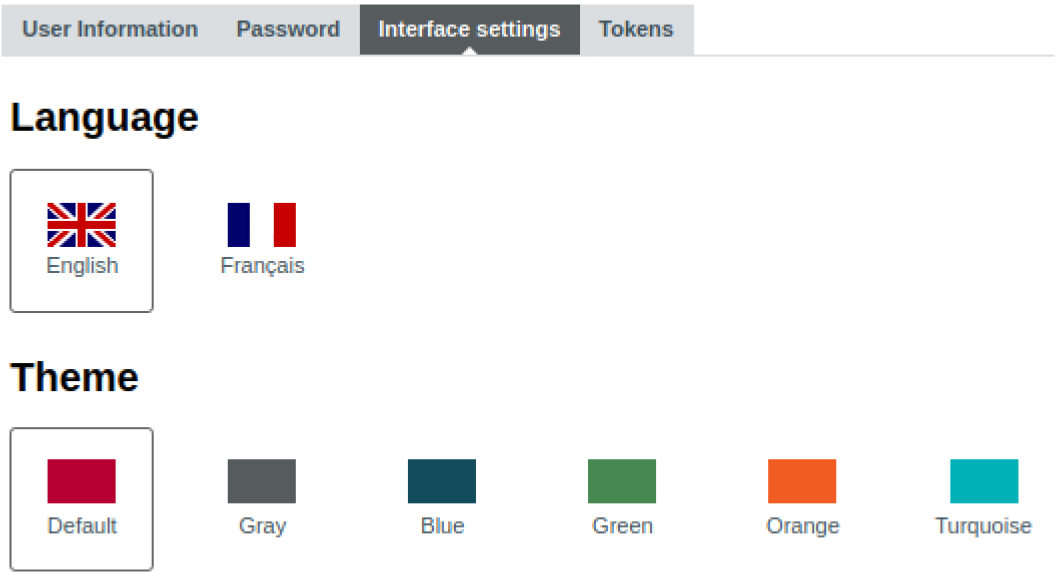
You can log out of Squore by clicking your user name in the menu bar and selecting the **Logout** option. Note that if you close your browser without logging out, your session will automatically time out after two hours.



The Logout entry in the user menu

Can I Tweak the Squore Look and Feel?

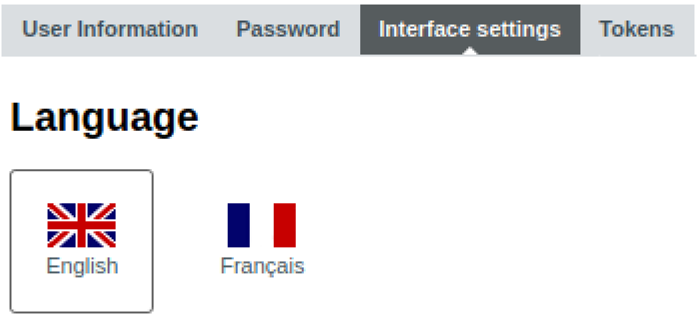
Using a Different Theme



Theme selection in the user menu

The Squore look and feel can be adapted to your liking, with six provided themes. They are accessible by clicking on your **<username>** in the menu bar, and then **Account Settings > Interface settings**. Select one of the available colour schemes to change the color of the interface. Your changes are saved using a browser cookie.

User Interface Language



Language selection in the user menu

You can use Squore in various languages. English and French are provided by default, and your Squore administrator can add more as needed. If you want to change the language of the Squore user interface, click on your **<username>** in the menu bar, then **Account Settings > Interface settings**, and select one of the flags available. The changes are applied immediately and your preferences are saved even after you log out.

Chapter 4. Creating Projects and Versions

In this chapter, you will learn about the various ways to create a project in Squore: using the UI, using a command line tool or triggering analyses in a continuous integration environment.

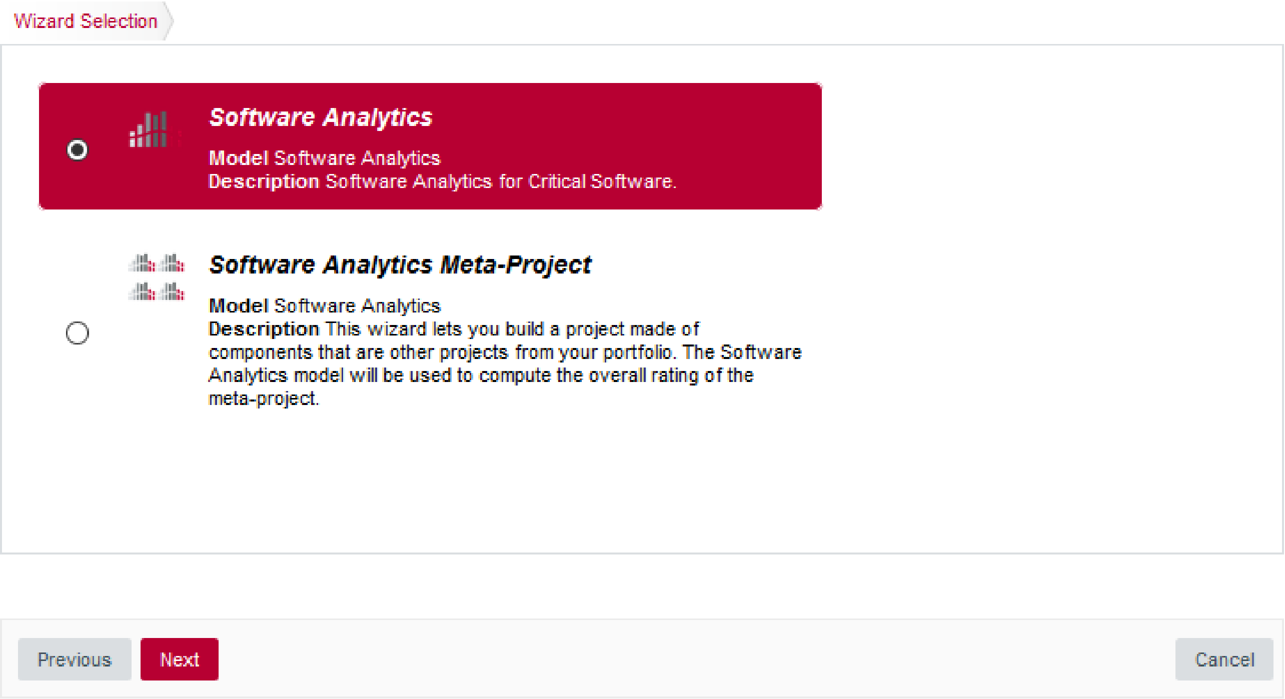
How Do I Create a Project in Squore?

Creating a project in Squore is as easy as following a wizard that will prompt you for information about the source material to analyse, and the external Data Providers to add to the analysis results.

The example below assumes that the source code for the sample project used is available on a network share. The path to the source files to analyse is relative to the server.

In order to create a project for the sample application Neptune2, follow these steps:

1. Access http://localhost:8180/SQuORE_Server in your browser. The log-in page appears.
2. Log in as the demo user with the login/password combination *demo/demo*.
3. Click the **Login** button. You are presented with the Squore Home page.
4. Click **Projects** to switch to the projects view and click **Create Project** to create the Neptune2 project.
5. The **Wizard Selection** screen appears. Project wizards allow you to use different analysis models and tools to analyse your projects. For this demo, click the **Software Analytics** wizard to start creating the project.



● ***Software Analytics***

Model Software Analytics
Description Software Analytics for Critical Software.

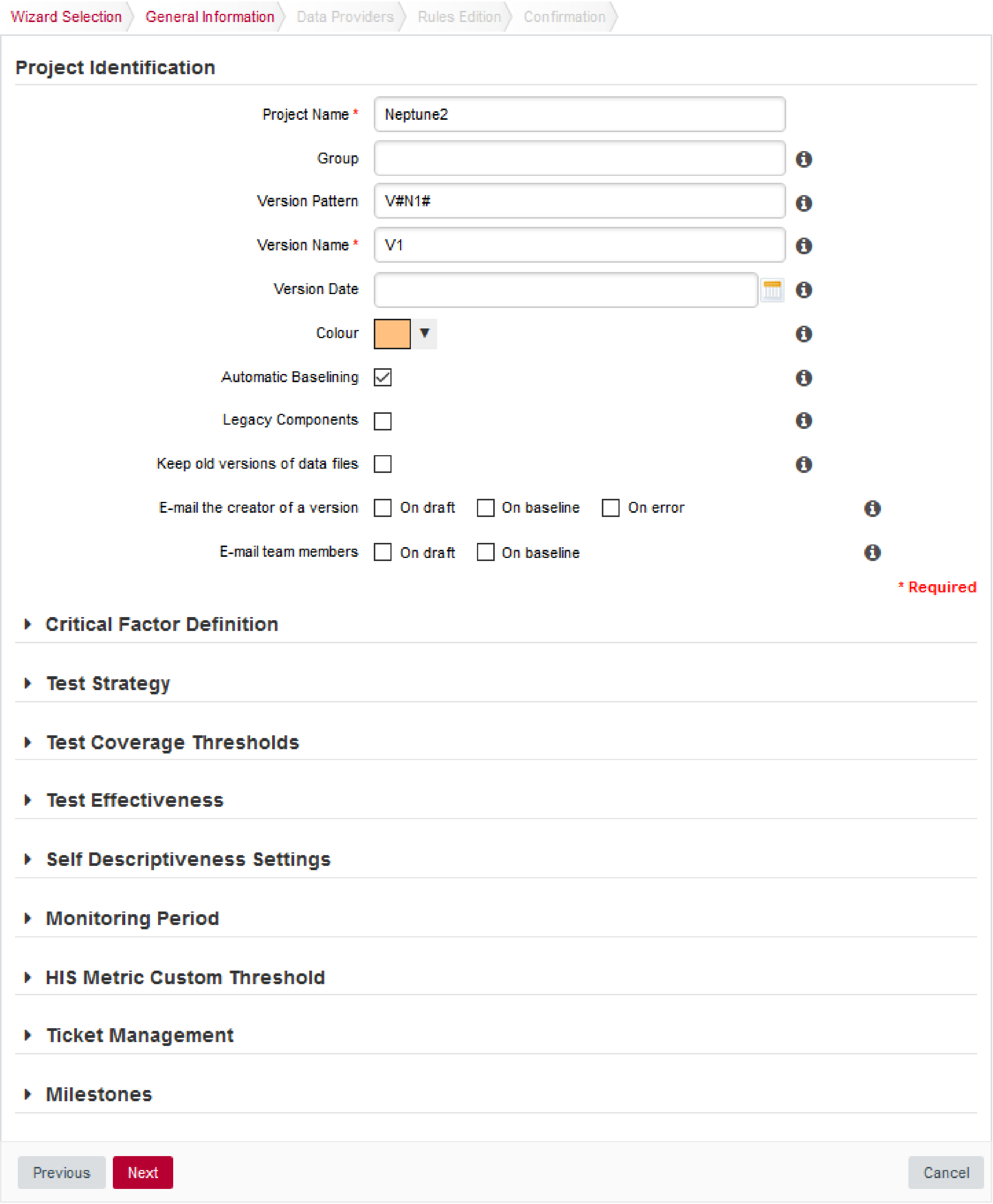
○ ***Software Analytics Meta-Project***

Model Software Analytics
Description This wizard lets you build a project made of components that are other projects from your portfolio. The Software Analytics model will be used to compute the overall rating of the meta-project.

The Wizard Selection screen

If your Squore installation has been customised to provide only one project wizard, the **Wizard Selection** is skipped.

6. The **General Information** screen appears.



Wizard Selection > General Information > Data Providers > Rules Edition > Confirmation

Project Identification

Project Name *	Neptune2
Group	
Version Pattern	V#N1#
Version Name *	V1
Version Date	
Colour	🟧 ▼
Automatic Baselining	☑
Legacy Components	□
Keep old versions of data files	□
E-mail the creator of a version	□ On draft □ On baseline □ On error
E-mail team members	□ On draft □ On baseline

* Required

- Critical Factor Definition
- Test Strategy
- Test Coverage Thresholds
- Test Effectiveness
- Self Descriptiveness Settings
- Monitoring Period
- HIS Metric Custom Threshold
- Ticket Management
- Milestones

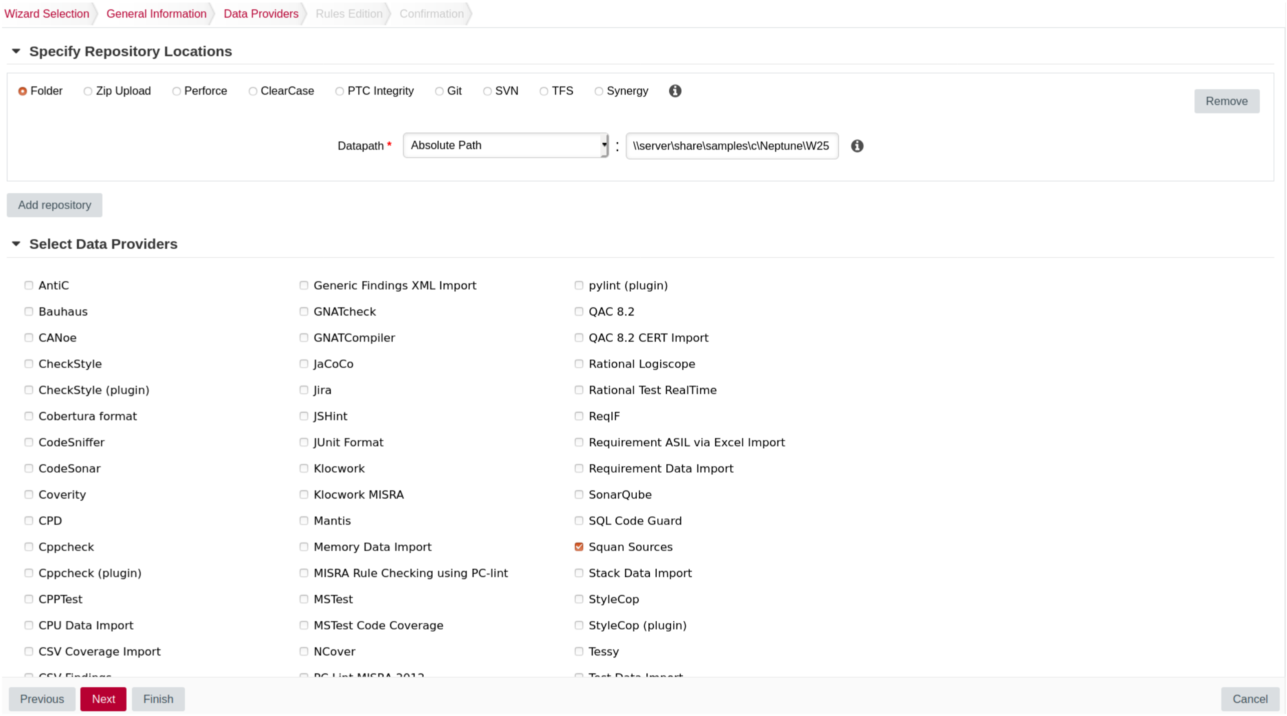
The General Information screen

On this screen, you can enter the information relative to your project in the **Project Identification** section

The **Version Date** field allows specifying a custom date for the analysis, so that different analyses can be placed correctly on a timeline later for certain charts in the dashboard. If you leave it empty, then the actual time at which you are running the analysis is used.

The Software Analytics model offers extra parameters below the Project Identification section, but you can ignore them for now.

7. Click the Next button. The **Data Providers** screen is shown:



Wizard Selection › General Information › Data Providers › Rules Edition › Confirmation

Specify Repository Locations

● Folder ○ Zip Upload ○ Perforce ○ ClearCase ○ PTC Integrity ○ Git ○ SVN ○ TFS ○ Synergy

Remove

Datapath * Absolute Path : \\server\share\samples\c\Neptune\W25

Add repository

Select Data Providers

- □ AntiC
- □ Bauhaus
- □ CANoe
- □ CheckStyle
- □ CheckStyle (plugin)
- □ Cobertura format
- □ CodeSniffer
- □ CodeSonar
- □ Coverity
- □ CPD
- □ Cppcheck
- □ Cppcheck (plugin)
- □ CPPTest
- □ CPU Data Import
- □ CSV Coverage Import
- □ CSV Findings
- □ Generic Findings XML Import
- □ GNATcheck
- □ GNATCompiler
- □ JaCoCo
- □ Jira
- □ JSHint
- □ JUnit Format
- □ Klocwork
- □ Klocwork MISRA
- □ Mantis
- □ Memory Data Import
- □ MISRA Rule Checking using PC-lint
- □ MSTest
- □ MSTest Code Coverage
- □ NCover
- □ PC Lint MISRA 2012
- □ pylint (plugin)
- □ QAC 8.2
- □ QAC 8.2 CERT Import
- □ Rational Logiscope
- □ Rational Test RealTime
- □ ReqIF
- □ Requirement ASIL via Excel Import
- □ Requirement Data Import
- □ SonarQube
- □ SQL Code Guard
- ☑ Squan Sources
- □ Stack Data Import
- □ StyleCop
- □ StyleCop (plugin)
- □ Tessy
- □ Test Data Import

Previous Next Finish Cancel

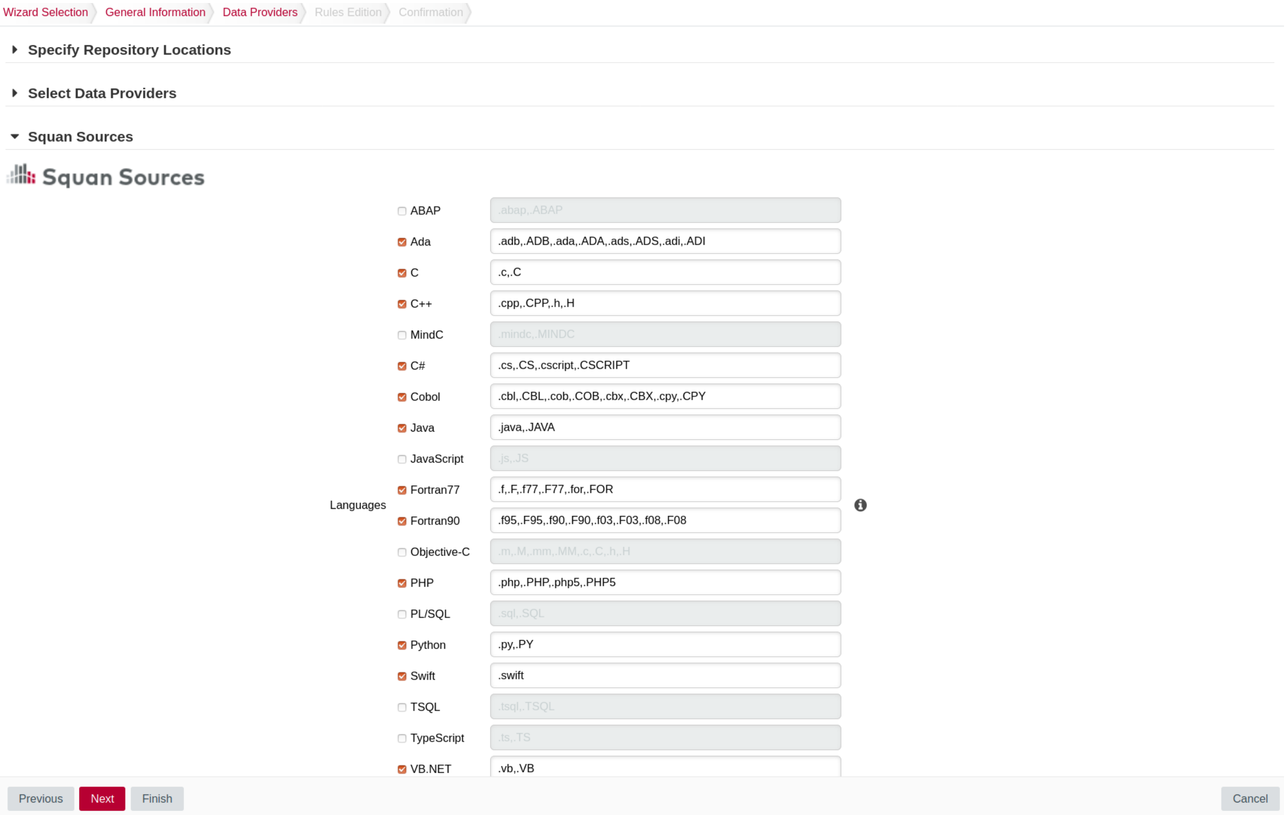
The Data Providers options screen

This screen allows configuring the repository locations and tools that will be used in your analysis. Set the source code files option to **Folder**. In the **Datapath** text box, type the path to the Neptune2 source code: \\*server**share**samples**c**Neptune**W25*.

The only Data Provider used in our analysis is Squan Sources, the source code analyser, so you can leave all the other tools unchecked.

If you want to learn more about the available Repository Connectors and Data Providers, consult Repository Connectors and Data Providers .

In the Squan Sources parameters, ensure that **C** is one of the programming languages selected, as shown below:

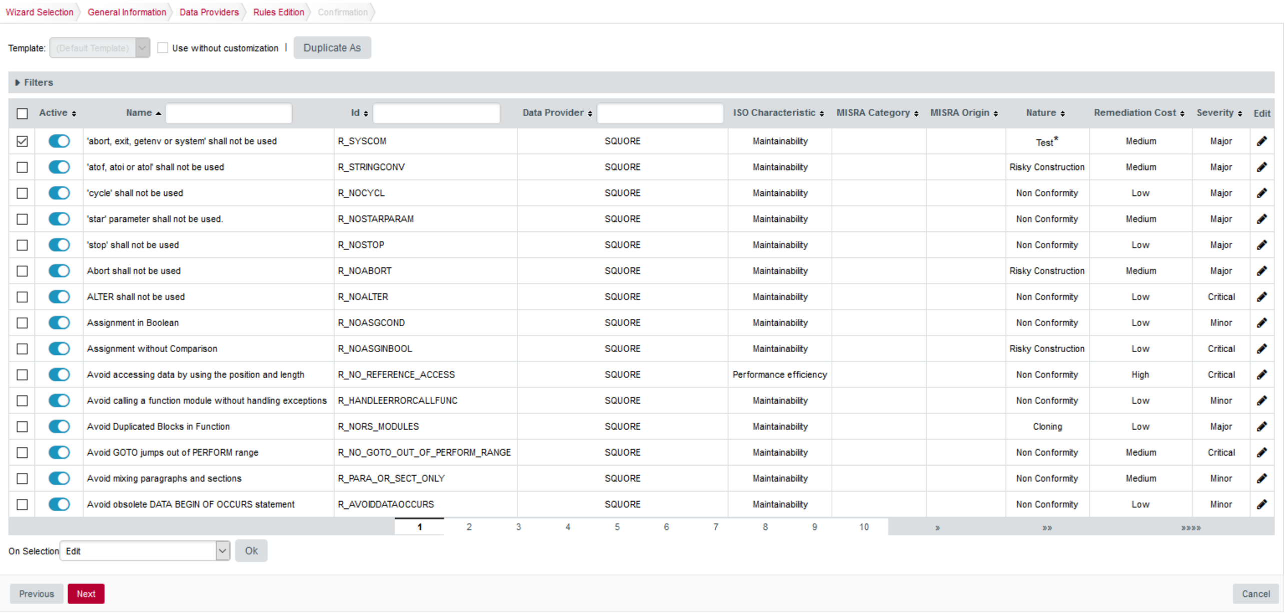


Languages

☐ ABAP	.abap,.ABAP
☑ Ada	.adb,.ADB,.ada,.ADA,.ads,.ADS,.adi,.ADI
☑ C	.c,.C
☑ C++	.cpp,.CPP,.h,.H
☐ MindC	.mindc,.MINDC
☑ C#	.cs,.CS,.cscript,.CSCRIPT
☑ Cobol	.cbl,.CBL,.cob,.COB,.cbx,.CBX,.cpy,.CPY
☑ Java	.java,.JAVA
☐ JavaScript	.js,.JS
☑ Fortran77	.f,.F,.f77,.F77,.for,.FOR
☑ Fortran90	.f95,.F95,.f90,.F90,.f03,.F03,.f08,.F08
☐ Objective-C	.m,.M,.mm,.MM,.c,.C,.h,.H
☑ PHP	.php,.PHP,.php5,.PHP5
☐ PL/SQL	.sql,.SQL
☑ Python	.py,.PY
☑ Swift	.swift
☐ TSQL	.tsql,.TSQL
☐ TypeScript	.ts,.TS
☑ VB.NET	.vb,.VB

The Squan Sources Data Provider parameters

8. Click the Next button to read the Rules Edition screen. This screen allows you to tweak the ruleset available in the analysis model.



☐	Active	Name	Id	Data Provider	ISO Characteristic	MISRA Category	MISRA Origin	Nature	Remediation Cost	Severity	Edit
☑	●	'abort, exit, getenv or system' shall not be used	R_SYSCOM	SQUORE	Maintainability			Test*	Medium	Major	✎
☐	●	'atof, atoi or atol' shall not be used	R_STRINGCONV	SQUORE	Maintainability			Risky Construction	Medium	Major	✎
☐	●	'cycle' shall not be used	R_NOCYCL	SQUORE	Maintainability			Non Conformity	Low	Major	✎
☐	●	'star' parameter shall not be used.	R_NOSTARPARAM	SQUORE	Maintainability			Non Conformity	Medium	Major	✎
☐	●	'stop' shall not be used	R_NOSTOP	SQUORE	Maintainability			Non Conformity	Low	Major	✎
☐	●	Abort shall not be used	R_NOABORT	SQUORE	Maintainability			Risky Construction	Medium	Major	✎
☐	●	ALTER shall not be used	R_NOALTER	SQUORE	Maintainability			Non Conformity	Low	Critical	✎
☐	●	Assignment in Boolean	R_NOASGCOND	SQUORE	Maintainability			Non Conformity	Low	Minor	✎
☐	●	Assignment without Comparison	R_NOASGINBOOL	SQUORE	Maintainability			Risky Construction	Low	Critical	✎
☐	●	Avoid accessing data by using the position and length	R_NO_REFERENCE_ACCESS	SQUORE	Performance efficiency			Non Conformity	High	Critical	✎
☐	●	Avoid calling a function module without handling exceptions	R_HANDLEERRORCALLFUNC	SQUORE	Maintainability			Non Conformity	Low	Minor	✎
☐	●	Avoid Duplicated Blocks in Function	R_NORS_MODULES	SQUORE	Maintainability			Cloning	Low	Major	✎
☐	●	Avoid GOTO jumps out of PERFORM range	R_NO_GOTO_OUT_OF_PERFORM_RANGE	SQUORE	Maintainability			Non Conformity	Medium	Critical	✎
☐	●	Avoid mixing paragraphs and sections	R_PARA_OR_SECT_ONLY	SQUORE	Maintainability			Non Conformity	Medium	Minor	✎
☐	●	Avoid obsolete DATA BEGIN OF OCCURS statement	R_AVOIDDATAOCCURS	SQUORE	Maintainability			Non Conformity	Low	Minor	✎

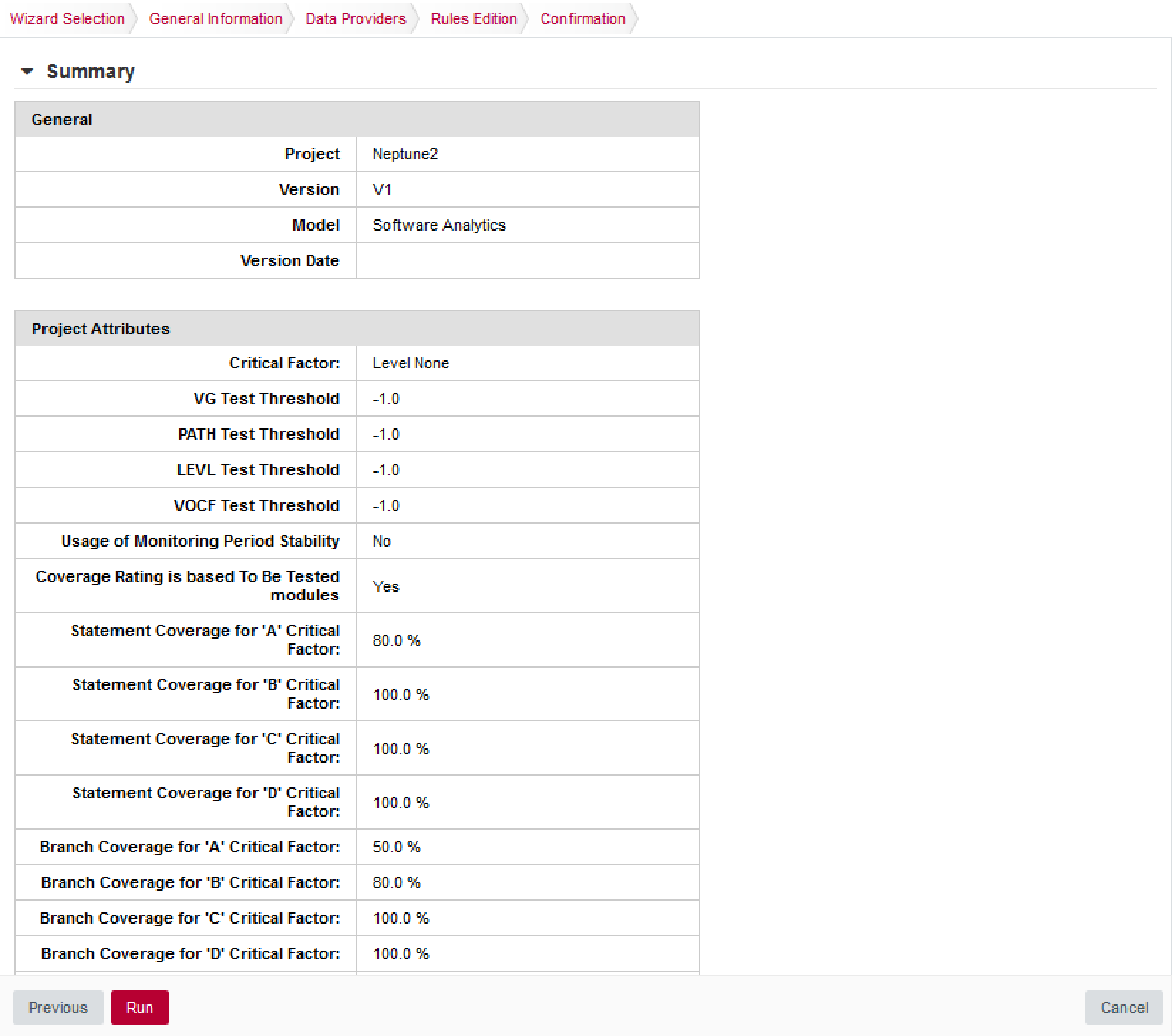
The Rules Edition screen

The table displays the entire model's ruleset, which you can filter and sort by data provider or category. Each rule can be turned on or off, and you can click the Edit button to adjust the categories for each rule. Note that any modifications from the original configuration are displayed with an asterisk.

Click the Next button when you are satisfied with your modifications. Note that your modifications are applied for any subsequent analysis of this project and do not affect other projects using the same model.

This screen may not be enabled in your wizard, as your administrator may have disabled it in your configuration. Your administrator can also decide to make modifications to the ruleset that apply to any project created with this model using the Analysis Model Editor. Consult Analysis Model Editor and Using Ruleset Templates to learn more.

9. Before launching the analysis, a summary of your selections is displayed. Review the information and click **Run** to confirm the project creation.



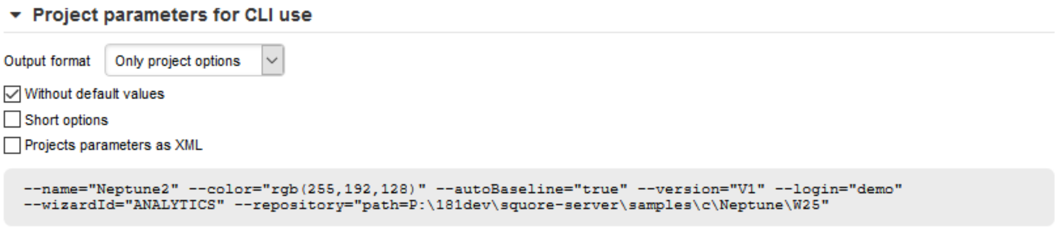
▾ Summary

General	
Project	Neptune2
Version	V1
Model	Software Analytics
Version Date	

Project Attributes	
Critical Factor:	Level None
VG Test Threshold	-1.0
PATH Test Threshold	-1.0
LEVL Test Threshold	-1.0
VOCF Test Threshold	-1.0
Usage of Monitoring Period Stability	No
Coverage Rating is based To Be Tested modules	Yes
Statement Coverage for 'A' Critical Factor:	80.0 %
Statement Coverage for 'B' Critical Factor:	100.0 %
Statement Coverage for 'C' Critical Factor:	100.0 %
Statement Coverage for 'D' Critical Factor:	100.0 %
Branch Coverage for 'A' Critical Factor:	50.0 %
Branch Coverage for 'B' Critical Factor:	80.0 %
Branch Coverage for 'C' Critical Factor:	100.0 %
Branch Coverage for 'D' Critical Factor:	100.0 %

The Rules Edition screen

The summary page lists all lists all the options you specified for the project creation and also allows outputting them in various formats so that you can repeat the project creation in command line. For more information about reusing the project parameters in a different context, consult the online help or Can I Create a Project Via the Command Line?.



▾ Project parameters for CLI use

Output format: Only project options

☑ Without default values

☐ Short options

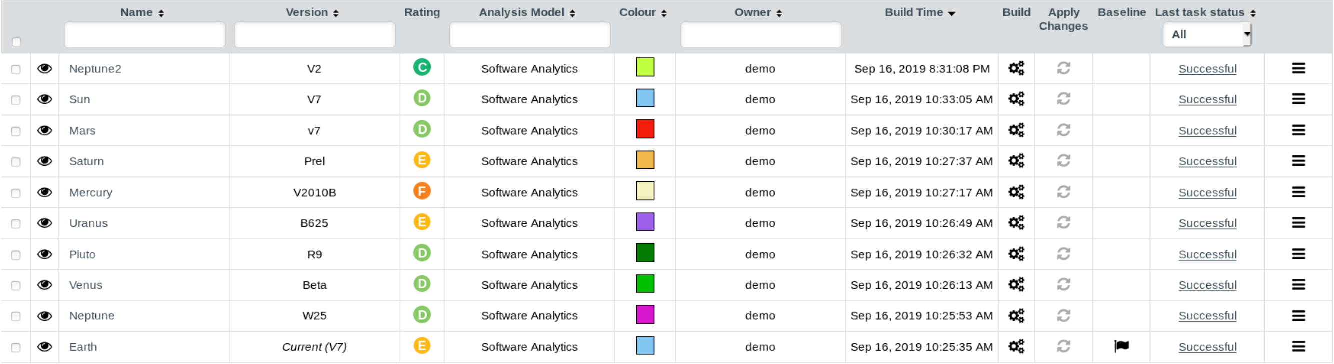
☐ Projects parameters as XML

```
--name="Neptune2" --color="rgb(255,192,128)" --autoBaseline="true" --version="V1" --login="demo"
--wizardId="ANALYTICS" --repository="path=P:\181dev\squore-server\samples\c\Neptune\W25"
```

Project Parameters for CLI use

How Do I Know the Project Creation Was Successful ?

When the project analysis completes, Squore shows you the list of projects. Neptune2 appears in the list, together with information about the current version and its computed rating:



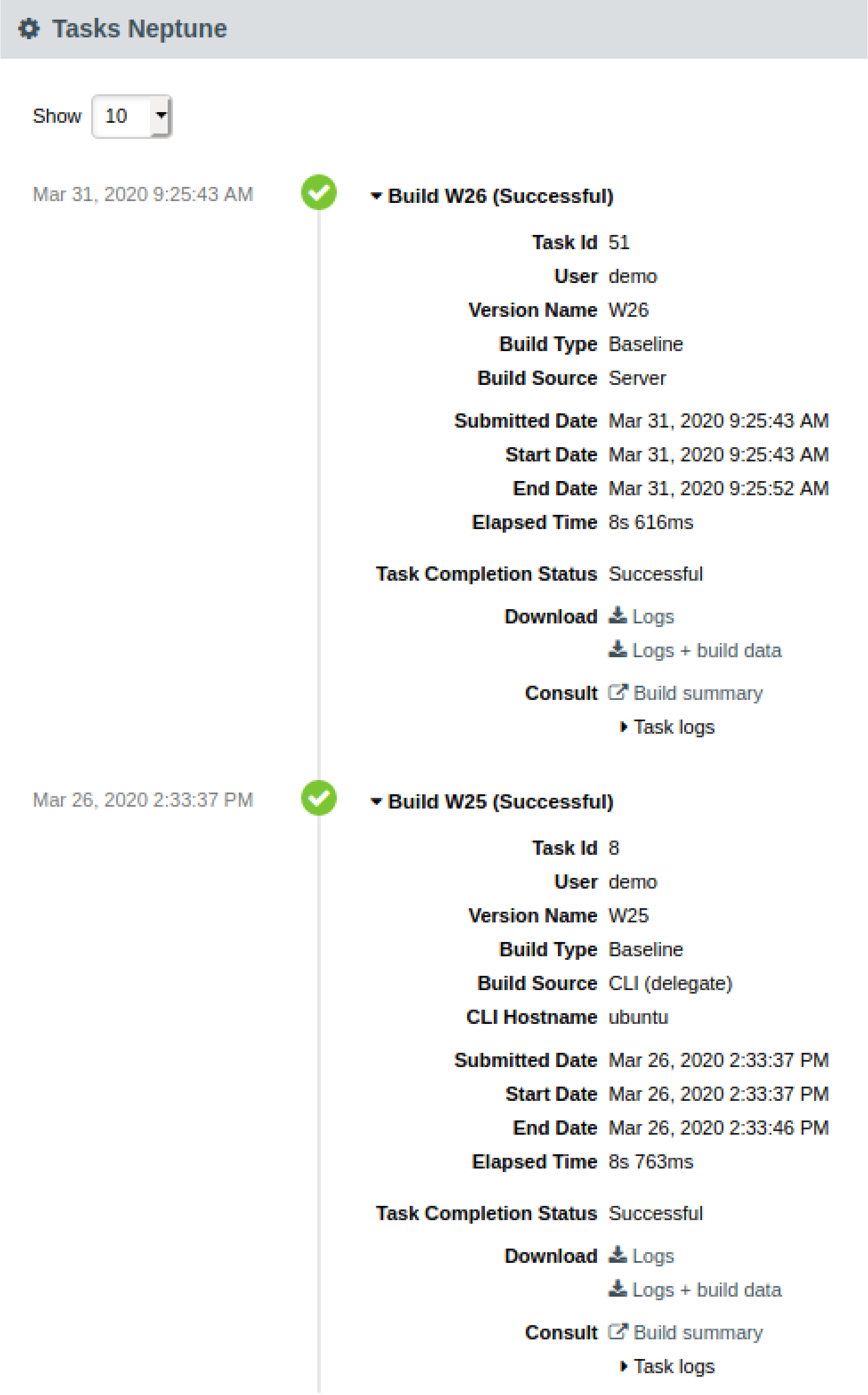
□		Name ⬍	Version ⬍	Rating	Analysis Model ⬍	Colour ⬍	Owner ⬍	Build Time ▾	Build	Apply Changes	Baseline	Last task status ⬍ All	
□	👁	Neptune2	V2	C	Software Analytics	🟩	demo	Sep 16, 2019 8:31:08 PM	⚙	⟳		Successful	☰
□	👁	Sun	V7	D	Software Analytics	🟦	demo	Sep 16, 2019 10:33:05 AM	⚙	⟳		Successful	☰
□	👁	Mars	v7	D	Software Analytics	🟥	demo	Sep 16, 2019 10:30:17 AM	⚙	⟳		Successful	☰
□	👁	Saturn	Prel	E	Software Analytics	🟧	demo	Sep 16, 2019 10:27:37 AM	⚙	⟳		Successful	☰
□	👁	Mercury	V2010B	F	Software Analytics	🟨	demo	Sep 16, 2019 10:27:17 AM	⚙	⟳		Successful	☰
□	👁	Uranus	B625	E	Software Analytics	🟪	demo	Sep 16, 2019 10:26:49 AM	⚙	⟳		Successful	☰
□	👁	Pluto	R9	D	Software Analytics	🟩	demo	Sep 16, 2019 10:26:32 AM	⚙	⟳		Successful	☰
□	👁	Venus	Beta	D	Software Analytics	🟩	demo	Sep 16, 2019 10:26:13 AM	⚙	⟳		Successful	☰
□	👁	Neptune	W25	D	Software Analytics	🟪	demo	Sep 16, 2019 10:25:53 AM	⚙	⟳		Successful	☰
□	👁	Earth	*Current (V7)*	E	Software Analytics	🟦	demo	Sep 16, 2019 10:25:35 AM	⚙	⟳	🏴	Successful	☰

The projects list

To consult the results of the analysis, click on the project name to view the Squore Dashboard. More information on how to read the Dashboard is available in Where Are My Analysis Results?.

In the last column you can see the status of the last executed task. If the status is at **Successful** then it's creation was completed successfully.

By clicking on the last task status you are redirected to the **Project's Tasks** page :



Tasks Neptune

Show 10

Mar 31, 2020 9:25:43 AM

Build W26 (Successful)

Task Id	51
User	demo
Version Name	W26
Build Type	Baseline
Build Source	Server
Submitted Date	Mar 31, 2020 9:25:43 AM
Start Date	Mar 31, 2020 9:25:43 AM
End Date	Mar 31, 2020 9:25:52 AM
Elapsed Time	8s 616ms
Task Completion Status	Successful
Download	Logs Logs + build data
Consult	Build summary Task logs

Mar 26, 2020 2:33:37 PM

Build W25 (Successful)

Task Id	8
User	demo
Version Name	W25
Build Type	Baseline
Build Source	CLI (delegate)
CLI Hostname	ubuntu
Submitted Date	Mar 26, 2020 2:33:37 PM
Start Date	Mar 26, 2020 2:33:37 PM
End Date	Mar 26, 2020 2:33:46 PM
Elapsed Time	8s 763ms
Task Completion Status	Successful
Download	Logs Logs + build data
Consult	Build summary Task logs

The Project's Tasks page

In this page you have access to the list of all the tasks executed on this project, as well as their logs : version builds, reapply model, cloning, etc...

Creating Version 2 of My Project

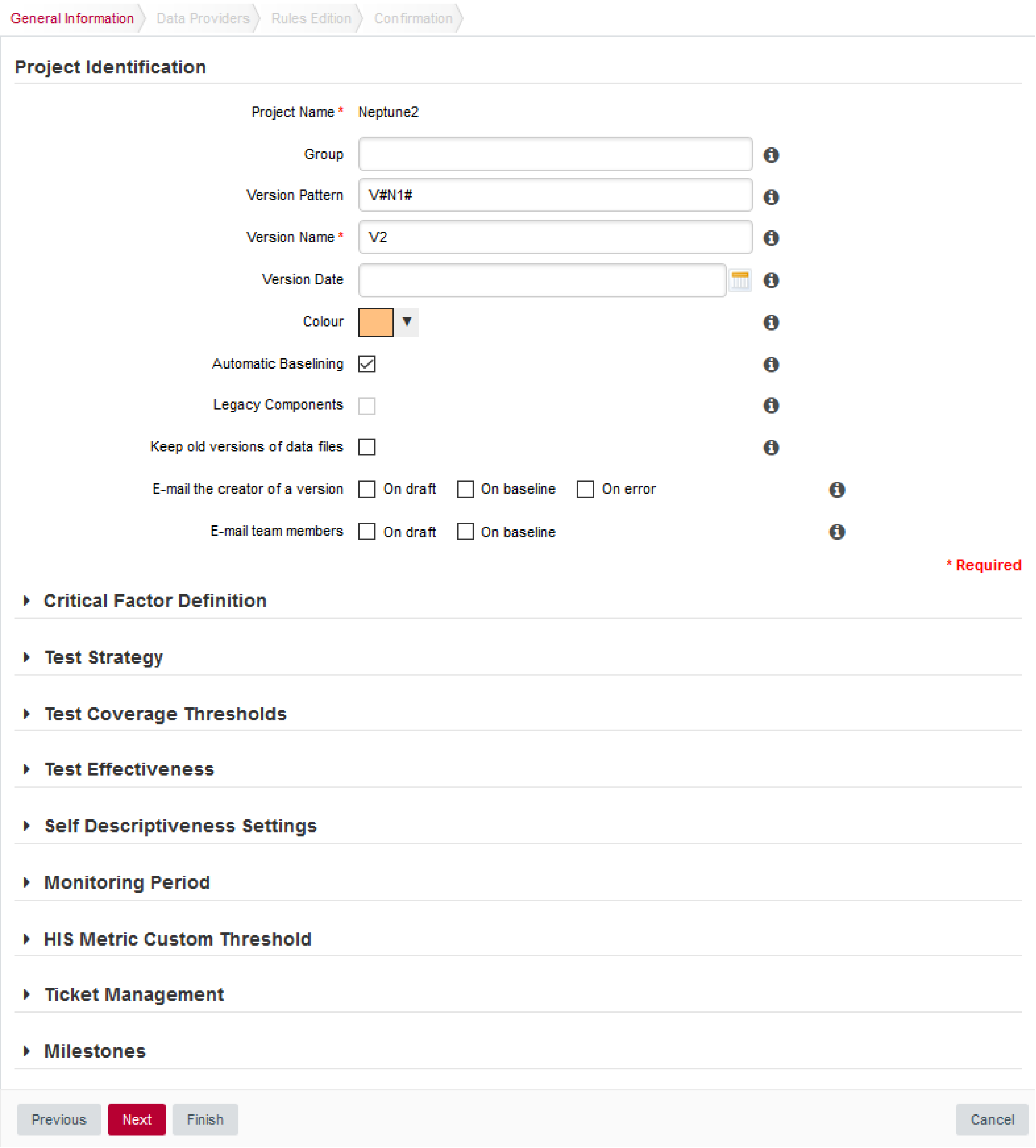
Adding a version to an already-existing project is a simple procedure that is carried out from the **Projects** page.

Follow these steps to create version 2 of your project:

1. After logging into Squore, click on **Projects**.

2. Click the **Build** icon (⚙) for the Neptune2 project in order to access the source code file options.

3. The first screen of the wizard enables you to specify the version name and to modify some of the project attributes if necessary.



Project Identification

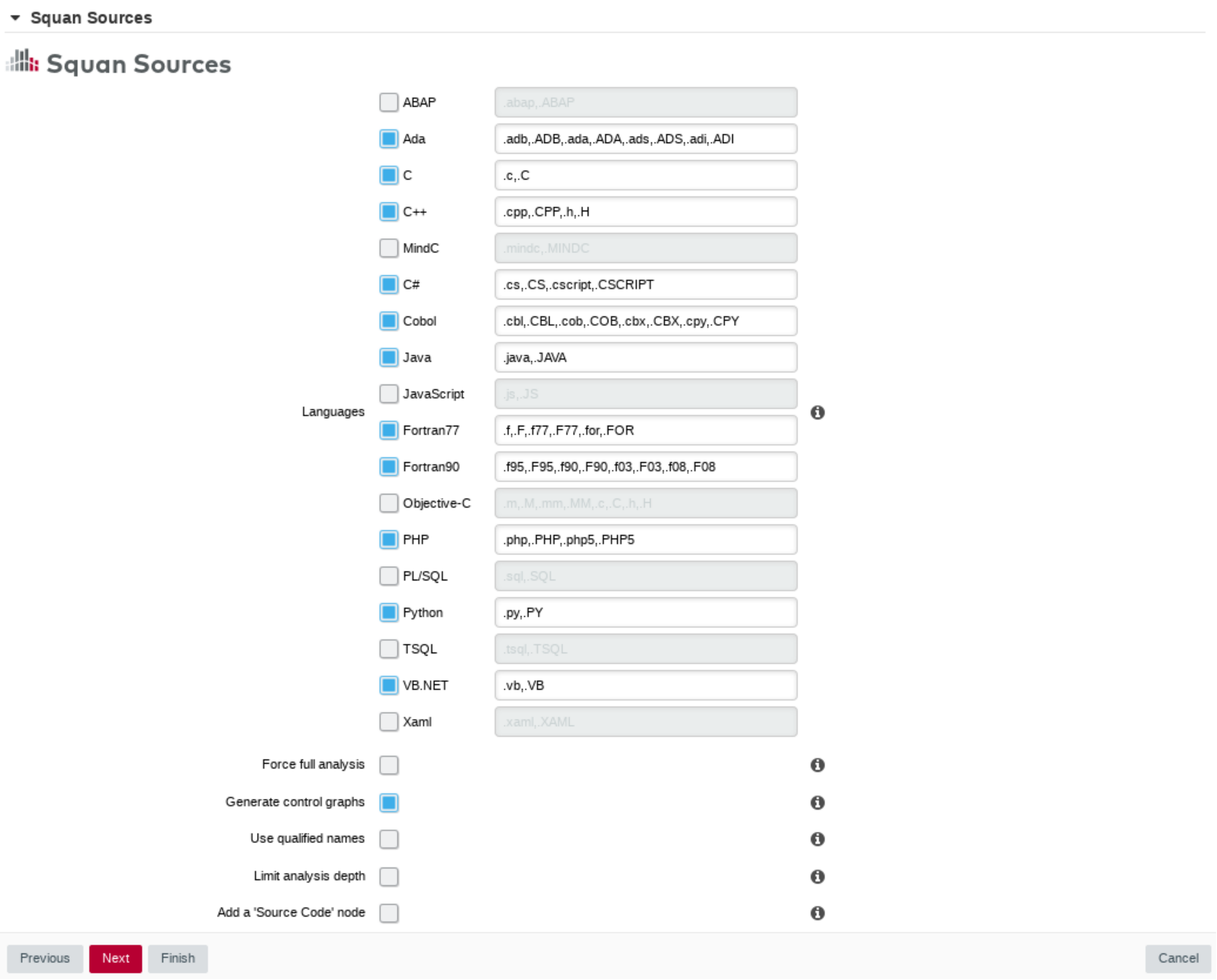
Project Name *	Neptune2
Group	
Version Pattern	V#N1#
Version Name *	V2
Version Date	
Colour	
Automatic Baselining	☑
Legacy Components	□
Keep old versions of data files	□
E-mail the creator of a version	□ On draft □ On baseline □ On error
E-mail team members	□ On draft □ On baseline

* Required

- Critical Factor Definition
- Test Strategy
- Test Coverage Thresholds
- Test Effectiveness
- Self Descriptiveness Settings
- Monitoring Period
- HIS Metric Custom Threshold
- Ticket Management
- Milestones

Parameters For the New Version of Neptune2

4. Click the Next button to reach the project language and source settings screen. On this screen, you can modify the path to the source code and point to the newer version. Note that by default, Squore displays the path used when analysing the last version. Leave the path as it was for version 1. We are going to create a version that analyses the same code in this example. If you scroll down to the code analysis option, you will notice that some of them are now disabled. This is because the project configuration was set in version 1 and is not allowed to be modified in subsequent analyses. This ensures that your project is scored using the same criteria every time you analyse new code.



▾ Squan Sources

Squan Sources

Languages

☐	ABAP	.abap,.ABAP
☑	Ada	.adb,.ADB,.ada,.ADA,.ads,.ADS,.adi,.ADI
☑	C	.c,.C
☑	C++	.cpp,.CPP,.h,.H
☐	MindC	.mindc,.MINDC
☑	C#	.cs,.CS,.cscript,.CSCRIPT
☑	Cobol	.cbl,.CBL,.cob,.COB,.cbx,.CBX,.cpy,.CPY
☑	Java	.java,.JAVA
☐	JavaScript	.js,.JS
☑	Fortran77	.f,.F,.f77,.F77,.for,.FOR
☑	Fortran90	.f95,.F95,.f90,.F90,.f03,.F03,.f08,.F08
☐	Objective-C	.m,.M,.mm,.MM,.c,.C,.h,.H
☑	PHP	.php,.PHP,.php5,.PHP5
☐	PL/SQL	.sql,.SQL
☑	Python	.py,.PY
☐	TSQL	.tsql,.TSQL
☑	VB.NET	.vb,.VB
☐	Xaml	.xaml,.XAML

Force full analysis ☐

Generate control graphs ☑

Use qualified names ☐

Limit analysis depth ☐

Add a 'Source Code' node ☐

Unavailable options when creating version 2 of a project

You can add new sources to the project at this stage if needed. Read more about projects using sources spread over multiple locations in Can I Create Projects with Sources From Multiple Locations?.

5. Click **Finish** and **Run** to launch the analysis of Neptune2 V2. When the analysis finishes, Neptune2 V2 will be listed in the list of projects on the Projects page.

Working with Draft and Baseline Versions

This section covers an essential workflow feature of Squore: baselining. While it is possible to keep every version of a project created in Squore, you may want to permanently keep analysis results only for particular milestones and work with an always updating draft version.

You can decide whether a version is a draft or a baseline when you create it, or after the analysis is finished.

Drafts and Baseline: The Basic Concepts

The most important thing to remember about a draft version is that it is a snapshot of your data at a given time. You can use it to compare the evolution of your project against the last baseline created. There is therefore only one draft version available per project (the latest version), which Squore creates automatically if your previous version was a baseline. A baseline version, on the other hand, is permanently saved and will not be overwritten the next time an analysis is launched.

When you create a draft version, it is always called Current and can be modified in several ways:

- Forms can be updated
- Attribute values can be modified so that a new value is taken into account in the next analysis
- Artefacts can be manually added, modified or deleted
- Folders and files can be relaxed or excluded from the project
- Action Items can have their status changed
- Rules and individual violations can be relaxed

Being able to view draft versions of a project is a user privilege that can be granted to users of a particular project role, and so is the ability to baseline a project. For more information about project roles, refer to User Project Roles. This means that as a project manager, you can give access to every version to users within your team, but can restrict the project visibility to the rest of the company to show them only baselined versions. You can also decide which members of your team are allowed to change the status of a version from draft to baseline.

Baselining at Version Creation

Use the Automatic Baselining option on the General Information screen of the project wizard to create a draft or baseline as follows:

- When the Automatic Baselining box is unchecked, a draft version is created and all subsequent versions will be draft versions by default.
- When the Automatic Baselining box is checked, a baseline version is created and all subsequent versions will be baseline versions by default.

Baselining After Review

You can use the Baseline option on the Projects page to create a baseline version of the current draft as follows:

1. Log into Squore and click on **Projects**.
2. Click the Baseline icon (⚑) next to the project you want to baseline.
3. Click the **Baseline** button to confirm.

After confirming the baseline creation, you are redirected to the Projects page and the last draft version becomes the new latest baseline. All changes made manually to artefacts and findings are kept, and will be incorporated the next time an analysis runs. Note that baselining is only available for users whose project role includes the **Baseline Projects** privilege. For more information about project roles, refer to User Project Roles.

> Baselining manually is useful if you have reviewed the current draft and have not made any changes to the analysis results. If you have modified form entries or relaxed artefacts and findings in a way that should impact the rating, consider launching a new build or using the Apply Changes button instead of baselining. See Handling Manual Modifications for more information.

Handling Manual Modifications

When you have made changes to form values or you have relaxed artefacts and findings in the current draft, there are two ways to get these changes reflected in the dashboard:

- running a new analysis
- clicking Apply Changes

Running a new analysis will allow you to change the source code repository settings and input files for data providers, or keep them. You can choose if this new analysis should produce a baseline or a draft version. In all cases, the artefacts you relaxed or excluded, the action items you modified and the findings you relaxed are taken into account to produce the rating of the new version.

Clicking Apply Changes (🔄) allows you to merge the manual modifications to artefacts, forms, action items and findings into a new draft version without reanalysing source code and re-running data providers. Manual modifications are simply merged with the already- existing results to update the rating, which is a lot faster than running a full analysis. Note that Apply Changes is not available when you have excluded artefacts.

You can also use Apply Changes after modifying your analysis model to migrate a project to the new version of a model without running a full analysis.

Can I Make Changes to My Project?

There are three types of changes you can make to Squore projects :

- Changes to attribute values
- Changes to source code locations
- Changes to some of the Data Provider options

Project attributes are always editable when creating a new version of a project, except for the name of the project.

The location of the source code can always be modified. When editing a project, you can also add more source locations as needed, following the steps described in Can I Create Projects with Sources From Multiple Locations?.

Whether you can edit the settings used in the Data Providers for the project depends on their ability to support edits. This ability is defined by a Squore administrator via the configuration of the Squore wizards. For more information, refer to the Squore Configuration Guide.

Can I Create a Project Via the Command Line?

Instead of creating a project from the Squore web interface, you can create a project directly from the command line using Squore CLI. Squore CLI is a client for Squore that enables you to create and analyse projects locally and send the results to Squore Server. Alternatively, you can use Squore CLI to instruct Squore Server to carry out the analysis.

If you have installed Squore CLI on your computer, you can call it using Java, passing the parameters you would have passed in the web interface to create projects. The following is an example of the command line you can use to create a project using Squore CLI on Windows:

```
@echo off
java -Dsquore.home.dir="%SQUORE_HOME%" ^
-jar %SQUORE_HOME%\lib\squore-engine.jar ^
--url=http://localhost:8180/SQuORE_Server ^
--commands=DELEGATE_CREATION ^
--name=Mars2 ^
--repository "type=FROMPATH,path=\server\share\samples\c\Mars2\V3.2.6" ^
--color=rgb(103,25,237) ^
--version=1.0 ^
--login=demo ^
--password=demo ^
--filter=APPLICATION,MEASURE,LEVEL ^
--wizardId="ANALYTICS" ^
--dp "type=SQuORE"
echo done
pause
```

The example above shows how to specify commands, parameters and project options to Squore CLI. This would create a project named **Mars2** in version **1.0**, analysing source code located in **\server\share\samples\c\Mars2\V3.2.6** with the Data Provider **SQuORE** (the internal name for Squan Sources).

You can find more information about using Squore CLI in the Command Line Interface manual, which explains how to install the client and create projects.

How Do I Connect Squore to My Continuous Integration System?

If you use a Continuous Integration tool like Jenkins or CruiseControl, you can add Squore to your build process and analyse projects every time your code is compiled. This requires the installation of Squore CLI on the continuous integration server, and is therefore described in greater details in the Command Line Interface Manual.

Can Squore Pull Source From My Version Control System?

The source code analysed by Squore does not have to be located on the same machine as Squore Server or Squore CLI. When you create a project, you get the option to choose from a range of Repository Connectors to pull source code from:

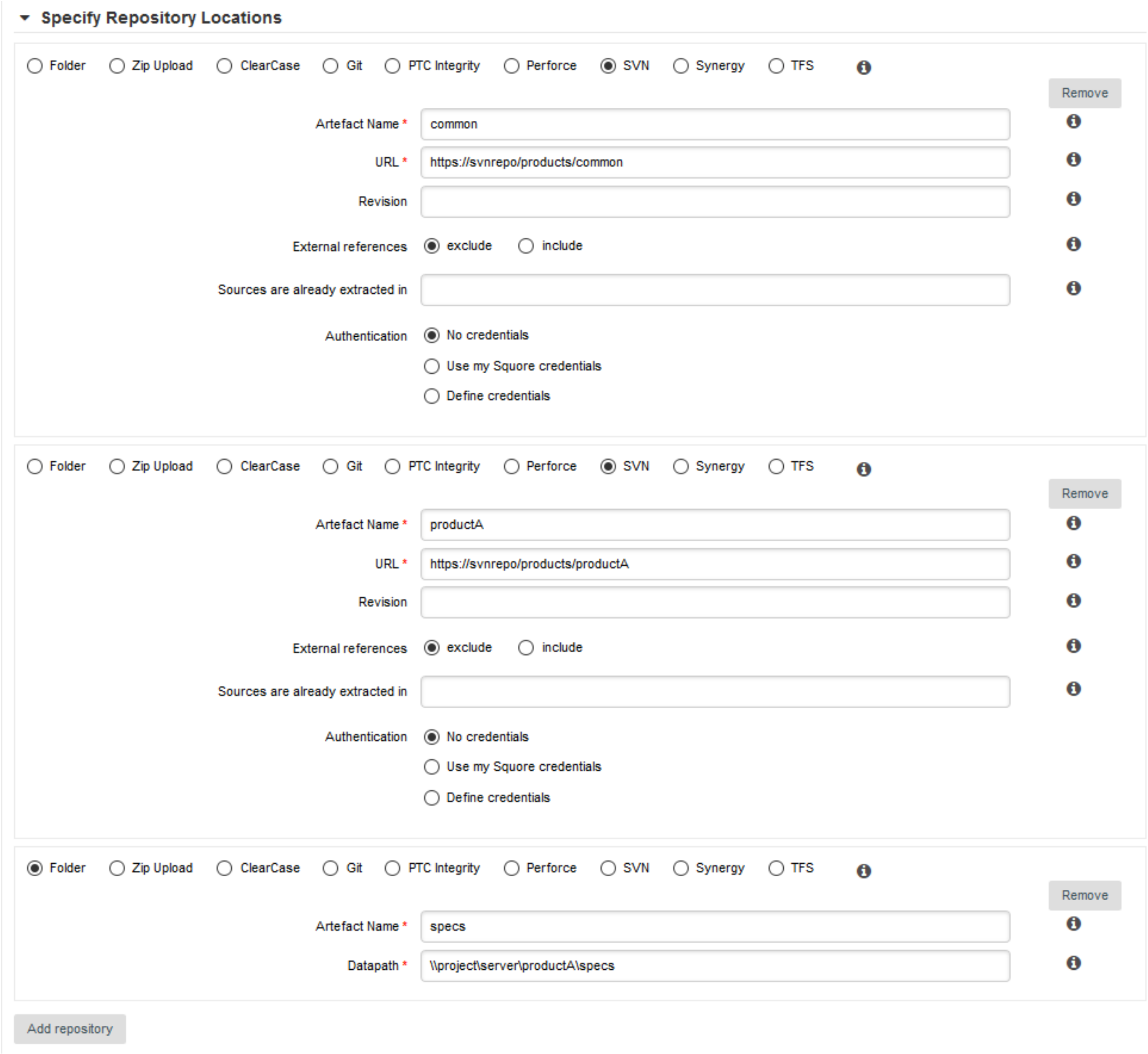
- Direct file system access (local drive, network share, mass storage media...)
- Zip upload
- A ClearCase view
- A CVS checkout
- A GNAThub database
- Git cloning
- An Integrity repository
- A Perforce depot
- A Subversion revision
- A Synergy database

- A TFS server

Each option requires different parameters, which can be specified from the project wizard, or via the command line. For more information, refer to Repository Connectors.

Can I Create Projects with Sources From Multiple Locations?

Squore provides support for analysing projects whose sources are spread over several locations or version control systems. If your source code resides in */products/common* and */projects/myproject*, you can specify these two locations in the Squore project wizard by clicking the **Add Repository** button. Similarly, if some of your code is managed by a SVN repository and the rest is handled by a Git server, you can configure both locations as part of the same project, as shown below:



▾ Specify Repository Locations

○ Folder ○ Zip Upload ○ ClearCase ○ Git ○ PTC Integrity ○ Perforce ● SVN ○ Synergy ○ TFS

Remove

Artefact Name *: common

URL *: https://svnrepo/products/common

Revision:

External references: ● exclude ○ include

Sources are already extracted in:

Authentication: ● No credentials ○ Use my Squore credentials ○ Define credentials

○ Folder ○ Zip Upload ○ ClearCase ○ Git ○ PTC Integrity ○ Perforce ● SVN ○ Synergy ○ TFS

Remove

Artefact Name *: productA

URL *: https://svnrepo/products/productA

Revision:

External references: ● exclude ○ include

Sources are already extracted in:

Authentication: ● No credentials ○ Use my Squore credentials ○ Define credentials

● Folder ○ Zip Upload ○ ClearCase ○ Git ○ PTC Integrity ○ Perforce ○ SVN ○ Synergy ○ TFS

Remove

Artefact Name *: specs

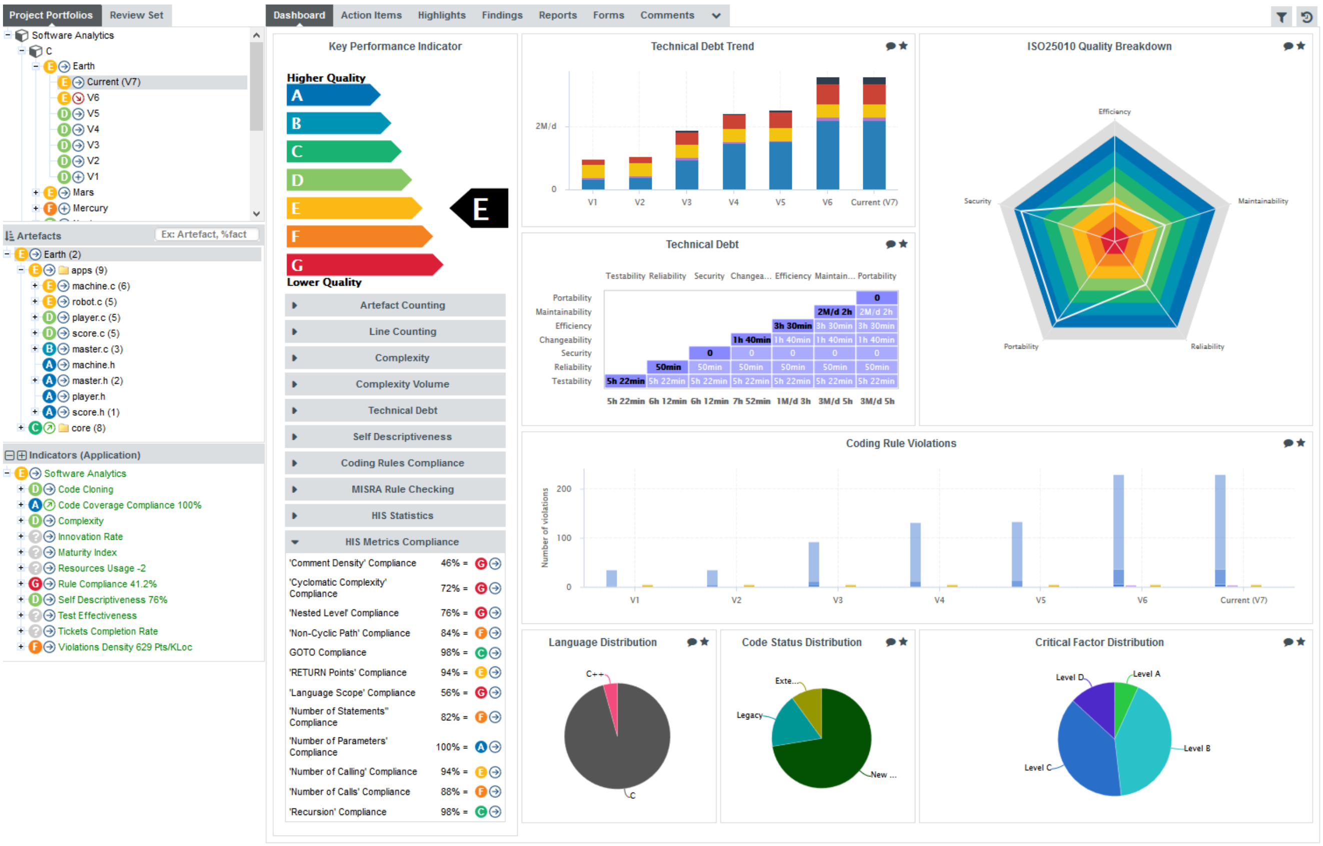
Datapath *: \\project\server\productA\specs

Add repository

A project using sources from two SVN repositories and a network drive

Where Are My Analysis Results?

Now that you have created a project, you are ready to start reviewing the analysis results in the main section of Squore, the Explorer, which consists of a set of trees for browsing through project artefacts and various dashboards to display the information associated with these artefacts.

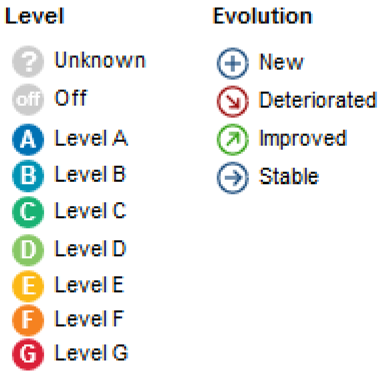


'Comment Density' Compliance	46% =	G
'Cyclomatic Complexity' Compliance	72% =	G
'Nested Level' Compliance	76% =	G
'Non-Cyclic Path' Compliance	84% =	F
GOTO Compliance	98% =	C
'RETURN Points' Compliance	94% =	E
'Language Scope' Compliance	56% =	G
'Number of Statements" Compliance	82% =	F
'Number of Parameters' Compliance	100% =	A
'Number of Calling' Compliance	94% =	E
'Number of Calls' Compliance	88% =	F
'Recursion' Compliance	98% =	C

	Testability	Reliability	Security	Changea...	Efficiency	Maintain...	Portability
Portability							0
Maintainability						2M/d 2h	2M/d 2h
Efficiency					3h 30min	3h 30min	3h 30min
Changeability				1h 40min	1h 40min	1h 40min	1h 40min
Security			0	0	0	0	0
Reliability		50min	50min	50min	50min	50min	50min
Testability	5h 22min	5h 22min	5h 22min	5h 22min	5h 22min	5h 22min	5h 22min
	5h 22min	6h 12min	6h 12min	7h 52min	1M/d 3h	3M/d 5h	3M/d 5h

The Squore Explorer

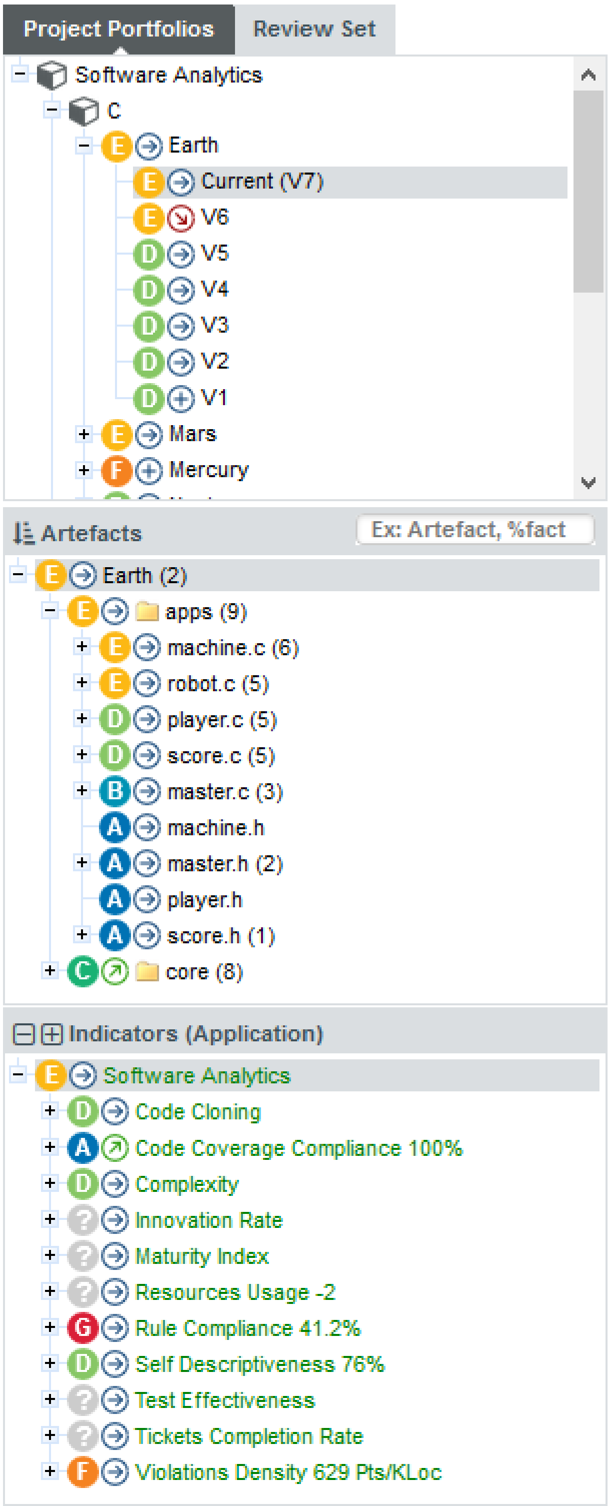
Common icons are used throughout the explorer to indicate the rating of a component and its evolution compared to the previous version. The image below shows the meaning of the different icons used:



The Squore Explorer icons

The Tree Pane

The left-hand part of the Explorer is a three-panel section containing expandable trees.



Project Portfolios | Review Set

- Software Analytics
  - C
    - E Earth
      - E Current (V7)
      - E V6
      - D V5
      - D V4
      - D V3
      - D V2
      - D V1
    - E Mars
    - F Mercury

Artefacts — Ex: Artefact, %fact

- E Earth (2)
  - E apps (9)
    - E machine.c (6)
    - E robot.c (5)
    - D player.c (5)
    - D score.c (5)
    - B master.c (3)
    - A machine.h
    - A master.h (2)
    - A player.h
    - A score.h (1)
  - C core (8)

Indicators (Application)

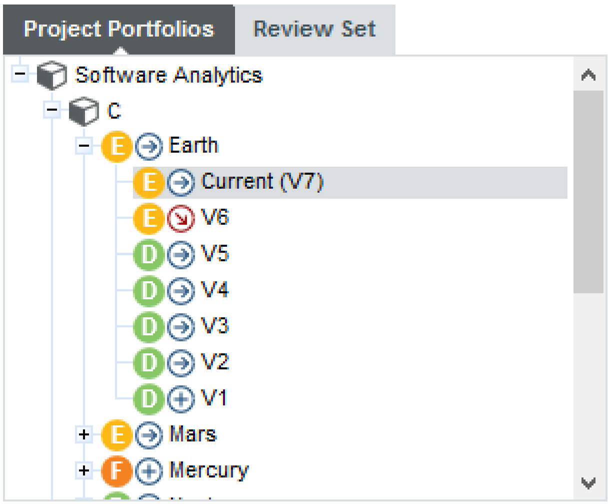
- E Software Analytics
  - D Code Cloning
  - A Code Coverage Compliance 100%
  - D Complexity
  - ? Innovation Rate
  - ? Maturity Index
  - ? Resources Usage -2
  - G Rule Compliance 41.2%
  - D Self Descriptiveness 76%
  - ? Test Effectiveness
  - ? Tickets Completion Rate
  - F Violations Density 629 Pts/KLoc

The Tree Pane

The top panel contains the **Project Portfolios** and the **Review Set**.

The Project Portfolios is a list of all the projects you have access to, grouped by analysis model. Each project is listed with its latest rating and evolution and can be expanded to show all versions of the project that were analysed with Squore.

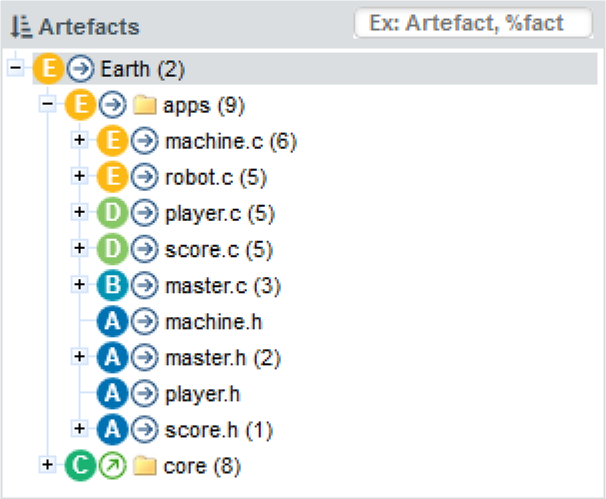
The Review Set is a flat list of artefacts you collect from various projects in order to review them. This list is saved when you log out and log into Squore again.



Software Analytics
C
E Earth
E Current (V7)
E V6
D V5
D V4
D V3
D V2
D V1
E Mars
F Mercury

The expanded Earth project, rated E, and its 7 versions in the Project Portfolios

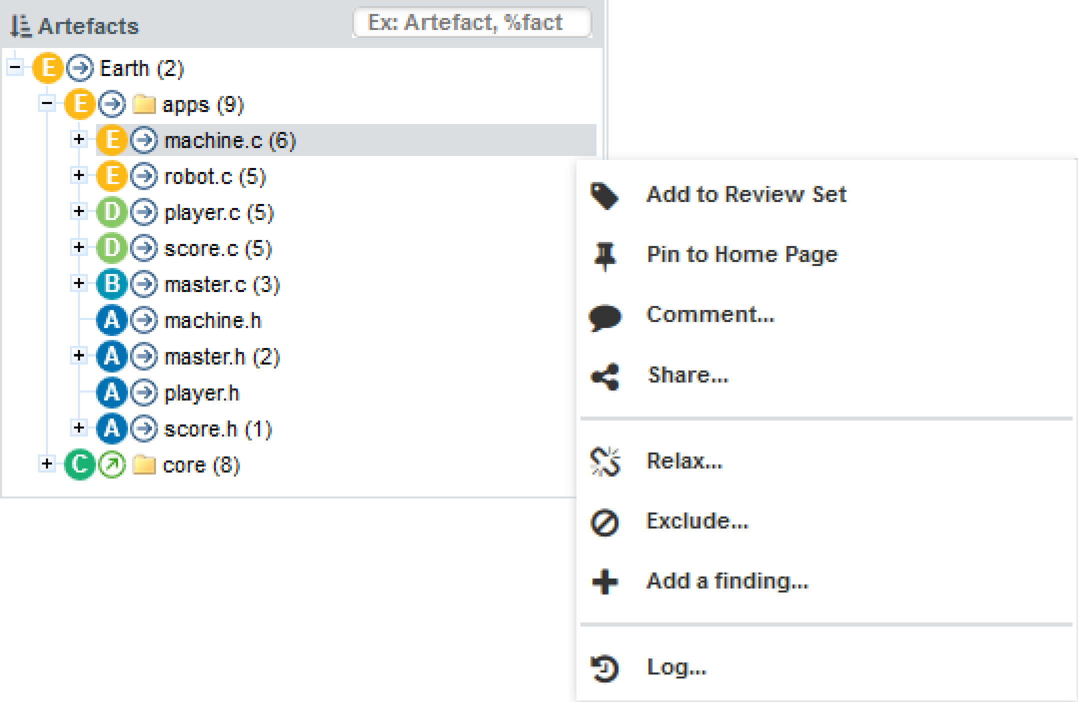
The tree in the middle panel is the **Artefact Tree**. When Squore analyses a project, it breaks it down into artefacts of various configurable types, down to the function-level for source code, according to the analysis model used. The artefacts in the tree are displayed for the version selected in the Project Portfolios. clicking a different version of a project refreshes the artefact tree with the ratings for the version just selected. Above the artefact tree are tools for sorting and searching artefacts. Each artefact is displayed with its current rating and can be expanded to reveal child artefacts if available. The number in brackets indicates the amount of child artefacts for the current artefact. You will learn about these tools later in Has the Quality of My Project Decreased Since the Previous Analysis? and How Do I Find and Keep Track of Artefacts?.



E Earth (2)
E apps (9)
E machine.c (6)
E robot.c (5)
D player.c (5)
D score.c (5)
B master.c (3)
A machine.h
A master.h (2)
A player.h
A score.h (1)
C core (8)

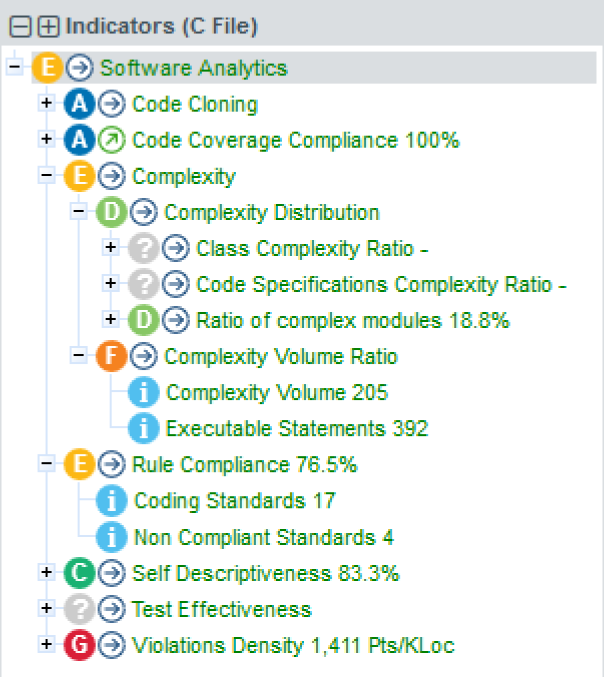
The Artefact Tree for version 7 of the Earth project

You can also interact with artefacts in the tree by using the Artefact Context Menu which can be accessed by hovering over an artefact name and clicking the menu icon.


Artefacts
Ex: Artefact, %fact
E Earth (2)
E apps (9)
E machine.c (6)
E robot.c (5)
D player.c (5)
D score.c (5)
B master.c (3)
A machine.h
A master.h (2)
A player.h
A score.h (1)
C core (8)

The Artefact Context Menu

The bottom panel is the **Indicator Tree** , in which ratings for the indicators defined in the analysis model at the current level are displayed. Each indicator can be expanded to display the rating of each of its sub-indicators. The Indicator Tree displays statistics for the artefact currently selected in the Artefact Tree and refreshes when the selection is changed. The type of artefact selected in indicated in brackets. Two shortcut buttons can be found above the top node to quickly expand and collapse the entire tree.


Indicators (C File)
E Software Analytics
A Code Cloning
A Code Coverage Compliance 100%
E Complexity
D Complexity Distribution
? Class Complexity Ratio -
? Code Specifications Complexity Ratio -
D Ratio of complex modules 18.8%
F Complexity Volume Ratio
Complexity Volume 205
Executable Statements 392
E Rule Compliance 76.5%
Coding Standards 17
Non Compliant Standards 4
C Self Descriptiveness 83.3%
? Test Effectiveness
G Violations Density 1,411 Pts/KLoc

The partly expanded Indicator Tree for version 7 of the Earth project at Application level

Clicking one of the tree nodes reveals more information about the indicator, including the formula used by Squore to compute its value and rating.

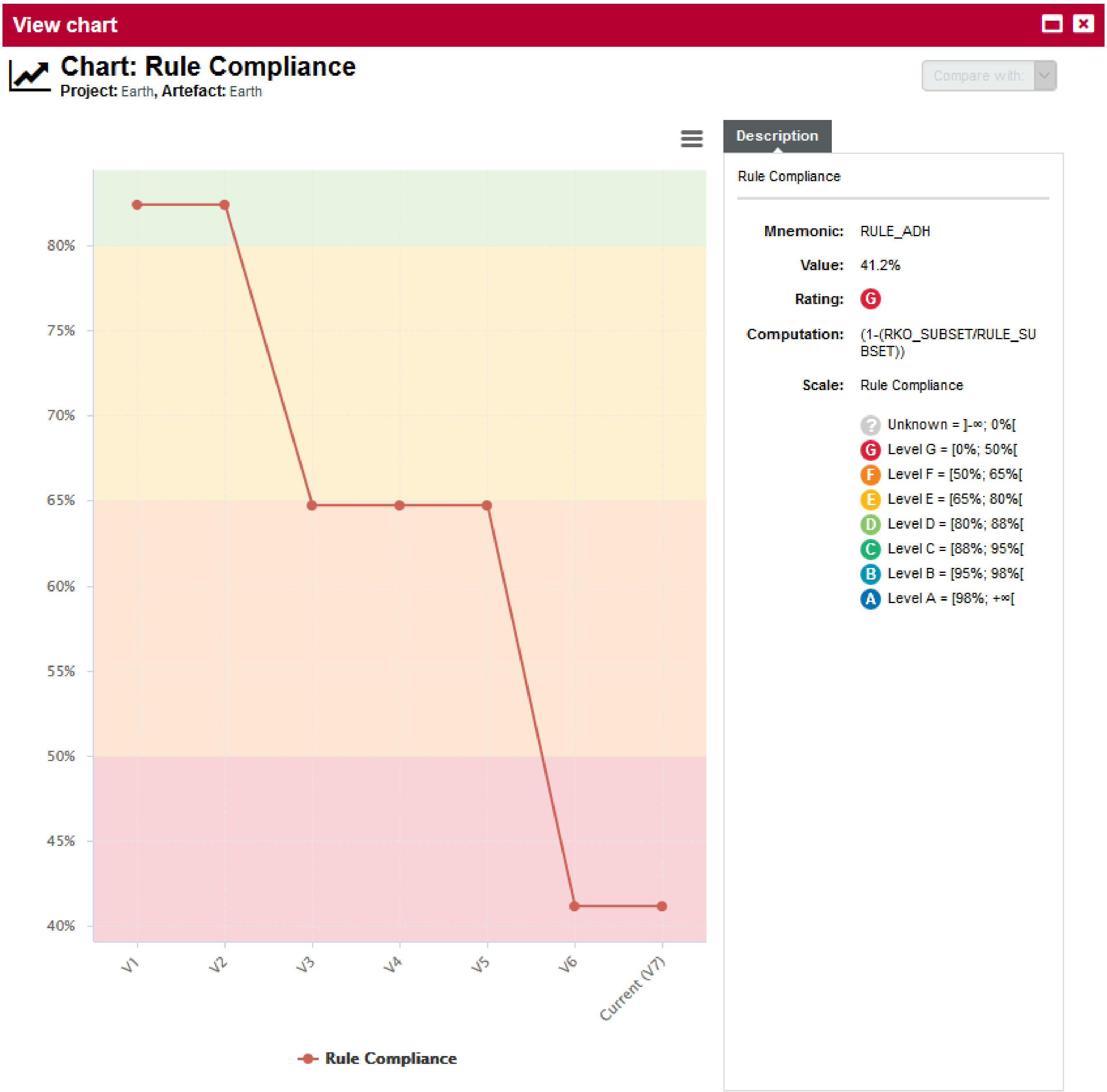


Chart: Rule Compliance
Project: Earth, **Artefact:** Earth

Rule Compliance

Mnemonic: RULE_ADH
Value: 41.2%
Rating: G
Computation: (1-(RKO_SUBSET/RULE_SUBSET))
Scale: Rule Compliance

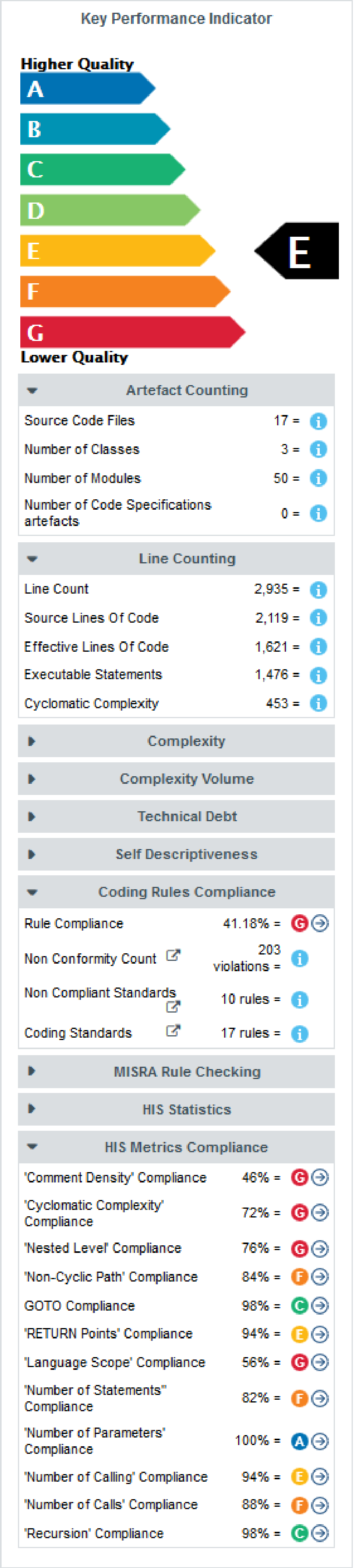
- ? Unknown =]-∞; 0%[
- G Level G = [0%; 50%[
- F Level F = [50%; 65%[
- E Level E = [65%; 80%[
- D Level D = [80%; 88%[
- C Level C = [88%; 95%[
- B Level B = [95%; 98%[
- A Level A = [98%; +∞[

The popup displayed when clicking the Complexity Distribution indicator at application level

The Dashboards

The right-hand side of the Squore Explorer contains a series of tabs, the first of which is the Dashboard. The Dashboard is dynamic and always displays information about the artefact currently selected in the artefact tree. There is not one Dashboard, but a Dashboard per node in the tree. Additionally, Dashboards can be customised by a Squore administrator so that users see different Dashboards according to their groups and to their role in a project, thus highlighting different information for project managers, quality engineers and developers for example. Ask your Squore administrator about Dashboard customisation, or refer to the Squore Configuration Guide for more information.

The left-hand area of the Dashboard contains the score card , which consists of a graphical representation of the key performance indicator for the current artefact, and some tables highlighting key metrics about the project.



Key Performance Indicator

▾ Artefact Counting

Source Code Files	17 = ⓘ
Number of Classes	3 = ⓘ
Number of Modules	50 = ⓘ
Number of Code Specifications artefacts	0 = ⓘ

▾ Line Counting

Line Count	2,935 = ⓘ
Source Lines Of Code	2,119 = ⓘ
Effective Lines Of Code	1,621 = ⓘ
Executable Statements	1,476 = ⓘ
Cyclomatic Complexity	453 = ⓘ

▸ Complexity

▸ Complexity Volume

▸ Technical Debt

▸ Self Descriptiveness

▾ Coding Rules Compliance

Rule Compliance	41.18% = G →
Non Conformity Count	203 violations = ⓘ
Non Compliant Standards	10 rules = ⓘ
Coding Standards	17 rules = ⓘ

▸ MISRA Rule Checking

▸ HIS Statistics

▾ HIS Metrics Compliance

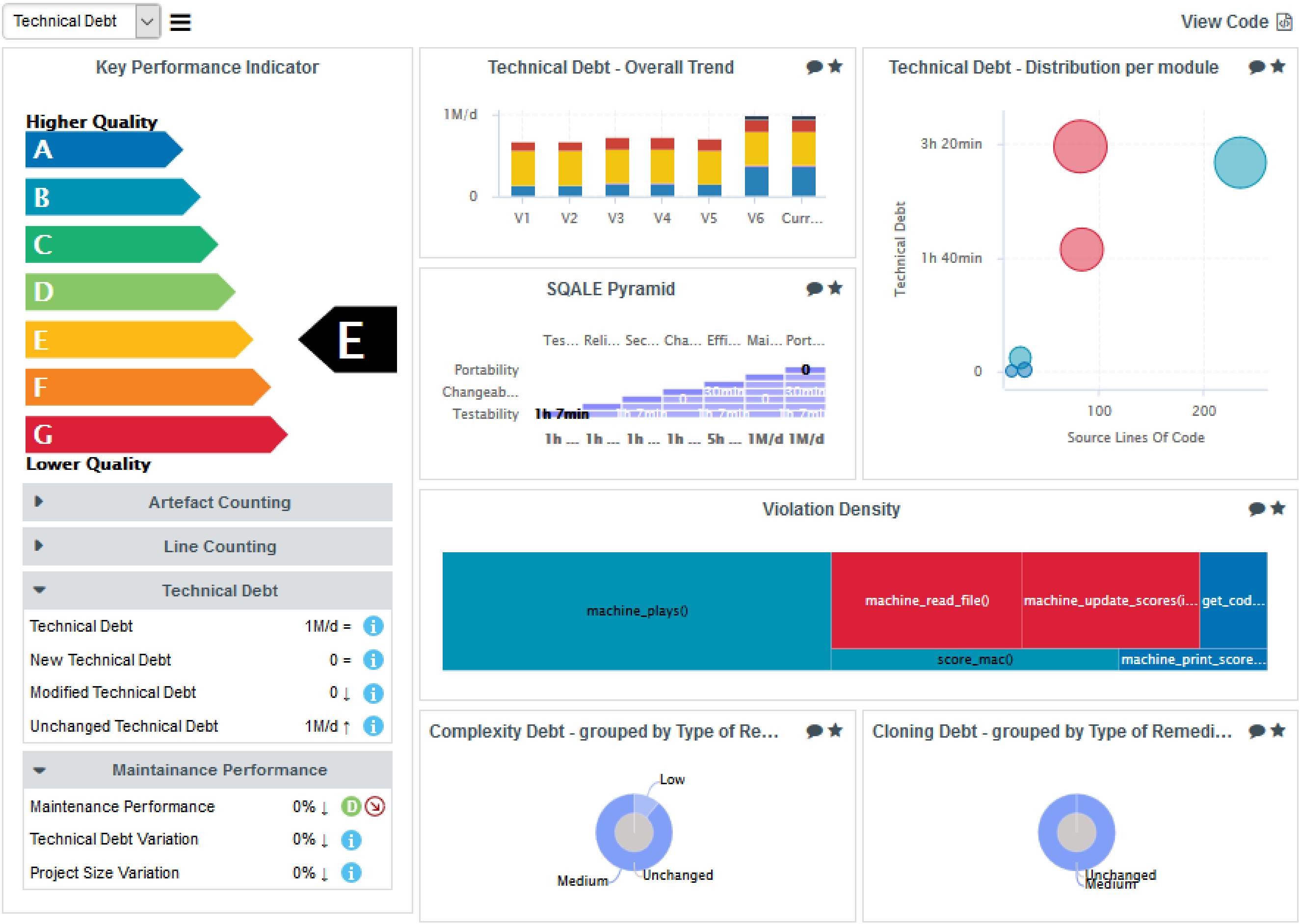
'Comment Density' Compliance	46% = G →
'Cyclomatic Complexity' Compliance	72% = G →
'Nested Level' Compliance	76% = G →
'Non-Cyclic Path' Compliance	84% = F →
GOTO Compliance	98% = C →
'RETURN Points' Compliance	94% = E →
'Language Scope' Compliance	56% = G →
'Number of Statements'' Compliance	82% = F →
'Number of Parameters' Compliance	100% = A →
'Number of Calling' Compliance	94% = E →
'Number of Calls' Compliance	88% = F →
'Recursion' Compliance	98% = C →

The score card area

Each table lines display a series of details about the key performance indicator:

- The name of the metric (e.g. **Rule Compliance**). When clicked, a popup shows the way the metric is computed. Optionally, some metrics may allow an extra link to be displayed. This link shows the list of findings taken into account when calculating this metric (See **Non Conformity Count**).
- The raw value of the metric and its evolution according to the previous version (e.g. **41.18% ↓**). Clicking a value ion this column displays a chart of the history of the last 10 values recorded for this metric.
- If the metric displayed is an indicator, the rating of the indicator is displayed, along with its evolution (e.g. **Level G, deteriorated**). If the metric is a measure, then an information icon is shown. In both cases, you can click the information in this column to display more details about how the metric is computed.

The right-hand area of the Dashboard contains a series of charts representing key information about the current artefact. Clicking a graph opens a larger version of the image so you can analyse the data. Note that the available charts will differ depending on the type of artefact selected in the tree. Files and functions also include a **Source Code** link at the top right corner (for users who have the privilege to browse source code), which does not appear in the Dashboard for folders and applications.



Artefact Counting

Line Counting

Technical Debt

Technical Debt	1M/d =
New Technical Debt	0 =
Modified Technical Debt	0 ↓
Unchanged Technical Debt	1M/d ↑

Maintainance Performance

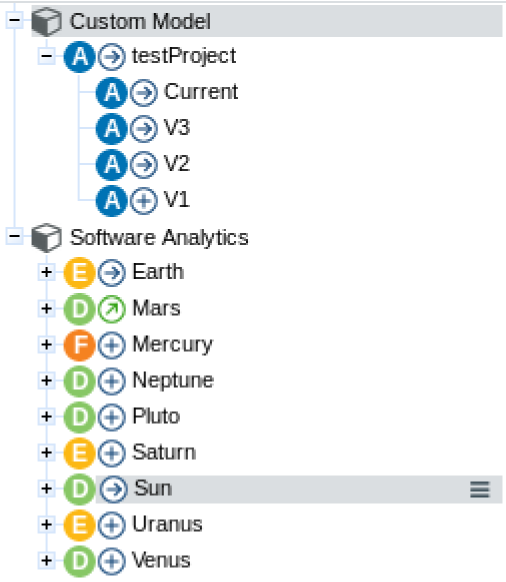
Maintenance Performance	0% ↓	D
Technical Debt Variation	0% ↓	
Project Size Variation	0% ↓	

The charts area

The Dashboard is only the first of a series of tabs in the Explorer. In the following chapter, you will find out more about the role of the **Action Items**, **Highlights**, **Findings**, **Forms**, **Reports**, **Indicators**, **Measures** and **Comments** tabs. Note however that like the Dashboard, the information displayed in each tab is always relative to the node currently selected in the Artefact Tree.

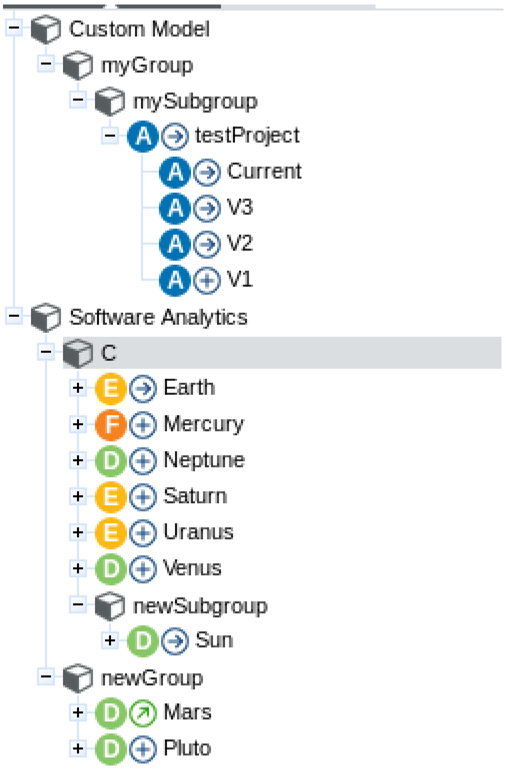
Organising Projects

By default, projects are sorted out according to their analysis model in the **Project Portfolios**.



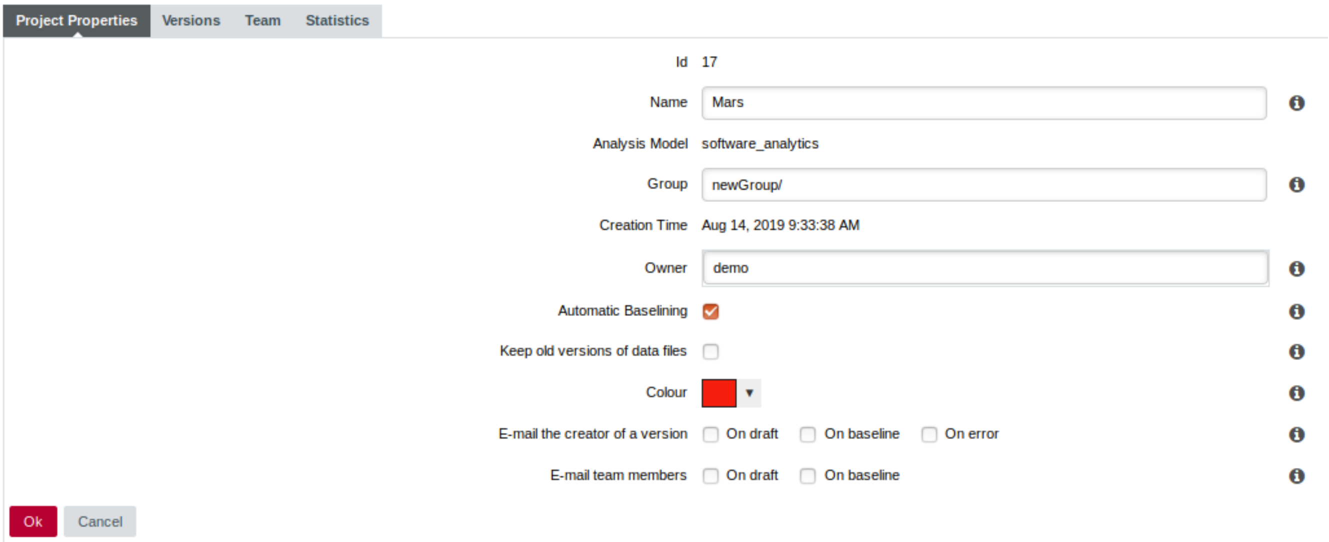
Project Portfolios default organization

But it is possible to create groups and subgroups in order to organise the project list.



Project Portfolios custom organization

To do so, click on "Manage" for a given project and in the "Project Properties" tab, fill in the "Group" field :



Id	17
Name	Mars
Analysis Model	software_analytics
Group	newGroup/
Creation Time	Aug 14, 2019 9:33:38 AM
Owner	demo
Automatic Baselining	☑
Keep old versions of data files	☐
Colour	🟥 ▾
E-mail the creator of a version	☐ On draft ☐ On baseline ☐ On error
E-mail team members	☐ On draft ☐ On baseline

Edit project's group

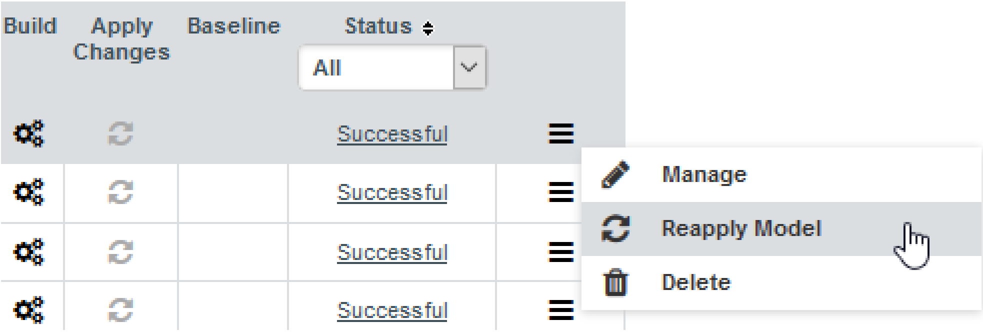
It is possible to create subgroups as well, just type in the full *path* to your group : "myGroup/mySubgroup".

To remove a project from a group just delete the content of the "Group" field. If a group becomes empty after removing its last project, it will be automatically removed from the **Project Portfolios**.

Reapply Model On Projects

This feature enables you to apply a new model, or an updated one, on an already existing project. This is done by using the project data in the database, without executing Repository Connectors and Data Providers.

Go to the "Projects" page, in the sandwich menu from the desired project, click "Reapply Model":



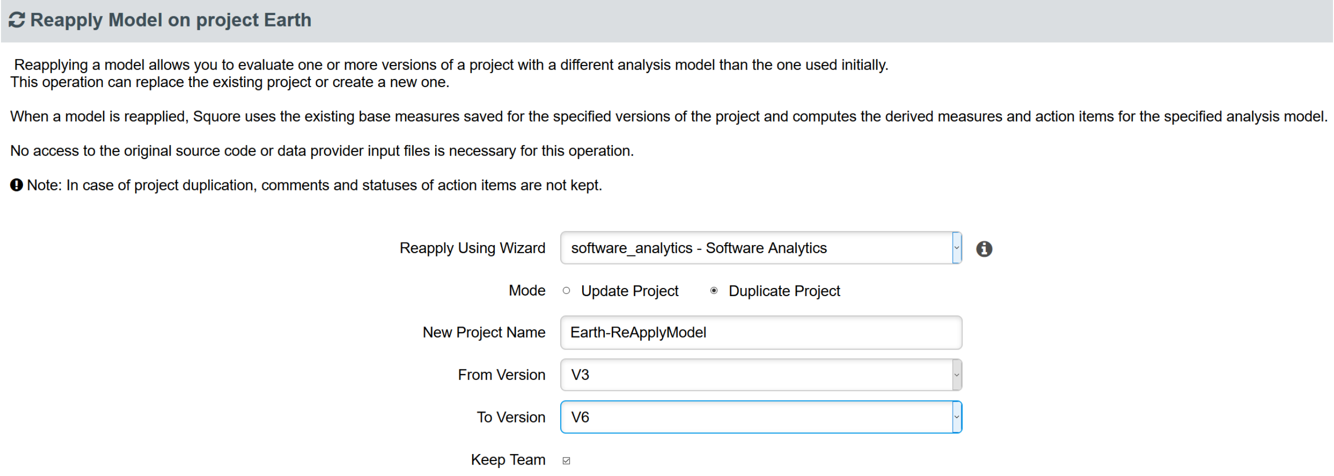
Reapply model on an existing project

In the Reapply Model menu page, choose the desired option:

- **Update Project**: will apply the new model onto the existing project, hence replacing it.
- **Duplicate Project**: will create a new project from the selected one and then apply the new model onto the duplicate.

In case of a duplication the following options are available :

- **New Project Name**: the name of the new project.
- **From Version**: the project version from which the reapply will start.
- **To Version**: the project version until which the reapply will be done.
- **Keep Team**: whether or not the new project keeps the same team setup as the existing one.



Reapply Model on project Earth

Reapplying a model allows you to evaluate one or more versions of a project with a different analysis model than the one used initially.
This operation can replace the existing project or create a new one.

When a model is reapplied, Squore uses the existing base measures saved for the specified versions of the project and computes the derived measures and action items for the specified analysis model.

No access to the original source code or data provider input files is necessary for this operation.

Note: In case of project duplication, comments and statuses of action items are not kept.

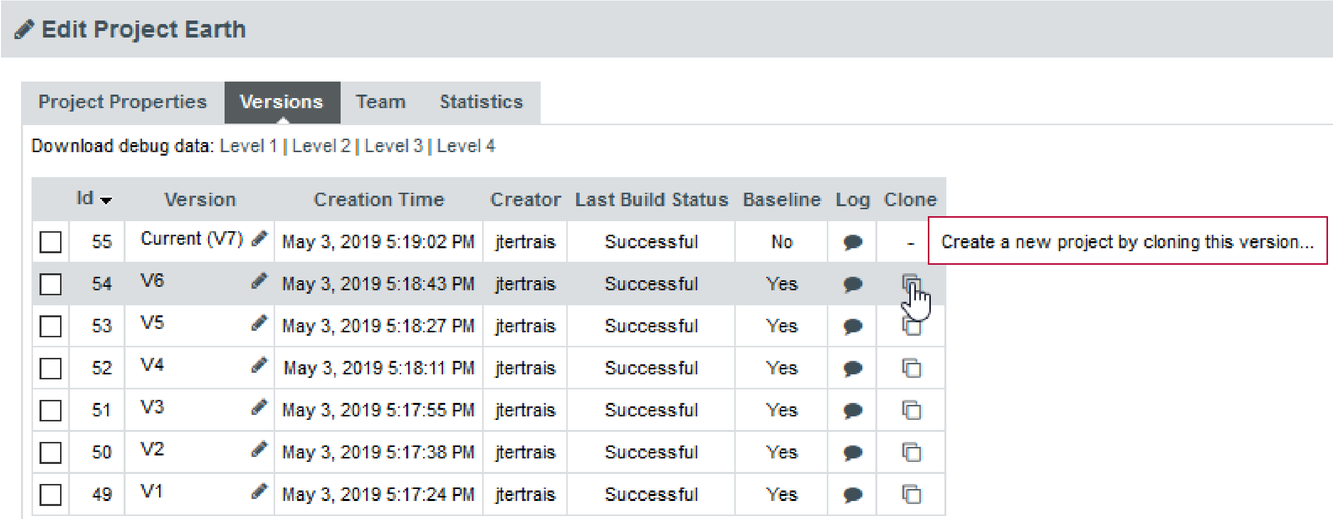
Reapply Using Wizard	software_analytics - Software Analytics
Mode	○ Update Project ● Duplicate Project
New Project Name	Earth-ReApplyModel
From Version	V3
To Version	V6
Keep Team	☑

Reapply model options

Cloning Projects

This feature enables you to branch a project while keeping all its version build information (relaxation states, milestones, etc...). First version of the new branched project is the one the clone has been created from.

Go to the "Projects" page, click "Manage" on the desired project, in the "Versions" tab a new "Clone" column is available:



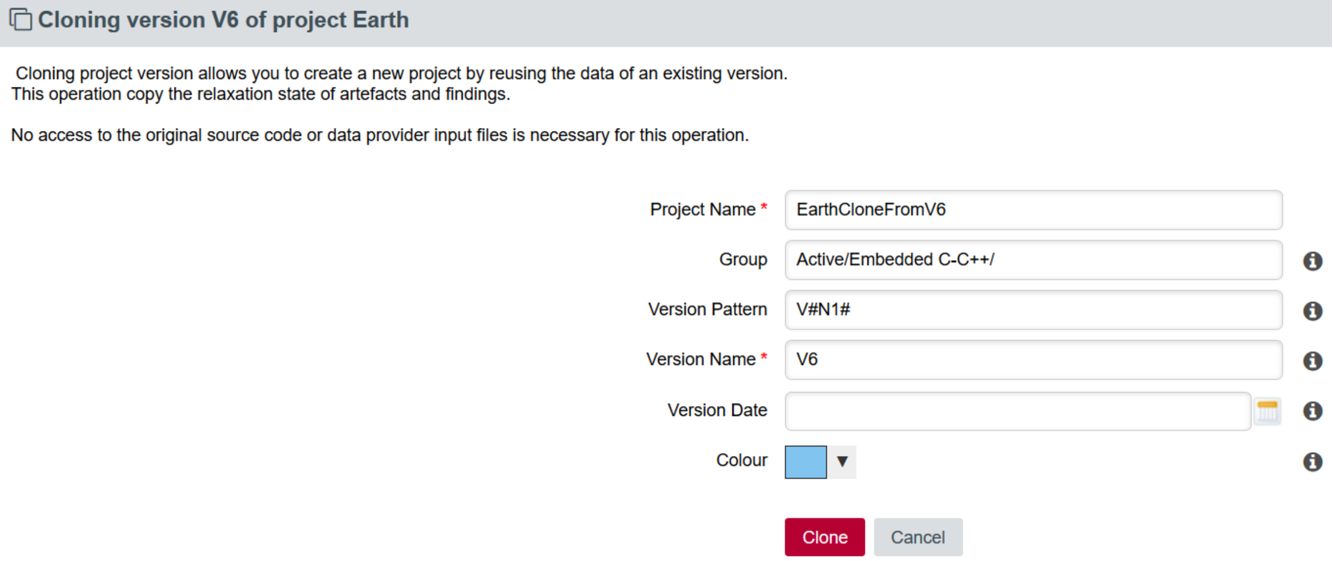
Edit Project Earth

Download debug data: Level 1 | Level 2 | Level 3 | Level 4

	Id	Version	Creation Time	Creator	Last Build Status	Baseline	Log	Clone
□	55	Current (V7)	May 3, 2019 5:19:02 PM	jtertrais	Successful	No	💬	-
□	54	V6	May 3, 2019 5:18:43 PM	jtertrais	Successful	Yes	💬	⧉
□	53	V5	May 3, 2019 5:18:27 PM	jtertrais	Successful	Yes	💬	⧉
□	52	V4	May 3, 2019 5:18:11 PM	jtertrais	Successful	Yes	💬	⧉
□	51	V3	May 3, 2019 5:17:55 PM	jtertrais	Successful	Yes	💬	⧉
□	50	V2	May 3, 2019 5:17:38 PM	jtertrais	Successful	Yes	💬	⧉
□	49	V1	May 3, 2019 5:17:24 PM	jtertrais	Successful	Yes	💬	⧉

Cloning a project from specific version

In the Clone project menu page, setup your project options:


Cloning version V6 of project Earth

Cloning project version allows you to create a new project by reusing the data of an existing version.
This operation copy the relaxation state of artefacts and findings.

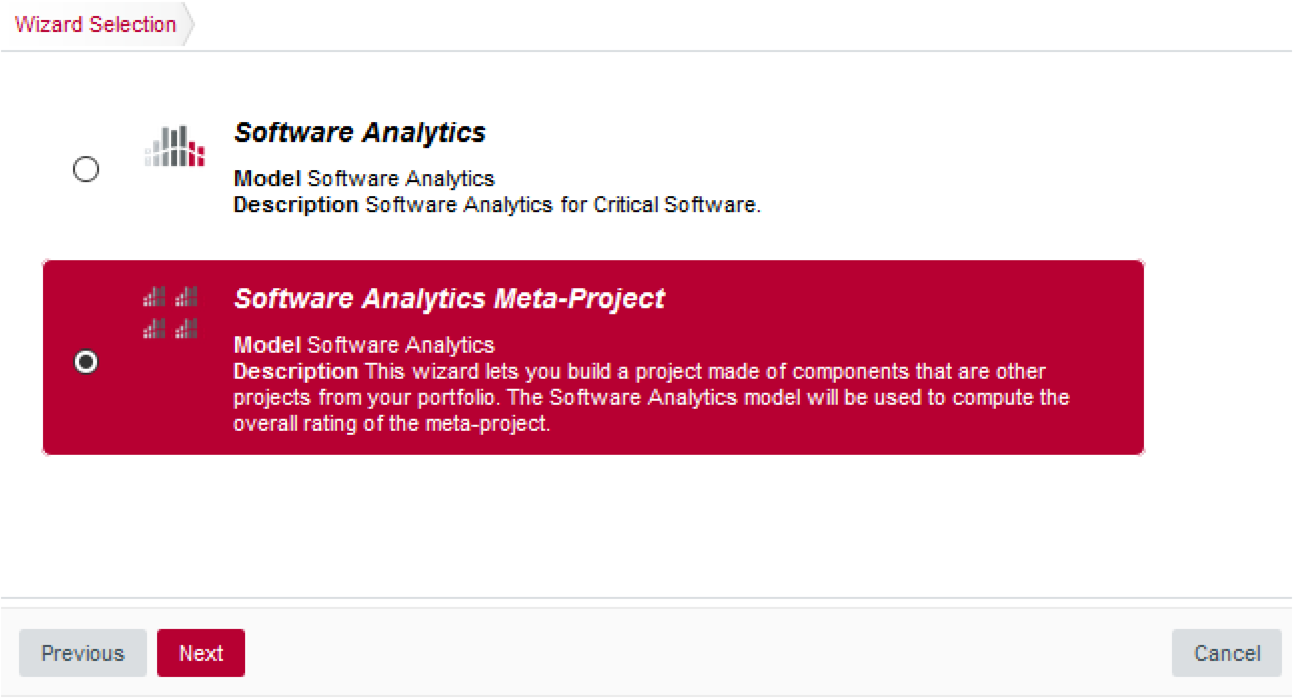
No access to the original source code or data provider input files is necessary for this operation.

Project Name *	EarthCloneFromV6
Group	Active/Embedded C-C++/
Version Pattern	V#N1#
Version Date	
Colour	▼

Setting-up project options

Creating Meta-Projects

In contexts where your projects reuse code from other projects that you also analyse in Squore, you can create a meta-project that will show the analysis results from the various software bricks in a single project.



○ **Software Analytics**

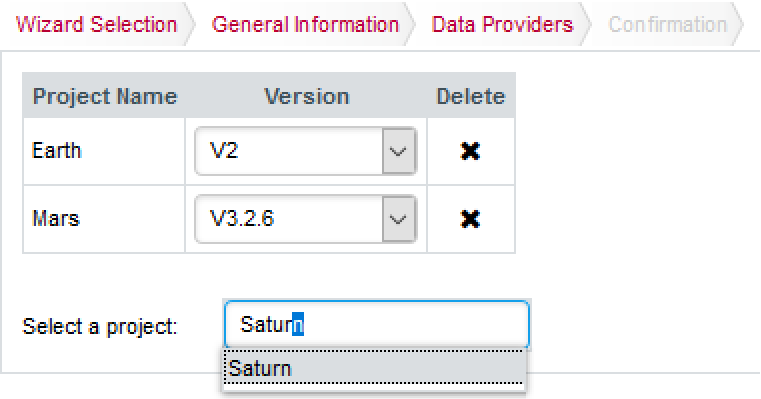
Model Software Analytics
Description Software Analytics for Critical Software.

● **Software Analytics Meta-Project**

Model Software Analytics
Description This wizard lets you build a project made of components that are other projects from your portfolio. The Software Analytics model will be used to compute the overall rating of the meta-project.

Selecting a wizard for building meta-projects

When you select a wizard that allows to create a meta-project, you do not have to specify any locations for source files or any data providers to run. Instead, you are presented with a project picker that allows you to tell Squore which sub-projects compose your meta-project. In the example below, we will create a project that uses Earth (V2), Mars (v3.2.6), and Saturn (Prel) as its parts. You can choose any baseline version of any project you have access to in Squore as a component of your meta-project.



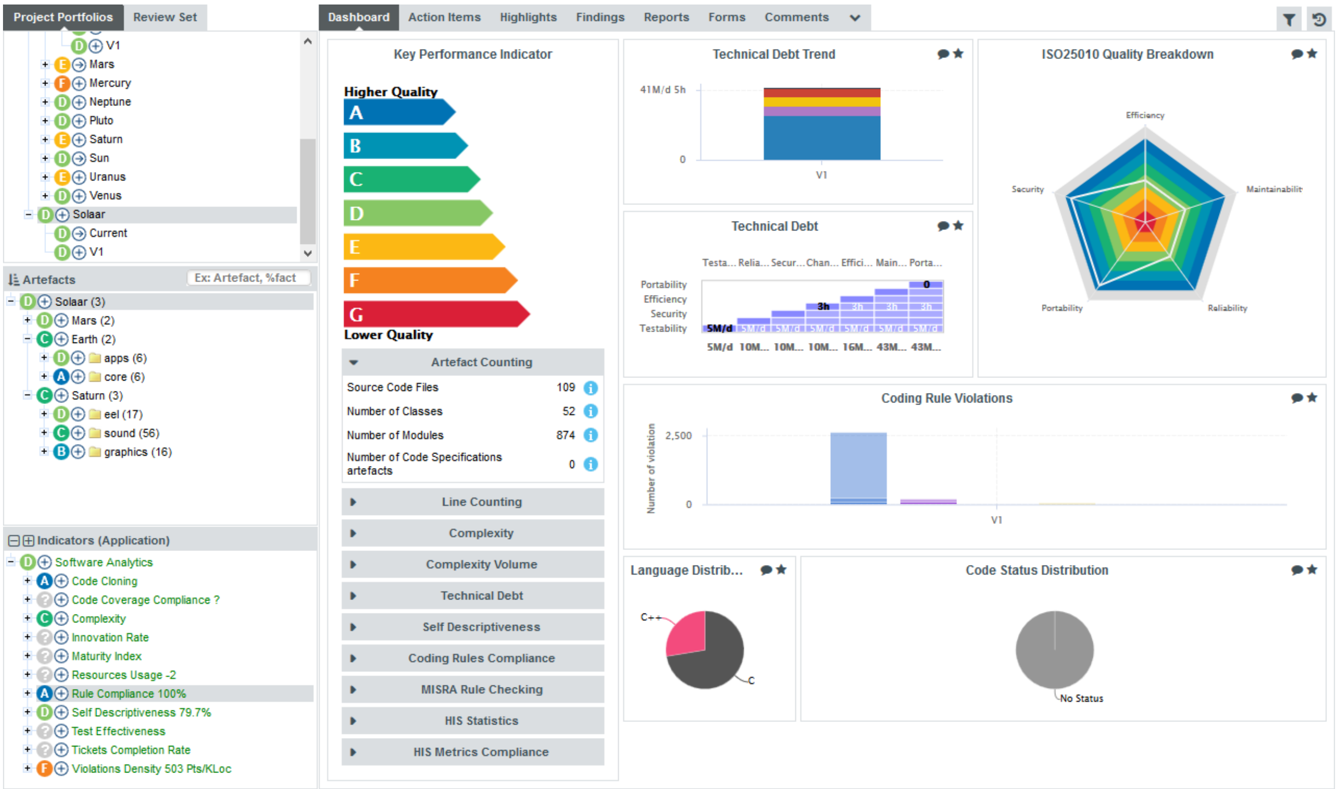
Project Name	Version	Delete
Earth	V2	✖
Mars	V3.2.6	✖

Select a project: Saturn

Saturn

Building a new meta-project with code from Earth, Mars and Saturn

When the analysis finishes, the meta-project is listed along with the other projects in the Project Portfolios. You can expand the Artefact Tree for your meta-project to browse the artefacts of the three sub-projects that are part of your meta-project, and consult all their Action Items, Findings and Highlights.



- D V1
- E Mars
- F Mercury
- D Neptune
- D Pluto
- E Saturn
- D Sun
- E Uranus
- D Venus
- D Solaar
  - D Current
  - D V1

Artefacts Ex: Artefact, %fact

- D Solaar (3)
  - D Mars (2)
  - C Earth (2)
    - D apps (6)
    - A core (6)
  - C Saturn (3)
    - D eel (17)
    - C sound (56)
    - B graphics (16)

Indicators (Application)

- D Software Analytics
  - A Code Cloning
  - ? Code Coverage Compliance ?
  - C Complexity
  - ? Innovation Rate
  - ? Maturity Index
  - ? Resources Usage -2
  - A Rule Compliance 100%
  - D Self Descriptiveness 79.7%
  - ? Test Effectiveness
  - ? Tickets Completion Rate
  - F Violations Density 503 Pts/KLoc

Key Performance Indicator

Artefact Counting

Source Code Files	109
Number of Classes	52
Number of Modules	874
Number of Code Specifications artefacts	0

- Line Counting
- Complexity
- Complexity Volume
- Technical Debt
- Self Descriptiveness
- Coding Rules Compliance
- MISRA Rule Checking
- HIS Statistics
- HIS Metrics Compliance

The dashboard for the meta-project Solar, composed of Earth, Mars and Saturn

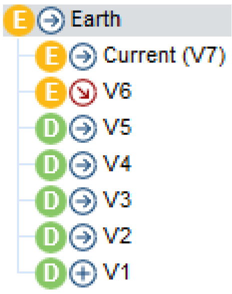
Chapter 5. Understanding Analysis Results

This chapter describes the main features available in the Explorer. By the end of the chapter, you should be able to make the most of the information and decisions presented by Squore and start applying them to improve your development practices.

Has the Quality of My Project Decreased Since the Previous Analysis?

After completing the analysis of a new version of your project, you will probably want to investigate how it has evolved, more specifically for which artefacts the quality has decreased. Let's look at the history of the Earth Project (which should be available if your Squore administrator has created the sample projects shipped with the Squore installation) to find out how to spot the worst-scored components in your project.

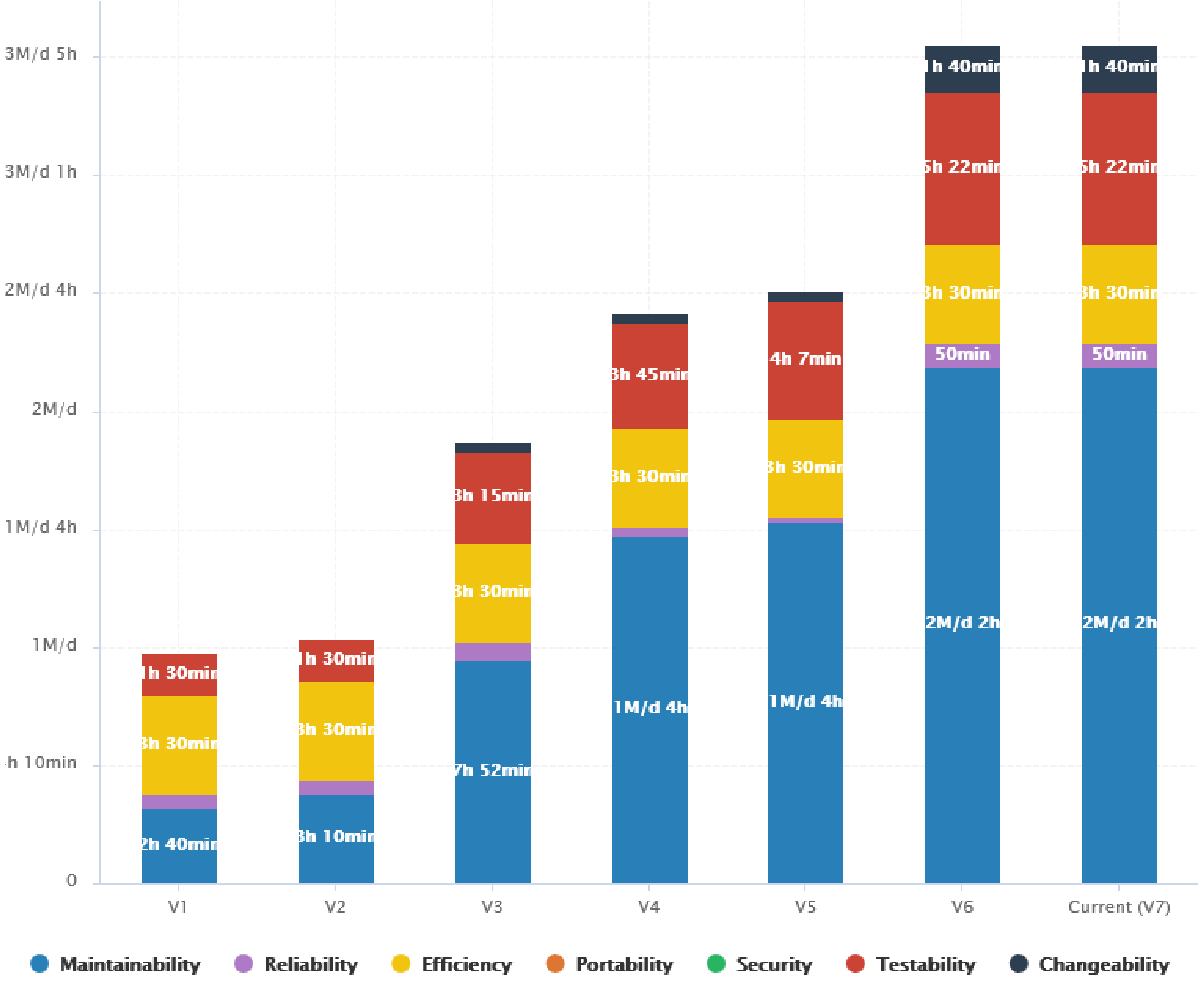
Log into Squore as the demo user using *demo/demo* and observe the evolution of the Earth project in the Project Portfolios:



E → Earth
E → Current (V7)
E ↘ V6
D → V5
D → V4
D → V3
D → V2
D + V1

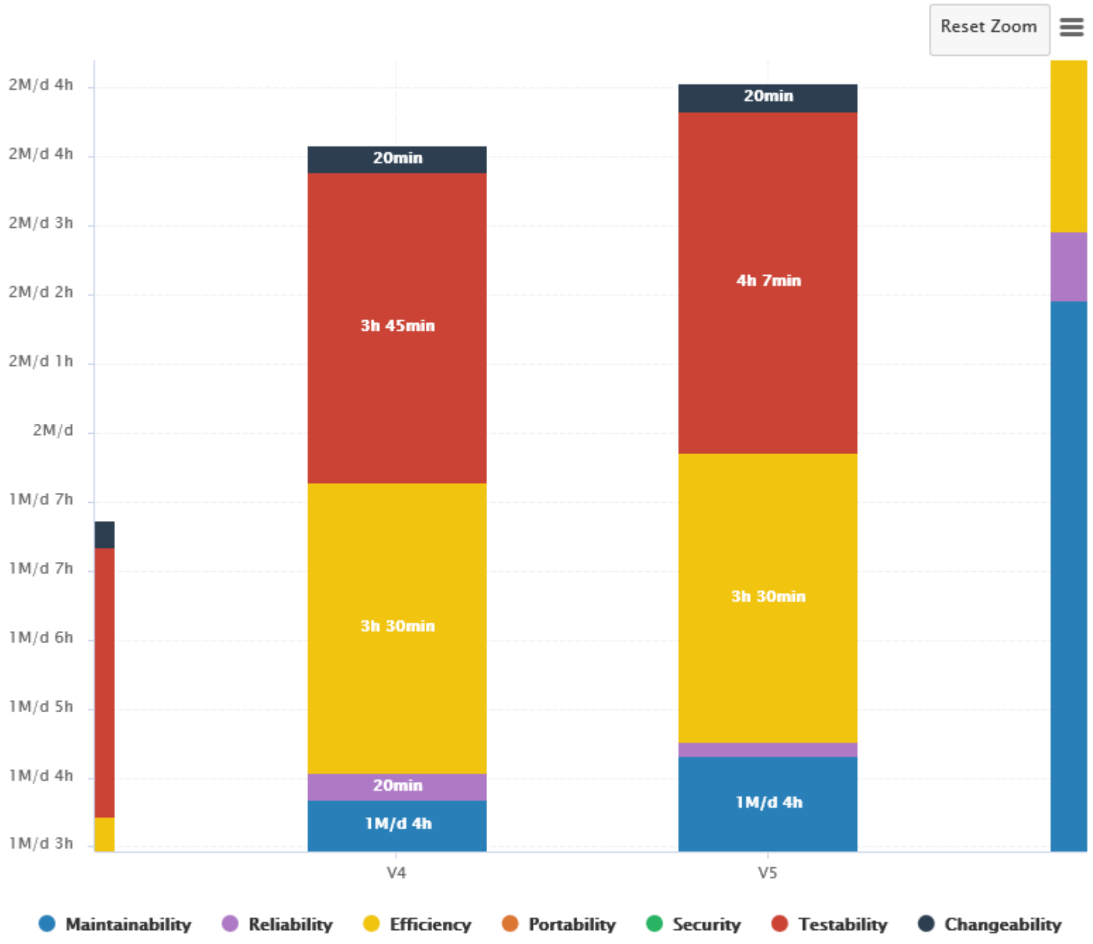
The versions of the Earth Project

The trend arrows before the version names in the Project Portfolios indicate that the overall rating has recently deteriorated (More information about the quality indicator icons is available in Where Are My Analysis Results?). If you take a closer look at the Technical Debt Trend chart, you can notice that the technical debt is growing for this project.



The Technical Debt Trend chart for the latest version of Earth

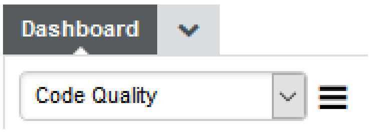
You can zoom in on the chart by dragging your mouse over the section of the data you are interested in. After zooming in, you can keep the selected zoom level and move around the chart by holding the **Shift** key and dragging the chart around, or go back to the original view by clicking the **Reset Zoom** button in the Chart Viewer.



A zoomed-in section of the Technical Debt Trend chart

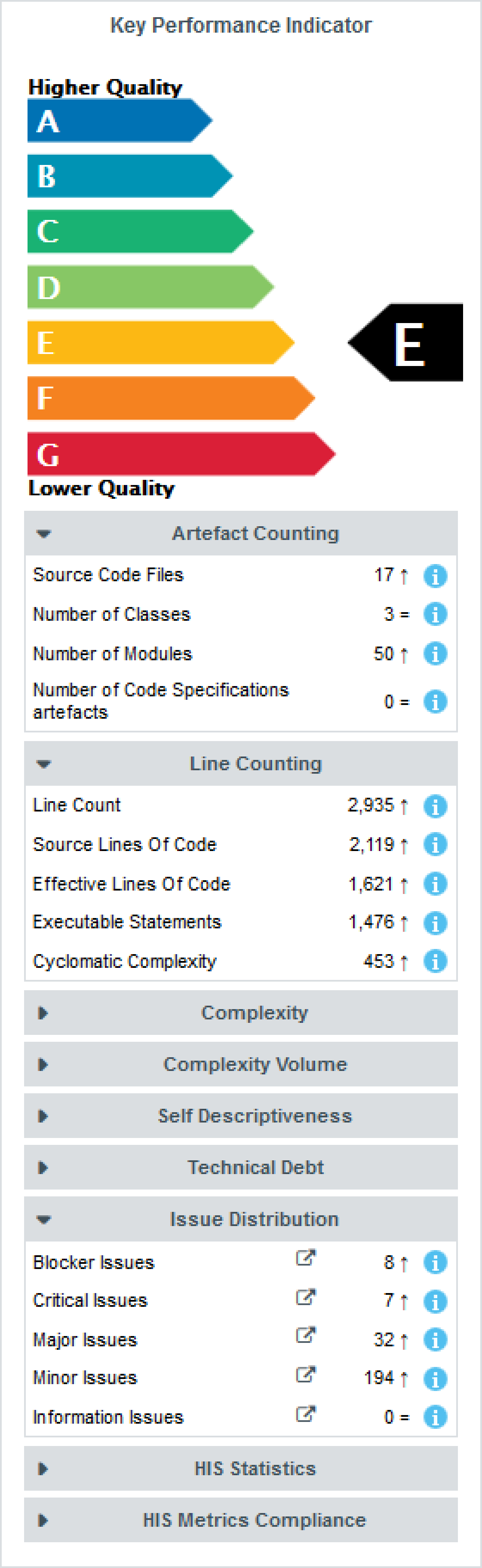
Since the trend accelerated in V6, we will focus on this version. Let's click V6 in the tree and start our evaluation by looking at the score card, which rates Earth at E.

Let's also look at the **Code Quality** dashboard, which offers more insight into coding violations. Select **Code Quality** in the Dashboard drop-down list:



The Dashboard drop-down list allows you to see all the dashboards that you have access to

Some values under **Artefact Counting**, **Line Counting** and **Issue Distribution** explain the lower score: the application contains more files and functions, more lines of code, less comments and more rules violations .



Key Performance Indicator

Artefact Counting	
Source Code Files	17 ↑
Number of Classes	3 =
Number of Modules	50 ↑
Number of Code Specifications artefacts	0 =

Line Counting	
Line Count	2,935 ↑
Source Lines Of Code	2,119 ↑
Effective Lines Of Code	1,621 ↑
Executable Statements	1,476 ↑
Cyclomatic Complexity	453 ↑

Complexity

Complexity Volume

Self Descriptiveness

Technical Debt

Issue Distribution	
Blocker Issues	8 ↑
Critical Issues	7 ↑
Major Issues	32 ↑
Minor Issues	194 ↑
Information Issues	0 =

HIS Statistics

HIS Metrics Compliance

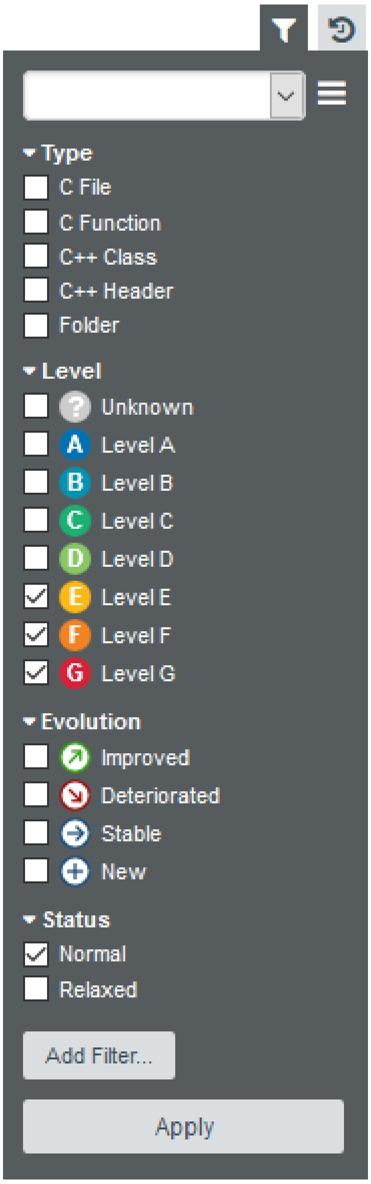
The score card for the version V6 of Earth

By now you probably want to find out which components in your project are responsible for increasing the technical debt the application in this version. If you want the Artefact Tree to reflect this information, you can change the sort order to show the worst scores first by clicking on the **sort** icon (↓☰) and selecting **LEVEL > Worst first** to display artefacts from worst-scored to best-scored.

Finding Artefacts Using Filters and Search

This section explains the basics of looking for artefacts using filters and search. For a more automated way to find artefact that fit a specific category, take a look at Finding Artefacts Using Highlights.

Click the Filter icon (▼) in the top-right corner of the Explorer to reveal the Filter Panel. The Filter Panel allows you to set criteria that artefacts need to meet in order to be displayed in the artefact tree or taken into account in charts on the dashboard. For this example, we want to restrict the visibility of artefacts to those rated E and lower:



Type
- ☐ C File
- ☐ C Function
- ☐ C++ Class
- ☐ C++ Header
- ☐ Folder

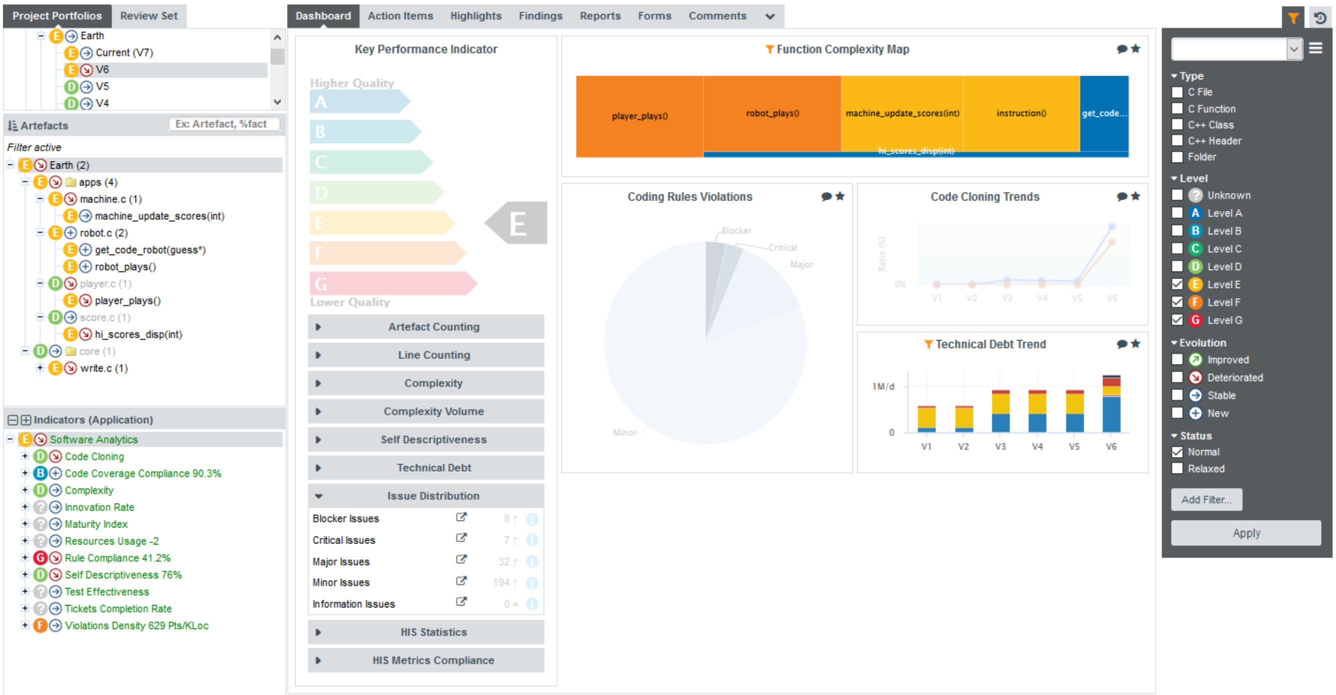
Level
- ☐ ? Unknown
- ☐ A Level A
- ☐ B Level B
- ☐ C Level C
- ☐ D Level D
- ☑ E Level E
- ☑ F Level F
- ☑ G Level G

Evolution
- ☐ Improved
- ☐ Deteriorated
- ☐ Stable
- ☐ New

Status
- ☑ Normal
- ☐ Relaxed

The Filter Panel with the boxes checked to filter artefacts rated E and lower

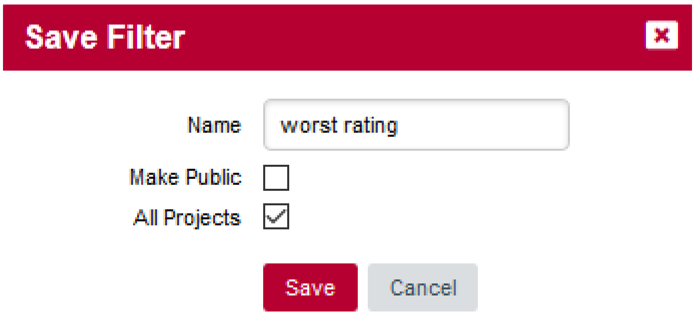
Click **Apply** to apply the changes The Artefact Tree and the dashboard refresh to show results for the artefacts in the levels selected, as shown below:



The filtered Explorer showing artefacts rated E and lower

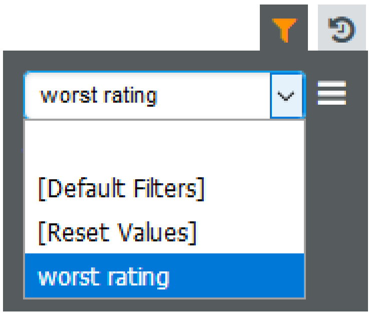
The notice **Filter active** is always displayed above the Artefact Tree and the filter icon turns orange when you are using a filter. The tree now only shows artefacts rated E and lower, along with their ancestors, which are greyed out if they are not rated E and lower. On the dashboard, the charts that support displaying filtered information are highlighted with an orange filter icon (▼) as in the **Function Complexity Map**. Other charts and score card items that cannot be filtered are greyed out.

You can save your filter for later use and even share it with other Squore users by clicking the hamburger menu in the Filter Panel and giving your filter a name.



The filter saving options

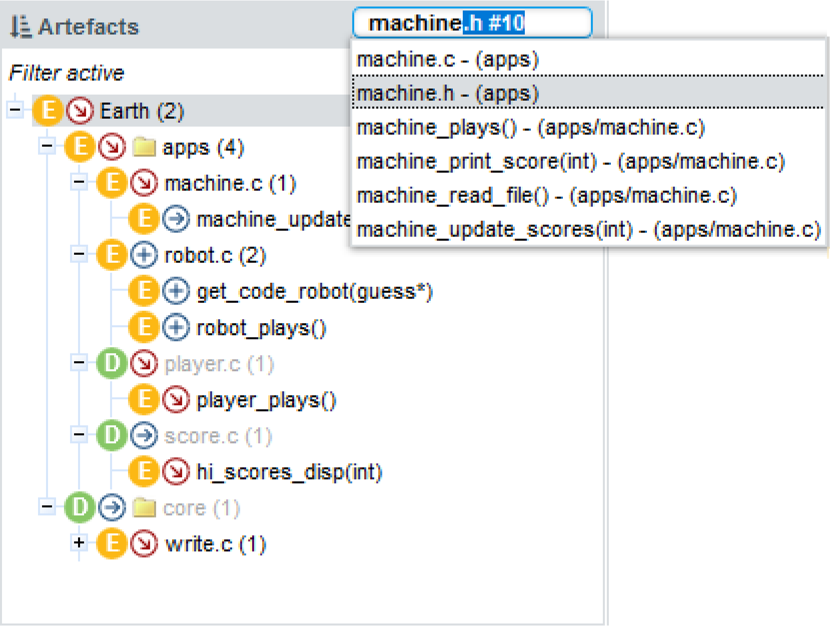
Saved filters are displayed in the dropdown list at the top of the Filter Panel so you can reuse them later.



The list of saved filters

For more details about advanced filtering functionality in Squore, consult Advanced Filtering.

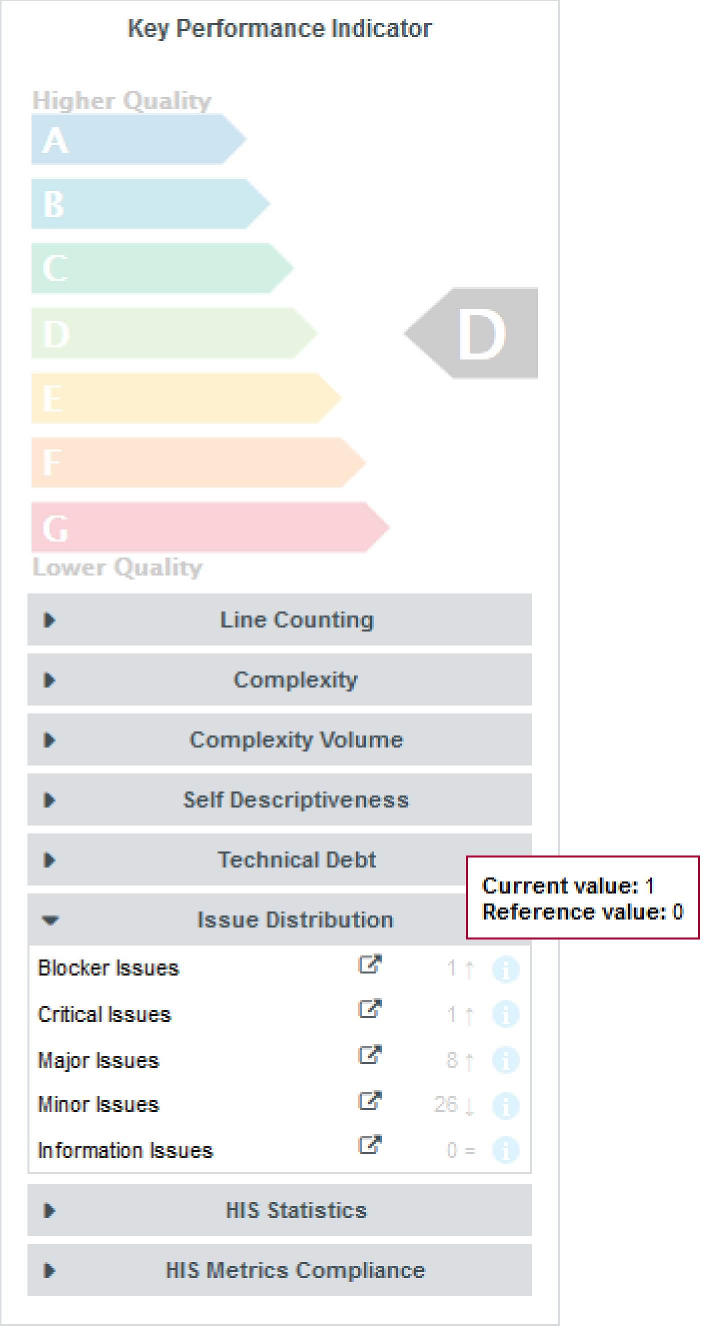
While a filter is active, you can still search for other artefacts by typing a search term in the search box. Try typing **ma** in the search box above the Artefact Tree, and watch the search results list get populated as you type:



The search results for the term entered in the search box

If you select the highlighted search result in the list above, you will open the dashboard for *machine.h*.

Let's go back to our filtered tree. The filter singled out three files whose rating deteriorated that contain functions in the required score range: *player.c*, *machine.c* and *write.c*. Click on the artefact *player.c* to view its dashboard. Note how the score card indicates that the artefact has more critical issues than in the previous analysis.



Key Performance Indicator

Line Counting

Complexity

Complexity Volume

Self Descriptiveness

Technical Debt

Issue Distribution

Blocker Issues	1 ↑
Critical Issues	1 ↑
Major Issues	8 ↑
Minor Issues	26 ↓
Information Issues	0 =

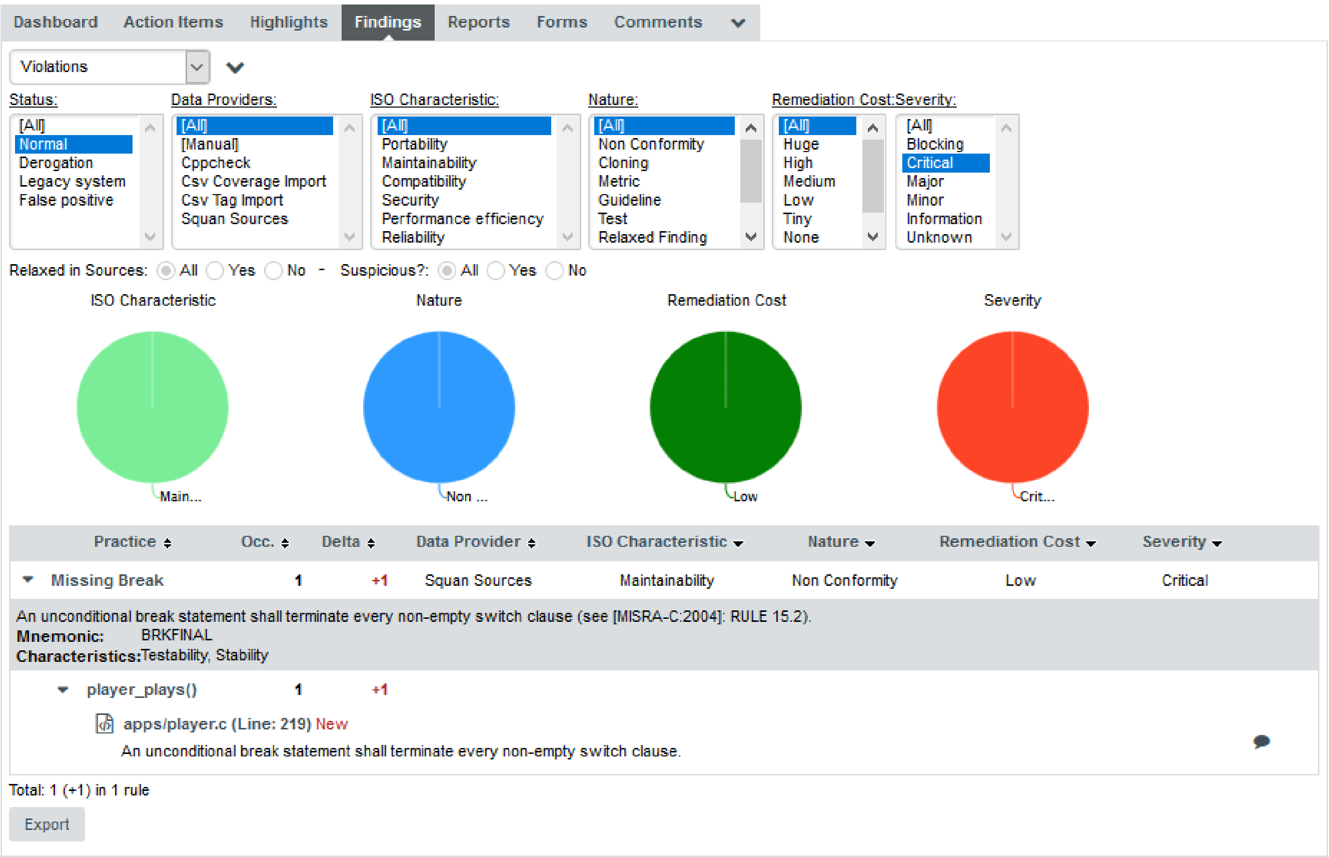
Current value: 1
Reference value: 0

HIS Statistics

HIS Metrics Compliance

One more critical issue for player.c in this version

You can click the link in the table to directly view the new critical violation on the Findings tab. In this case, the rule **BRKFINAL** was broken:



Status: [All] / **Normal** / Derogation / Legacy system / False positive

Data Providers: **[All]** / [Manual] / Cppcheck / Csv Coverage Import / Csv Tag Import / Squan Sources

ISO Characteristic: **[All]** / Portability / Maintainability / Compatibility / Security / Performance efficiency / Reliability

Nature: **[All]** / Non Conformity / Cloning / Metric / Guideline / Test / Relaxed Finding

Remediation Cost: **[All]** / Huge / High / Medium / Low / Tiny / None

Severity: [All] / Blocking / **Critical** / Major / Minor / Information / Unknown

Relaxed in Sources: ● All ○ Yes ○ No - Suspicious?: ● All ○ Yes ○ No

Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Missing Break	1	+1	Squan Sources	Maintainability	Non Conformity	Low	Critical

An unconditional break statement shall terminate every non-empty switch clause (see [MISRA-C:2004]: RULE 15.2).
Mnemonic: BRKFINAL
Characteristics: Testability, Stability

player_plays() 1 +1

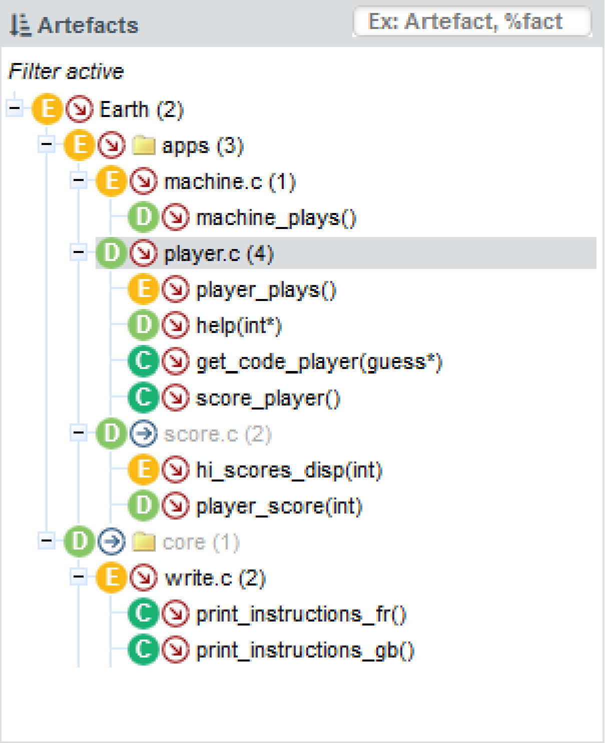
apps/player.c (Line: 219) New
An unconditional break statement shall terminate every non-empty switch clause.

Total: 1 (+1) in 1 rule

A new critical issue for player.c in this version

Another convenient way to try and find why a project's quality is deteriorating is to filter on the trend of an artefact.

Select version V6 of Earth again and edit the active filter: Uncheck the boxes for levels E and lower, and select the **Deteriorated** category in the Evolution section. When applying the filter, you should see the artefacts in the tree that have the ⦪ icon next to their name.



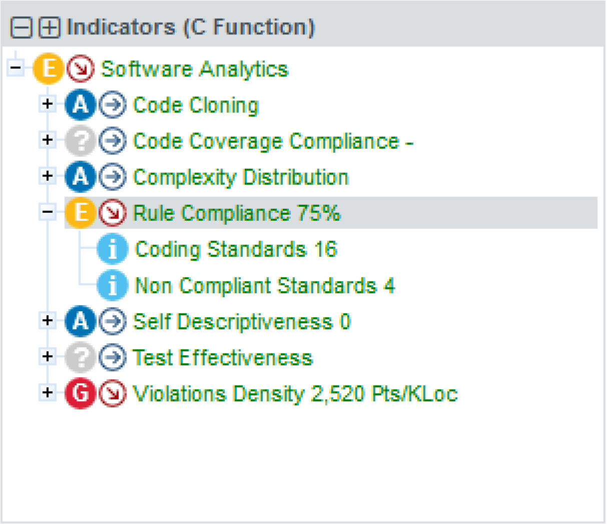
Filter active

- E Earth (2)
  - E apps (3)
    - E machine.c (1)
      - D machine_plays()
    - D player.c (4)
      - E player_plays()
      - D help(int*)
      - C get_code_player(guess*)
      - C score_player()
    - D score.c (2)
      - E hi_scores_disp(int)
      - D player_score(int)
  - D core (1)
    - E write.c (2)
      - C print_instructions_fr()
      - C print_instructions_gb()

The artefacts that deteriorated in version V6 of Earth

The files you inspected earlier are still here, but there are more deteriorated artefacts that you can start reviewing. If you click on *hi_scores_disp(int)* for example, which is rated E.

In order to find out where the degradation took place, you can look at the indicators tree to understand where the decline in quality comes from. Expand all the nodes in the indicators tree to reveal the issues with the artefact:



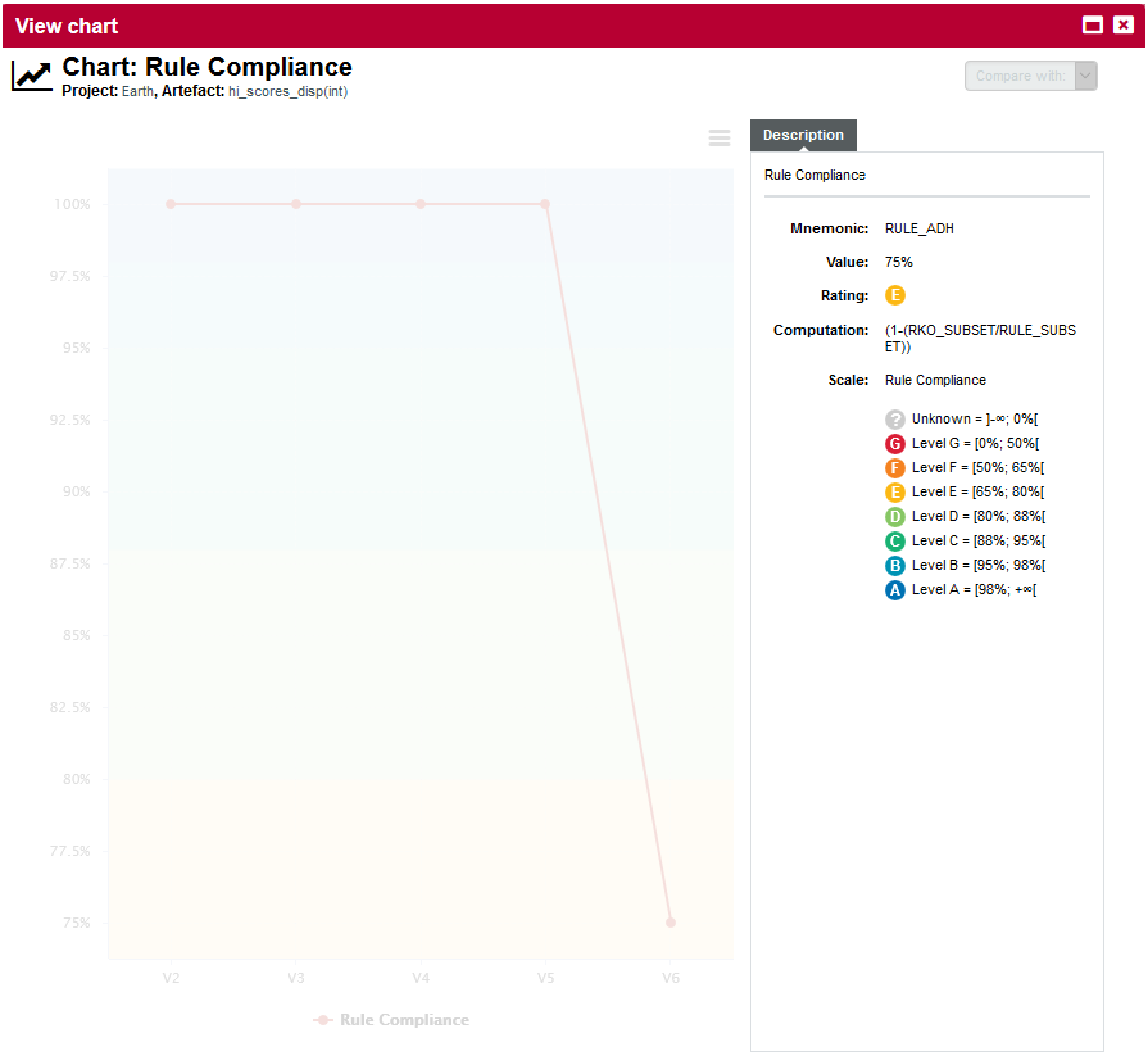
Indicators (C Function)

- E Software Analytics
  - A Code Cloning
  - ? Code Coverage Compliance -
  - A Complexity Distribution
  - E Rule Compliance 75%
    - i Coding Standards 16
    - i Non Compliant Standards 4
  - A Self Descriptiveness 0
  - ? Test Effectiveness
  - G Violations Density 2,520 Pts/KLoc

The Indicator Tree for hi_scores_disp(int)

Squore makes it easy to spot the irregularities quickly, like the fact that the **Rule Compliance** indicator is one of the causes for the worse score in this version. This is probably the first item to review in this function.

By clicking the Rule Compliance indicator in the tree, you can learn more about its history and how it is computed:



View chart

Chart: Rule Compliance

Project: Earth, **Artefact:** hi_scores_disp(int)

Rule Compliance

Mnemonic: RULE_ADH

Value: 75%

Rating: E

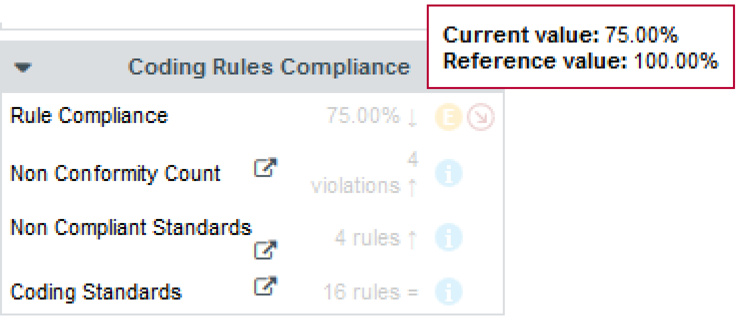
Computation: (1-(RKO_SUBSET/RULE_SUBSET))

Scale: Rule Compliance

- ? Unknown =]-∞; 0%[
- G Level G = [0%; 50%[
- F Level F = [50%; 65%[
- E Level E = [65%; 80%[
- D Level D = [80%; 88%[
- C Level C = [88%; 95%[
- B Level B = [95%; 98%[
- A Level A = [98%; +∞[

The description and history of the Rule Compliance indicator for hi_scores_disp(int)

Finally, you can take a look at the Coding Rules Compliance section of the artefact's scorecard to confirm the results:



Coding Rules Compliance		
Rule Compliance	75.00% ↓	E
Non Conformity Count	4 violations ↑	
Non Compliant Standards	4 rules ↑	
Coding Standards	16 rules =	

Current value: 75.00%
Reference value: 100.00%

The scorecard for hi_scores_disp(int)

By clicking the link icon, you can directly view the violations for this artefact in the Findings tab.

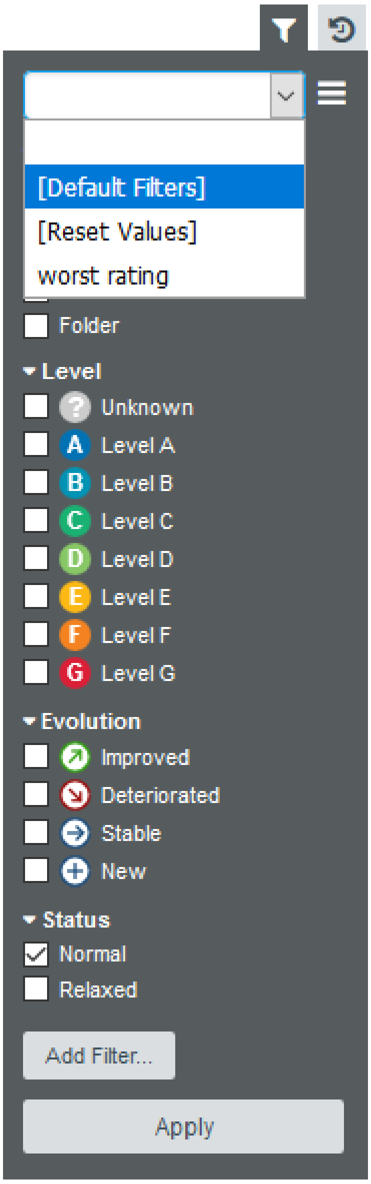
You can dive further into the analysis results by looking at the information contained in other tabs and assign action items to your team by referring to Managing Your To-Do List With Squore or report your progress as explained in Communicating With Squore.

Advanced Filtering

In the previous section, you learned how to filter artefacts based on their overall rating or trend, but Squore allows you to filter on more than the artefact's main indicator. This section covers filtering on more than one metric, and how you can also save and share your filters with other Squore users.

This example explores how to develop a strategy to reduce cloning in your application by finding the artefacts with the highest cloning that have also been modified since the previous analysis.

Start by displaying the dashboard for version V6 of Earth. In the Filter Panel, clear any existing filter by selecting **[Default Filters]** and clicking **Apply** to go back to an unfiltered dashboard.



□ Folder

Level

□ Unknown
□ Level A
□ Level B
□ Level C
□ Level D
□ Level E
□ Level F
□ Level G

Evolution

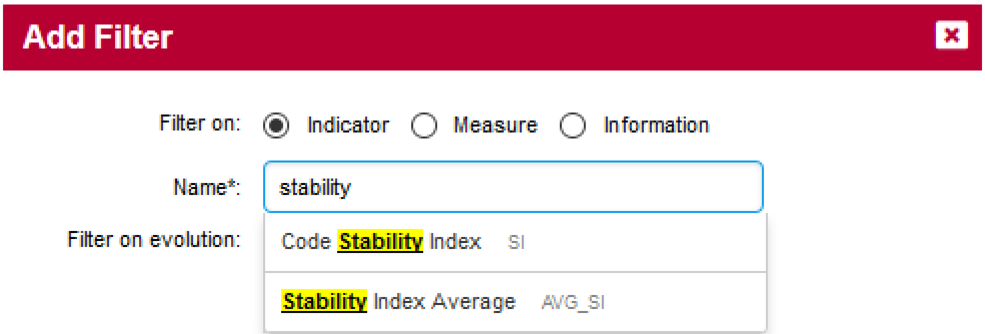
□ Improved
□ Deteriorated
□ Stable
□ New

Status

☑ Normal
□ Relaxed

*The **[Default Filters]** in the Filter Panel*

In order to find artefacts that have recently been modified, use Squore's Stability Index indicator, a feature of the source code parser that computes the amount of changes in a file since the previous analysis. Add a filtering criterion by clicking **Add Filter...** in the Filter Panel. A popup appears so you can type the name of the indicator to filter on. Type **stability** and select **Code Stability Index** as shown below:



Add Filter

Filter on: ● Indicator ○ Measure ○ Information

Name*: stability

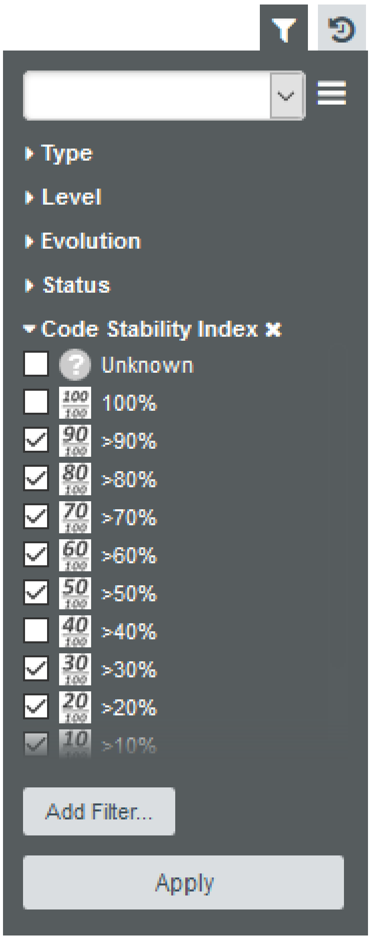
Filter on evolution:

Code Stability Index SI

Stability Index Average AVG_SI

The Filter Panel showing metrics whose names match stability

Clicking **Add** updates the Filter Panel with a section where you can select **Code Stability Index** levels to filter on. Since 100% means no changes, select all the levels except for 100%:

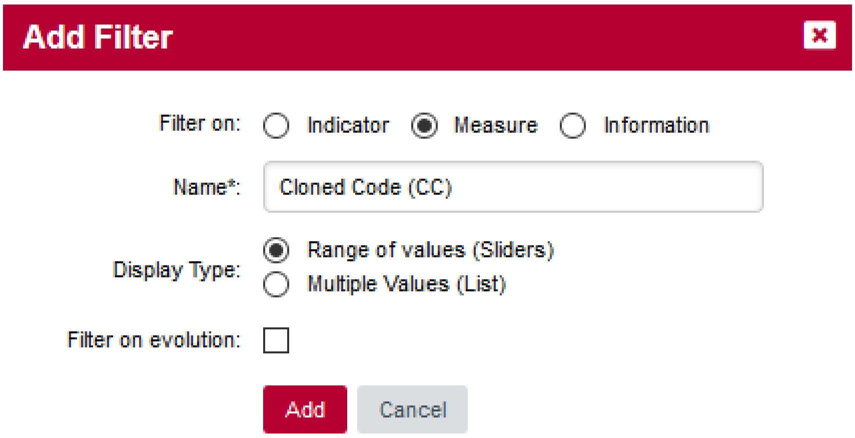


Code Stability Index

- □ Unknown
- □ 100%
- ☑ >90%
- ☑ >80%
- ☑ >70%
- ☑ >60%
- ☑ >50%
- □ >40%
- ☑ >30%
- ☑ >20%
- ☑ >10%

The Code Stability Index filter in the Filter Panel (other filters minimised for clarity)

In order to find artefacts with the worst cloning, click **Add Filter...**, select **Measure** and find the metric called **Cloned Code (CC)**. Configure this filter as a slider so we can dynamically change the range of cloning to look out for:



Filter on: ○ Indicator ● Measure ○ Information

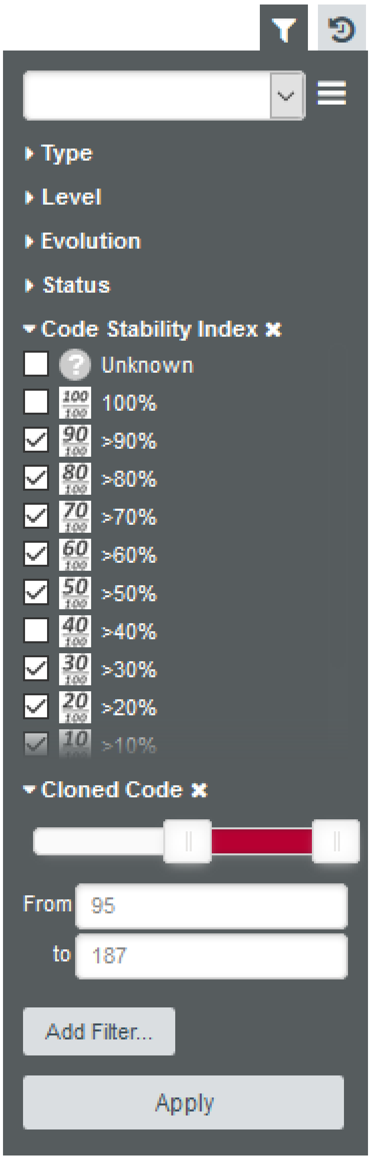
Name*: Cloned Code (CC)

Display Type: ● Range of values (Sliders) ○ Multiple Values (List)

Filter on evolution: □

The slider option selected for the Code Cloning metric in the filter popup

Click **Add** to add your filter. The slider shows you the minimum and maximum values for cloning in this project. Values can be modified either by entering values in the text boxes or by moving the sliders left and right. Move the left slider to about half the range for Code Cloning.



Code Stability Index ✖

- □ Unknown
- □ 100%
- ☑ >90%
- ☑ >80%
- ☑ >70%
- ☑ >60%
- ☑ >50%
- □ >40%
- ☑ >30%
- ☑ >20%
- ☑ >10%

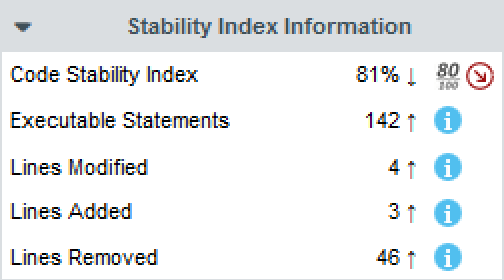
Cloned Code ✖

From 95

to 187

The slider for Code Cloning in the Filter Panel

When clicking **Apply** to view the results of your filter, the artefact *player.c* is singled out as a recently modified file with high cloning. By switching to the **Software Safety** dashboard, you can view details about its stability index in the score card.

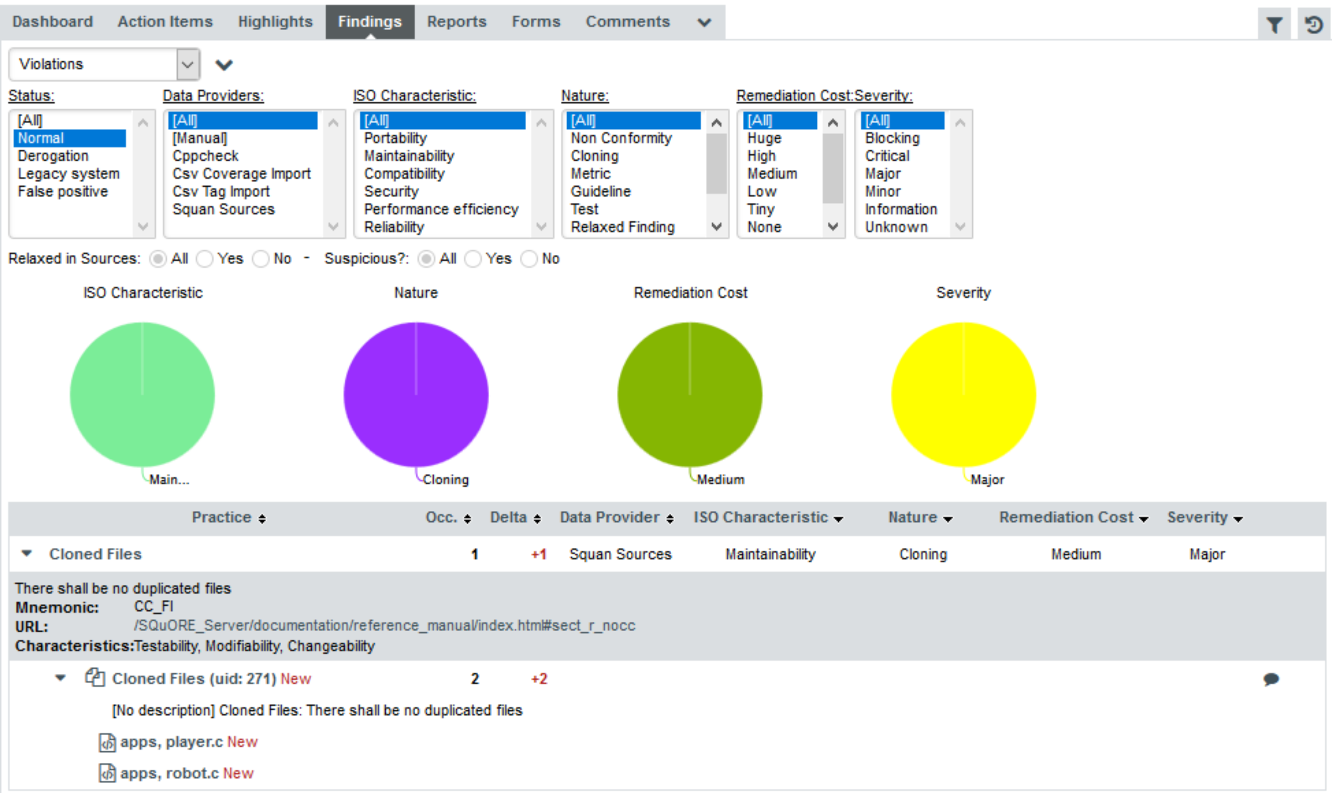


Stability Index Information

Code Stability Index	81% ↓ 80/100
Executable Statements	142 ↑
Lines Modified	4 ↑
Lines Added	3 ↑
Lines Removed	46 ↑

The Stability Index Information for player.c

Clicking the Findings tab allows you to confirm that the cloning detected by Squan Sources was indeed introduced since the latest analysis.



Status: [All], Normal, Derogation, Legacy system, False positive

Data Providers: [All], [Manual], Cppcheck, Csv Coverage Import, Csv Tag Import, Squan Sources

ISO Characteristic: [All], Portability, Maintainability, Compatibility, Security, Performance efficiency, Reliability

Nature: [All], Non Conformity, Cloning, Metric, Guideline, Test, Relaxed Finding

Remediation Cost: [All], Huge, High, Medium, Low, Tiny, None

Severity: [All], Blocking, Critical, Major, Minor, Information, Unknown

Relaxed in Sources: ● All ○ Yes ○ No - Suspicious?: ● All ○ Yes ○ No

Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Cloned Files	1	+1	Squan Sources	Maintainability	Cloning	Medium	Major

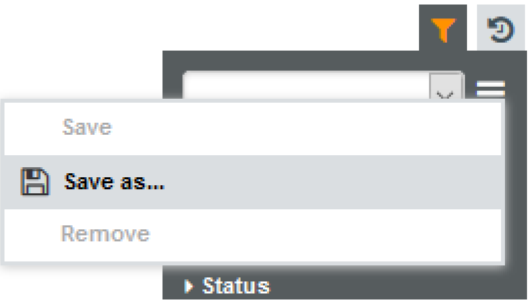
There shall be no duplicated files
Mnemonic: CC_FI
URL: /SQuORE_Server/documentation/reference_manual/index.html#sect_r_nocc
Characteristics: Testability, Modifiability, Changeability

Cloned Files (uid: 271) New 2 +2
[No description] Cloned Files: There shall be no duplicated files
apps, player.c New
apps, robot.c New

The new cloning violation for player.c

Since this is a new finding in a recently modified file, it makes sense to address the issue to avoid creeping technical debt in the project.

If you want to repeat this exploration at a later time, you can save the filter you just created.

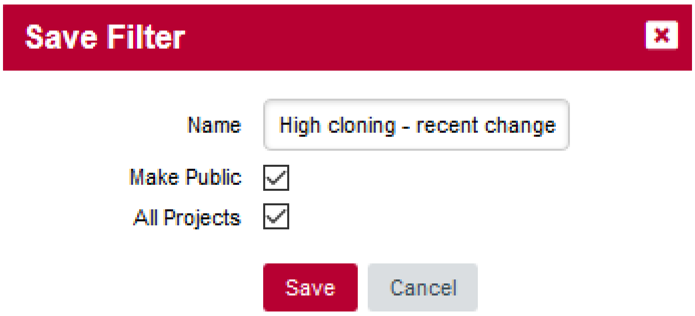


The Filter Panel menu

In order to save a filter, click the hamburger menu in the Filter Panel and select **Save as...**. The **Save Filter** popup appears and gives you the option to:

- Name your filter

- Make it public so other Squore users can apply it and adopt the same cloning exploration practice
- Make it available across all projects in Squore



Save Filter

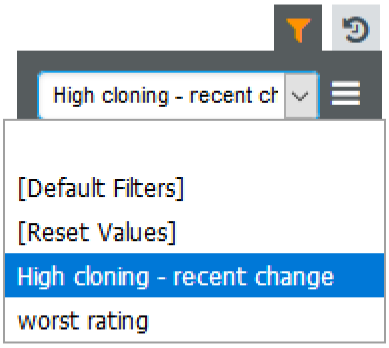
Name: High cloning - recent change

Make Public ☑

All Projects ☑

The Save Filter popup

After you save a filter it becomes available in the drop-down list of filters in the Filter Panel:



High cloning - recent ch

[Default Filters]

[Reset Values]

High cloning - recent change

worst rating

The filter saving procedure

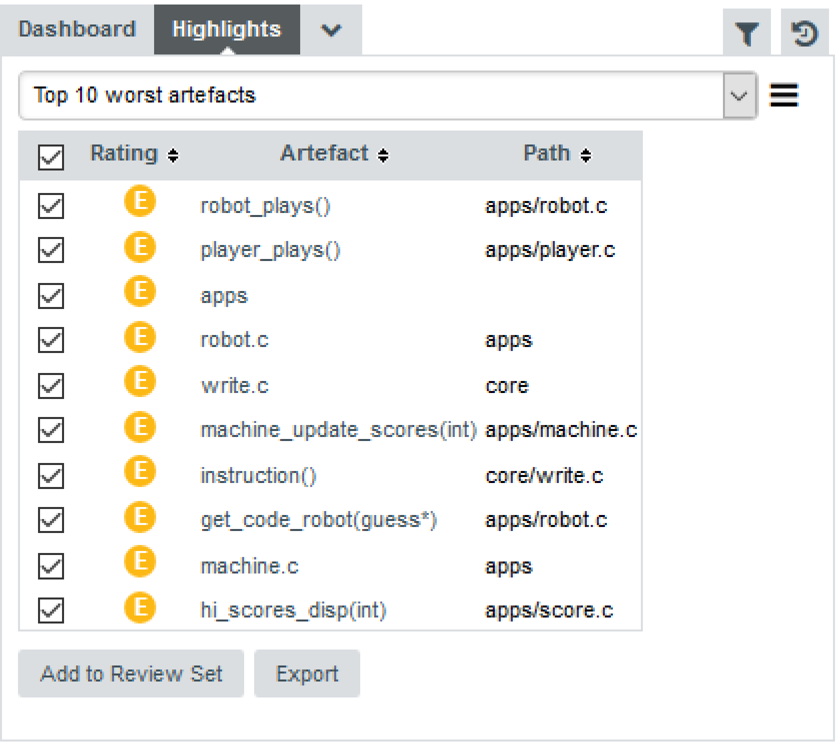
There are more filtering options for metrics and textual information you can explore directly in Squore's web interface and online help.

Finding Artefacts Using Highlights

In the previous section (Finding Artefacts Using Filters and Search), you got familiar with searching and filtering to find the artefact that has a negative impact on the overall rating of a project. in this section, you will learn to master the Highlights functionality, which aims to make the process of finding certain categories of artefacts easier and allows to display additional information about each artefact.

Highlights are flat lists of artefacts ordered in predefined categories for a model.

Let's try to confirm our findings about the worst and most deteriorated artefacts in Earth. Click on the version V6 of Earth and clear the filter. Click the **Highlights** tab of the Explorer and select the **Top 10 worst artefacts** highlight. The list appears as shown below:



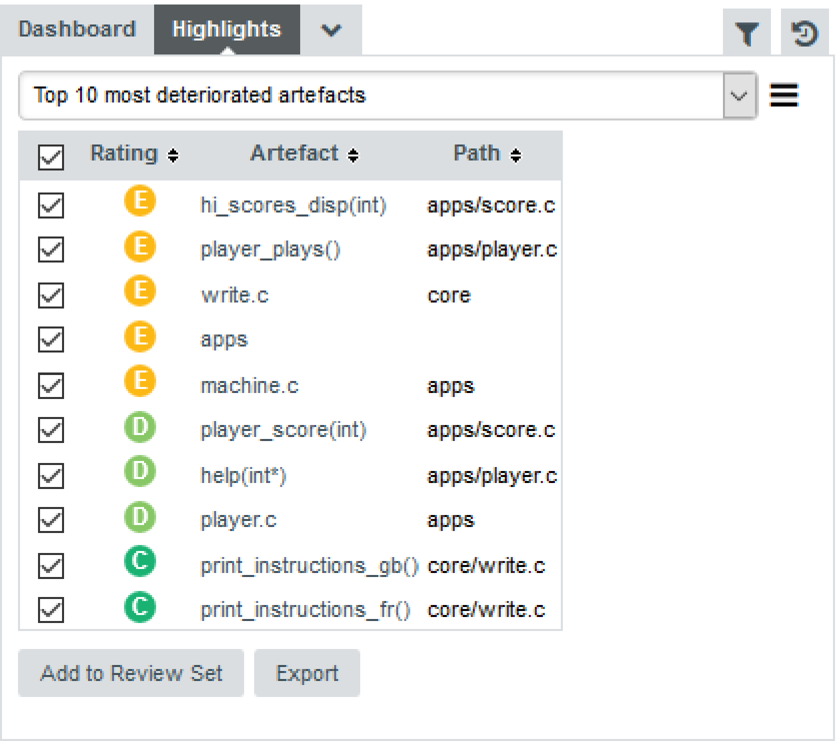
Top 10 worst artefacts

☑	Rating	Artefact	Path
☑	E	robot_plays()	apps/robot.c
☑	E	player_plays()	apps/player.c
☑	E	apps	
☑	E	robot.c	apps
☑	E	write.c	core
☑	E	machine_update_scores(int)	apps/machine.c
☑	E	instruction()	core/write.c
☑	E	get_code_robot(guess*)	apps/robot.c
☑	E	machine.c	apps
☑	E	hi_scores_disp(int)	apps/score.c

The Top 10 worst artefacts in the current version of Earth

The list confirms that some fo the worst-rated artefacts are the ones you explored before. The Highlights table shows you the artefact rating, name and path, and allows you to go to the artefact's dashboard directly by clicking the artefact name.

Now you can also find the deteriorated artefact hi_scores_disp(int) that you identified with a filter earlier in Finding Artefacts Using Filters and Search: select the **Top 10 most deteriorated artefacts** highlight to see the artefact appear in the list of deteriorated artefacts in this version.



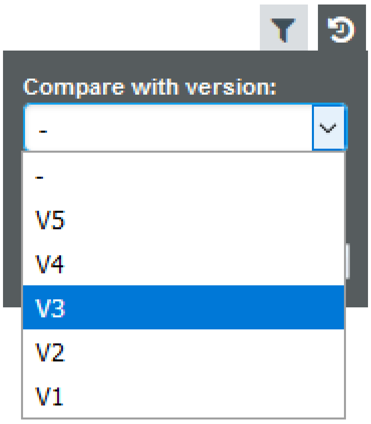
Top 10 most deteriorated artefacts

☑	Rating	Artefact	Path
☑	E	hi_scores_disp(int)	apps/score.c
☑	E	player_plays()	apps/player.c
☑	E	write.c	core
☑	E	apps	
☑	E	machine.c	apps
☑	D	player_score(int)	apps/score.c
☑	D	help(int*)	apps/player.c
☑	D	player.c	apps
☑	C	print_instructions_gb()	core/write.c
☑	C	print_instructions_fr()	core/write.c

The Top 10 most deteriorated artefacts in the current version of Earth

Artefacts are sorted by degradation, i.e. the difference between the value of the main indicator in the previous baseline version compared to the current value. By clicking the Export button, you can export the selected items to a CSV file. If the Export button is greyed out, your licence does not include the option to export data to CSV files.

By default, the list of most deteriorated artefacts is compiled based on the previous version of the project. You can compare results with another version of the project by setting your reference version in the Reference Panel, as shown below.



Compare with version:

- -
- V5
- V4
- V3
- V2
- V1

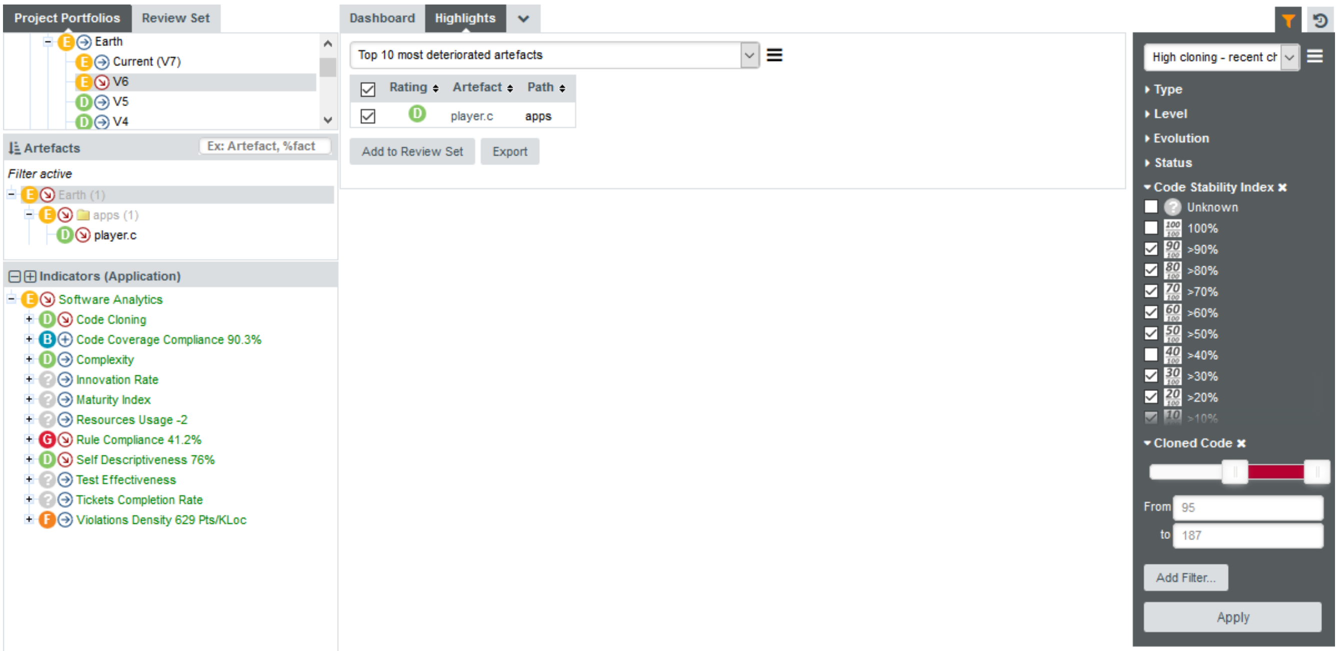
Choosing a project version in the Reference Panel

Changing settings in the Reference Panel impacts deltas and trends in most tabs on the dashboard and is covered in more details in Comparing versions and hiding old analyses with the Reference Panel.

Creating Highlights

From the Highlights tab, you can also create new highlights. These new highlights, like your filters can be saved and shared with other Squore users.

Let's extract more data from the artefacts singled out by your advanced filter in Advanced Filtering, where you found recently modified code with a high cloning ratio. To begin, simply select your saved filter from the Filter Panel and apply it. As you apply and modify the filter, the list of artefacts for the current highlight responds to the filter parameters:



Project Portfolios | Review Set

- E Earth
  - E Current (V7)
  - E V6
  - D V5
  - D V4

Artefacts — Ex: Artefact, %fact

Filter active

- E Earth (1)
  - E apps (1)
    - D player.c

Indicators (Application)

- E Software Analytics
  - D Code Cloning
  - B Code Coverage Compliance 90.3%
  - D Complexity
  - ? Innovation Rate
  - ? Maturity Index
  - ? Resources Usage -2
  - G Rule Compliance 41.2%
  - D Self Descriptiveness 76%
  - ? Test Effectiveness
  - ? Tickets Completion Rate
  - F Violations Density 629 Pts/KLoc

Dashboard | Highlights

Top 10 most deteriorated artefacts

☑	Rating	Artefact	Path
☑	D	player.c	apps

Add to Review Set | Export

High cloning - recent ch

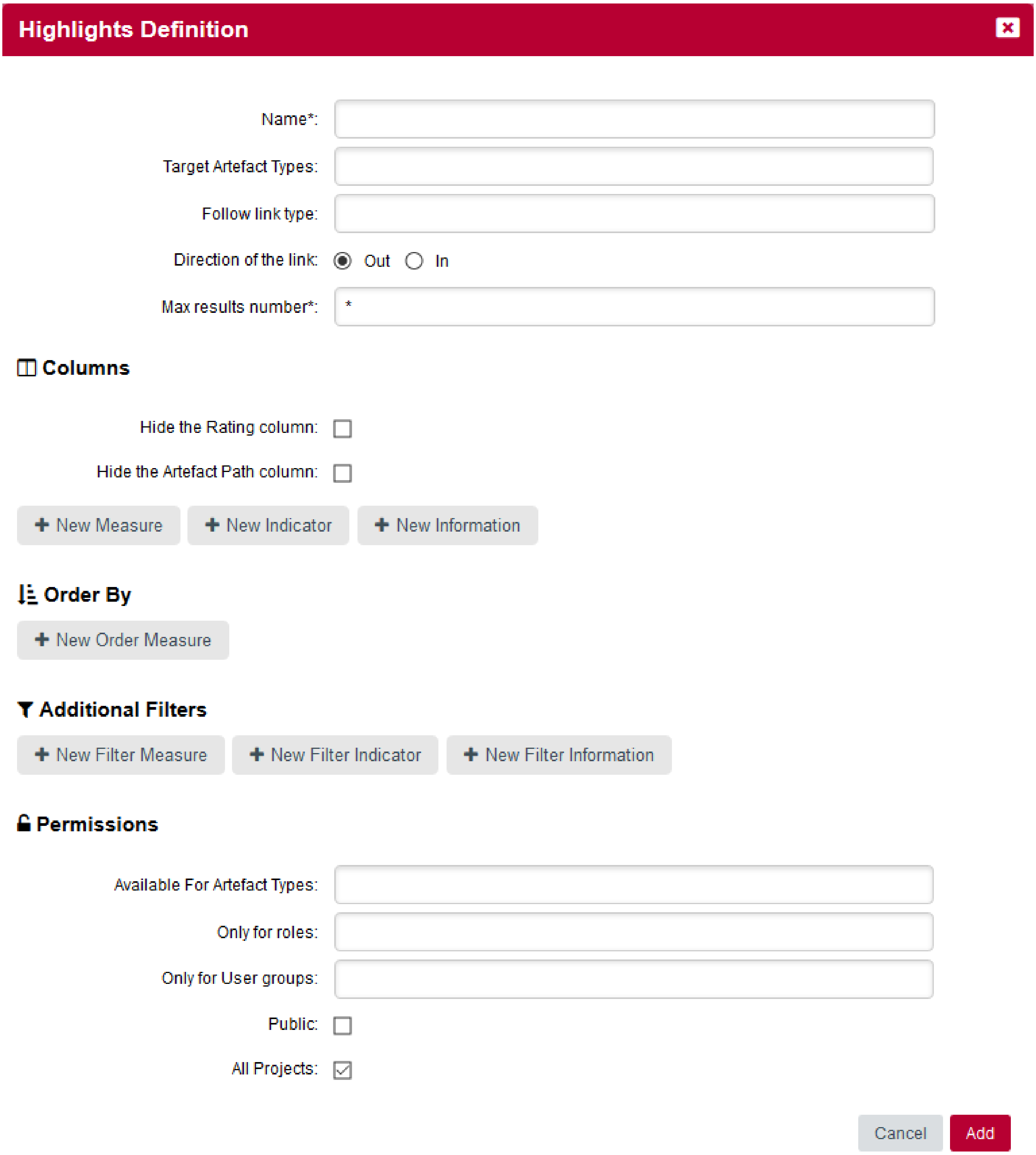
- Type
- Level
- Evolution
- Status
- Code Stability Index ✖
  - □ Unknown
  - □ 100%
  - ☑ >90%
  - ☑ >80%
  - ☑ >70%
  - ☑ >60%
  - ☑ >50%
  - □ >40%
  - ☑ >30%
  - ☑ >20%
  - ☑ >10%
- Cloned Code ✖
  - From 95
  - to 187

Add Filter... | Apply

The highlight filtered result

When a filter is applied, you can select a dynamic highlight called **Currently Fitered Artefacts**. This highlight displays a simple list of artefacts that match the filter that you can add to your review set or export to a CSV file.

Let's now build a highlight from scratch that uses the same filters and displays more information about the artefacts. Click the hamburger menu next to the list of highlights and select **Add Definition**. The popup that opens lets you configure the data columns to display for each artefact, as well as apply filters the same way you did in the Filter Panel:


Highlights Definition

Name*:

Target Artefact Types:

Follow link type:

Direction of the link: ● Out ○ In

Max results number*: *

Columns

Hide the Rating column: □

Hide the Artefact Path column: □

Order By

Additional Filters

Permissions

Available For Artefact Types:

Only for roles:

Only for User groups:

Public: □

All Projects: ☑

The Highlights Definition popup

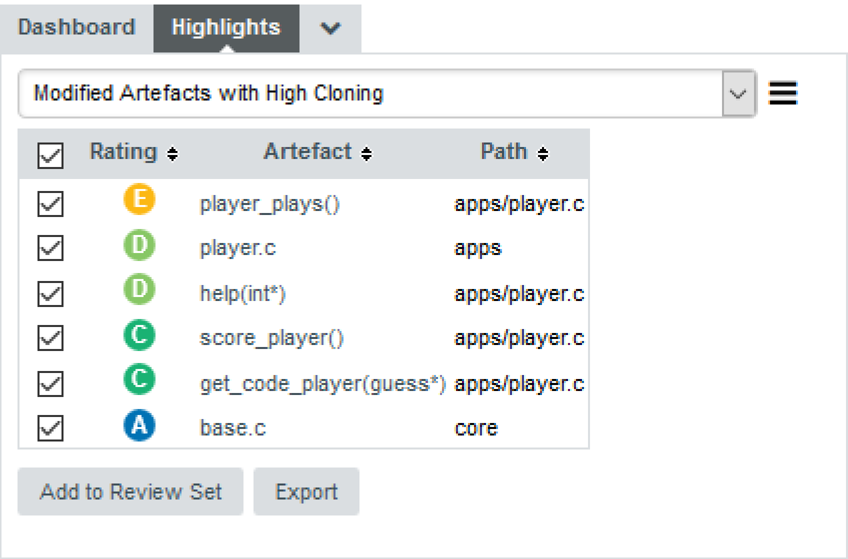
The values in the popup above reproduce conditions of the filter you used earlier:

- It applies to file and function artefacts (**Target Artefact Types**)
- It displays all results with no limit (**Max results number**)
- It defines two filter conditions
  - The measure **Code Stability Index** is lower than 100% (i.e.: the artefact was modified)

- The measure **Cloned Code** is higher than 0 (i.e.: there is some cloning)

Please note that you can also hide the path column & the rating column from your Highlights by selecting the checkboxes present in the "Columns" section.

Click **Add** to create the highlight and view the results. You can also clear the current filter since the highlight defines the same conditions.

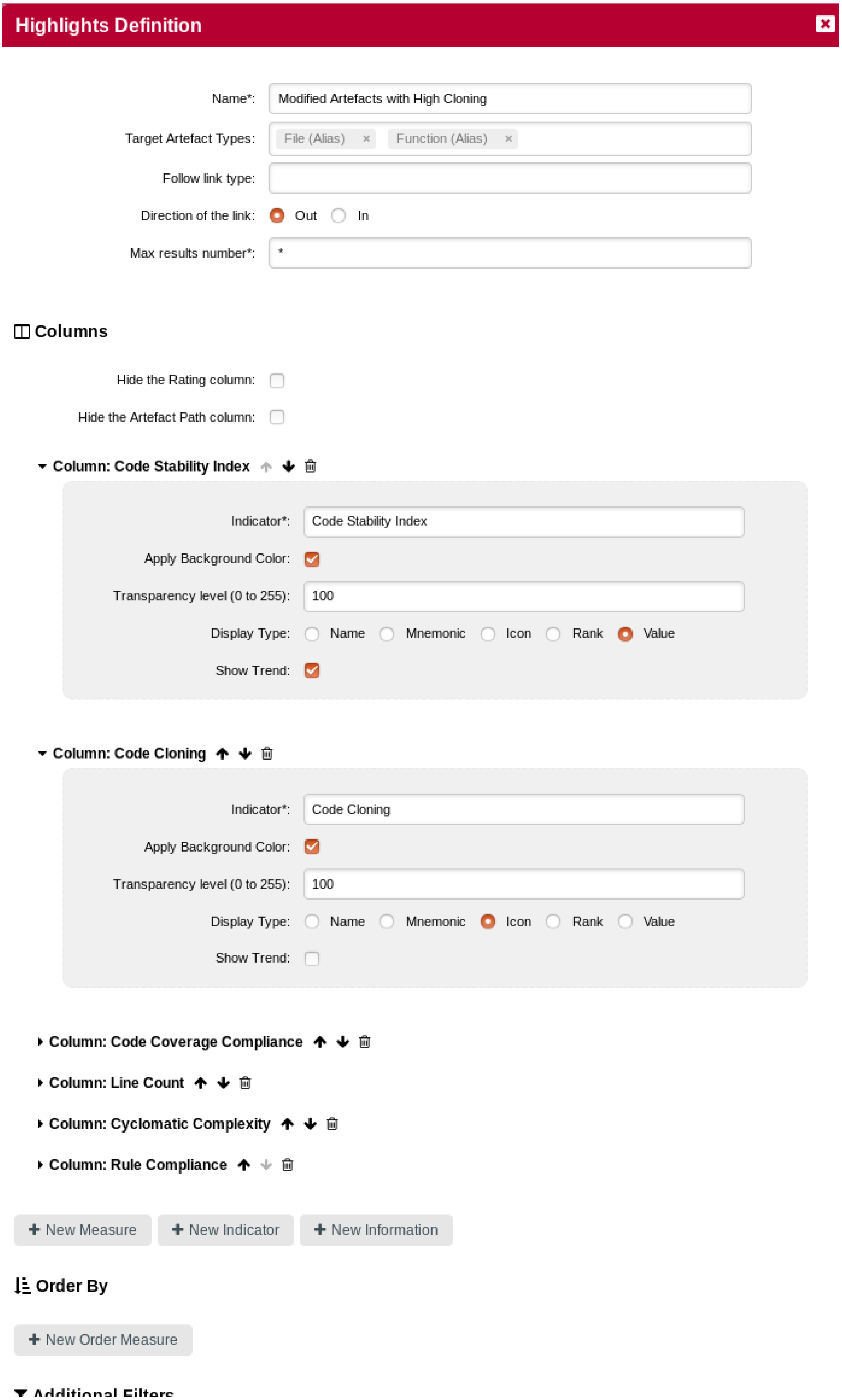


Modified Artefacts with High Cloning

☑	Rating	Artefact	Path
☑	E	player_plays()	apps/player.c
☑	D	player.c	apps
☑	D	help(int*)	apps/player.c
☑	C	score_player()	apps/player.c
☑	C	get_code_player(guess*)	apps/player.c
☑	A	base.c	core

Your basic highlight definition for modified artefacts with cloning

In order to add information about the artefacts displayed in the list, click the hamburger menu and select **Edit...**. The **Columns** section fo the popup lets you select additional data to display about each artefact. It makes sense to display the cloning ratio and stability details, as well as other indicators and metrics that will help you decide if modifying the code is risky or safe.



Highlights Definition

Name*: Modified Artefacts with High Cloning

Target Artefact Types: File (Alias) × Function (Alias) ×

Follow link type:

Direction of the link: ● Out ○ In

Max results number*: *

Columns

Hide the Rating column: □

Hide the Artefact Path column: □

Column: Code Stability Index

Indicator*: Code Stability Index

Apply Background Color: ☑

Transparency level (0 to 255): 100

Display Type: ○ Name ○ Mnemonic ○ Icon ○ Rank ● Value

Show Trend: ☑

Column: Code Cloning

Indicator*: Code Cloning

Apply Background Color: ☑

Transparency level (0 to 255): 100

Display Type: ○ Name ○ Mnemonic ● Icon ○ Rank ○ Value

Show Trend: □

Column: Code Coverage Compliance

Column: Line Count

Column: Cyclomatic Complexity

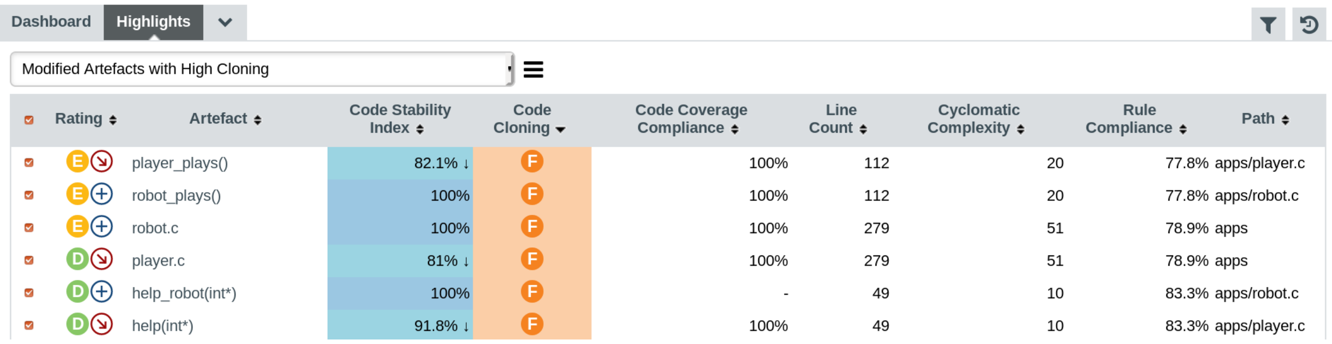
Column: Rule Compliance

Order By

Additional Filters

Adding columns to the new highlight

When you click **Update**, the highlight now shows all the requested details for the artefacts:



Dashboard | Highlights

Modified Artefacts with High Cloning

Rating	Artefact	Code Stability Index	Code Cloning	Code Coverage Compliance	Line Count	Cyclomatic Complexity	Rule Compliance	Path
E	player_plays()	82.1% ↓	F	100%	112	20	77.8%	apps/player.c
E	robot_plays()	100%	F	100%	112	20	77.8%	apps/robot.c
E	robot.c	100%	F	100%	279	51	78.9%	apps
D	player.c	81% ↓	F	100%	279	51	78.9%	apps
D	help_robot(int*)	100%	F	-	49	10	83.3%	apps/robot.c
D	help(int*)	91.8% ↓	F	100%	49	10	83.3%	apps/player.c

The extra columns for your new highlight

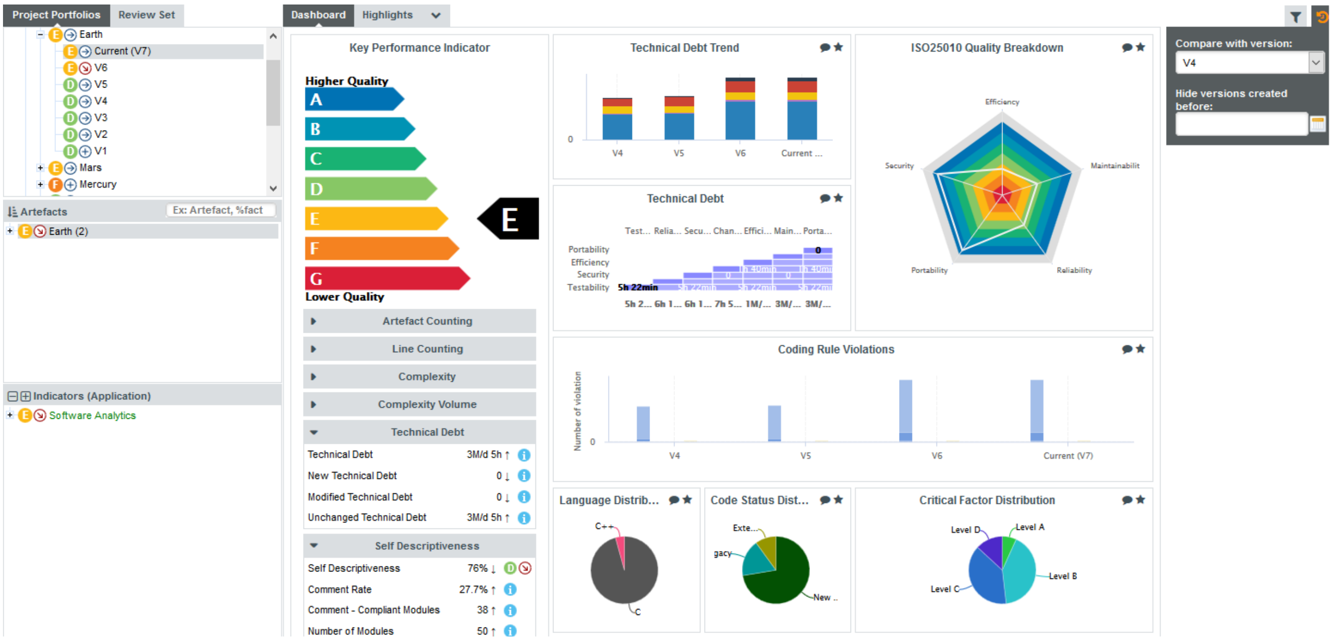
When you save a highlight definition, you can decide to make it available for all projects that use the same analysis model, and make it public so other Squore users can use it.

Comparing versions and hiding old analyses with the Reference Panel

A project with a long history can make it difficult to analyse results. Squore provides two ways to make dashboards clearer by:

- Picking a reference version to compare results with
- Hiding old versions completely from the Explorer

Both filtering options are available in the Reference Panel, which is always accessible from the Explorer by clicking the Reference icon (⟲).



Technical Debt	3M/d 5h ↑
New Technical Debt	0 ↓
Modified Technical Debt	0 ↓
Unchanged Technical Debt	3M/d 5h ↑
Self Descriptiveness	76% ↓
Comment Rate	27.7% ↑
Comment - Compliant Modules	38 ↑
Number of Modules	50 ↑

Compare with version: V4

Hide versions created before:

Comparing Earth's current version with V4 and hiding versions created before February 26th 2018

Changing the settings in the Reference Panel impacts the following items in the Explorer:

- Trends in the Dashboard and Highlights tabs are computed based on the version you specified instead of the previous version
- Charts that display delta values are updated to compute deltas against the reference version instead of the previous version

- Temporal charts are updated to hide versions whose version date is older than the reference date
- Deltas on the Findings tab are computed based on the specified reference version instead of the previous version
- Versions whose version date is older than the reference date are hidden from the Project Portfolios.

The Reference icon turns orange when the dashboard is impacted by your settings in the Reference Panel.

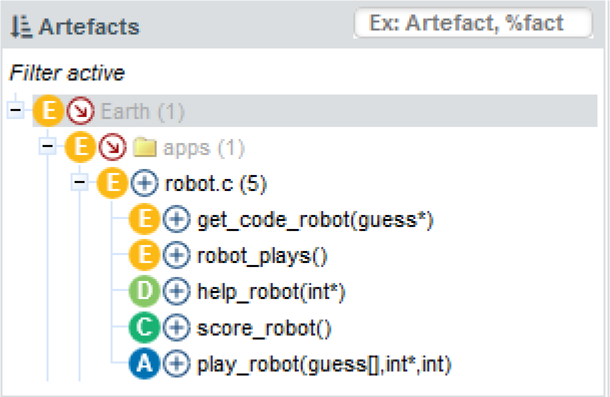
How Do I Find and Keep Track of Artefacts?

For some projects, you may want to collect artefacts so you can review them later. Squore enables you to build a Review Set, a list containing artefacts that you want to keep track of. Let's log in as the demo user to review all the new artefacts added to a project, in order to evaluate their level of quality.

Isolating the new artefacts can be done in three steps:

1. Log in using the demo user (*demo/demo*).
2. Click on version V6 of **Earth** in the Project Portfolios to display the dashboard for the last version of the project.
3. Create a filter to display only items in the Evolution column with the status **New** and apply your changes

You should see the following artefacts in the Artefact Tree:



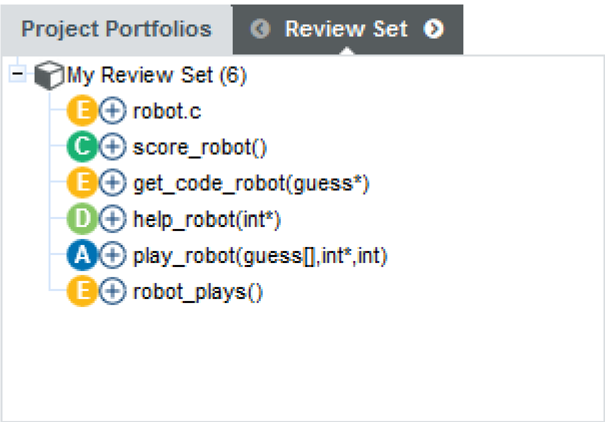
Filter active

- E Earth (1)
  - E apps (1)
    - E robot.c (5)
      - E get_code_robot(guess*)
      - E robot_plays()
      - D help_robot(int*)
      - C score_robot()
      - A play_robot(guess[],int*,int)

The new artefacts in the current version of Earth

Squore makes it easy for you to keep track of these artefacts. Click on the icon above the Artefact Tree and select **Add Filtered Results to Review Set**.

You can now clear your filter, the artefacts you want to review are stored in your Review Set. Click the **Review Set** tab in the left pane of the explorer to find the items you just saved.



My Review Set (6)
- E robot.c
- C score_robot()
- E get_code_robot(guess*)
- D help_robot(int*)
- A play_robot(guess[],int*,int)
- E robot_plays()

The Review Set filled with new artefacts for version V6 of Earth

At any moment, the artefact currently selected in the Artefact Tree can be sent to the Review Set as well. Simply display the context menu for an artefact and click **Add to Review Set** to add it to the Review Set. Clicking an item in the Review Set pane has the same effect as clicking it in the Artefact Tree: the dashboard refreshes to show the information for that artefact. You can use the left and right arrows in the Review Set pane to go to the previous and next artefact in the list.

If you want to know more about what actions you can take after reviewing artefacts, refer to Managing Your To-Do List With Squore and Communicating With Squore.

How can I Understand and Enhance My Model?

Squore provides tools to understand, verify and enhance your model under the Models menu.

- The **Viewer**, a graphical representation of all the analysis models on Squore Server
- The **Validator**, a debug tool for model writers
- The **Dashboard Editor**, which allows customising the dashboards that all users will see

* The **Analysis Model Editor**, which allows modifying the model's ruleset

Users whose global role grants the "View Models" permission have access to the first two tools.

Users whose global role grants the "Modify Models" permission have access to the last two tools.

Viewer

To use the Viewer:

1. Click **Models > Viewer** in the toolbar.
2. Select the analysis model you want to browse.
3. Select the artefact level you want to browse.
4. Choose your preferred graphical representation between Space-Tree

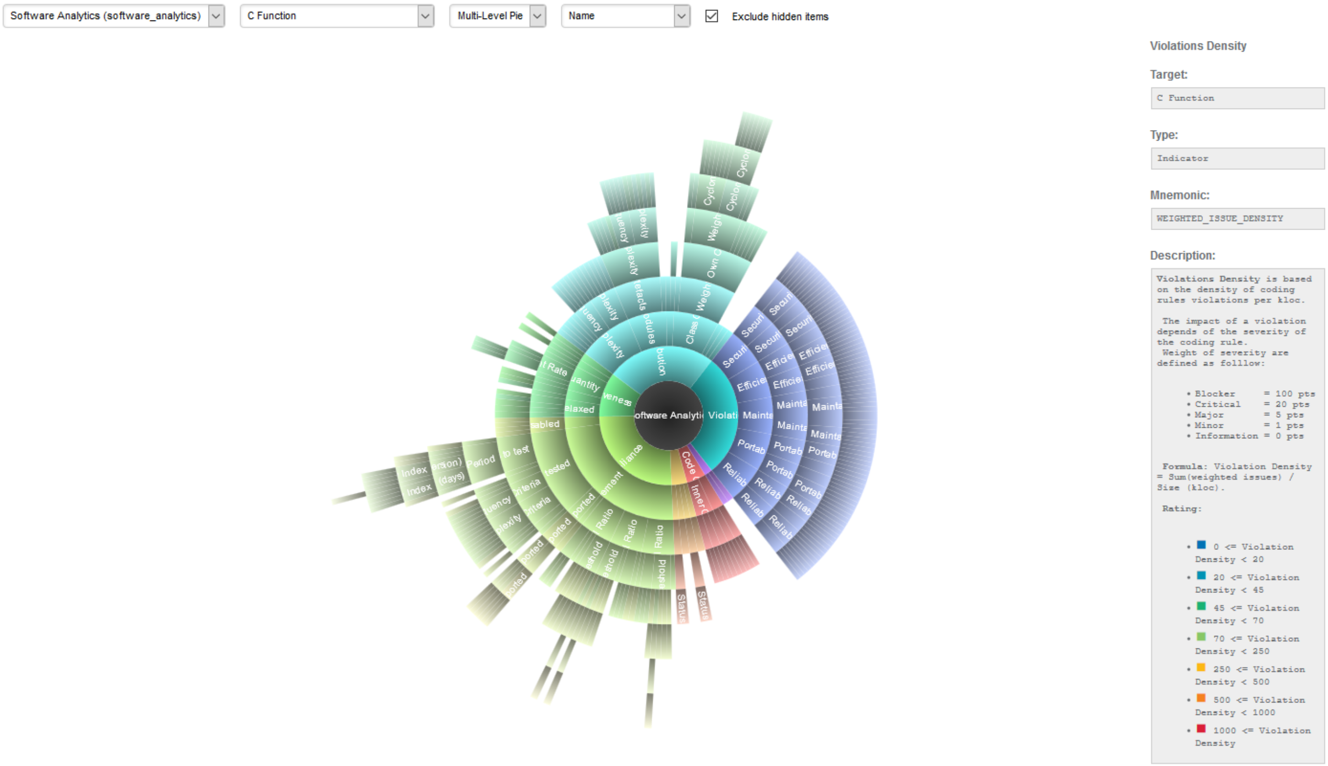
   and Multi-level pie.

5. Select whether measures are displayed using their full name or their mnemonic

   .

Upon selecting the parameters above, the page is refreshed with the top-level indicators in the model, and you can click each indicator to unveil sub-indicators and their characteristics. You can drag the tree left and right to reveal all sub-levels if necessary. For each indicator selected, Squore displays the following information:

- **Target** is the target artefact type for the selected item
- **Type** is the type of the selected item
- **Mnemonic** is the shortcode for the selected item
- **description** is the description of the selected item
- **Data Provider** is the Data Provider responsible for computing the selected item
- **MeasureId** is the measure ID of the selected item
- **Computation** is the formula used to compute the value of the selected item
- **ScaleId** is ID of the scale associated with the selected item
- **Levels** is the list of levels available for the selected item and their ranges



Software Analytics (software_analytics) | C Function | Multi-Level Pie | Name | ☑ Exclude hidden items

Violations Density

Target:

C Function

Type:

Indicator

Mnemonic:

WEIGHTED_ISSUE_DENSITY

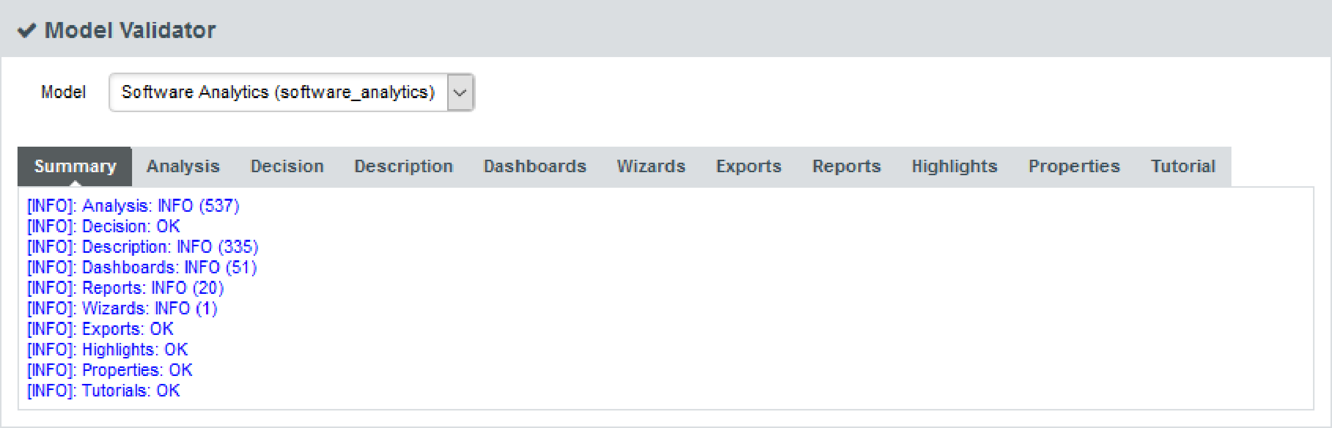
Description:

```
Violations Density is based
on the density of coding
rules violations per kloc.

 The impact of a violation
depends of the severity of
the coding rule.
 Weight of severity are
defined as folllow:

    • Blocker     = 100 pts
    • Critical    = 20 pts
    • Major       = 5 pts
    • Minor       = 1 pts
    • Information = 0 pts

 Formula: Violation Density
= Sum(weighted issues) /
Size (kloc).

 Rating:

    • 🟦 0 <= Violation
      Density < 20
    • 🟦 20 <= Violation
      Density < 45
    • 🟩 45 <= Violation
      Density < 70
    • 🟩 70 <= Violation
      Density < 250
    • 🟨 250 <= Violation
      Density < 500
    • 🟧 500 <= Violation
      Density < 1000
    • 🟥 1000 <= Violation
      Density
```

The software_analytics model presented as a multi-level pie in the Viewer

Validator

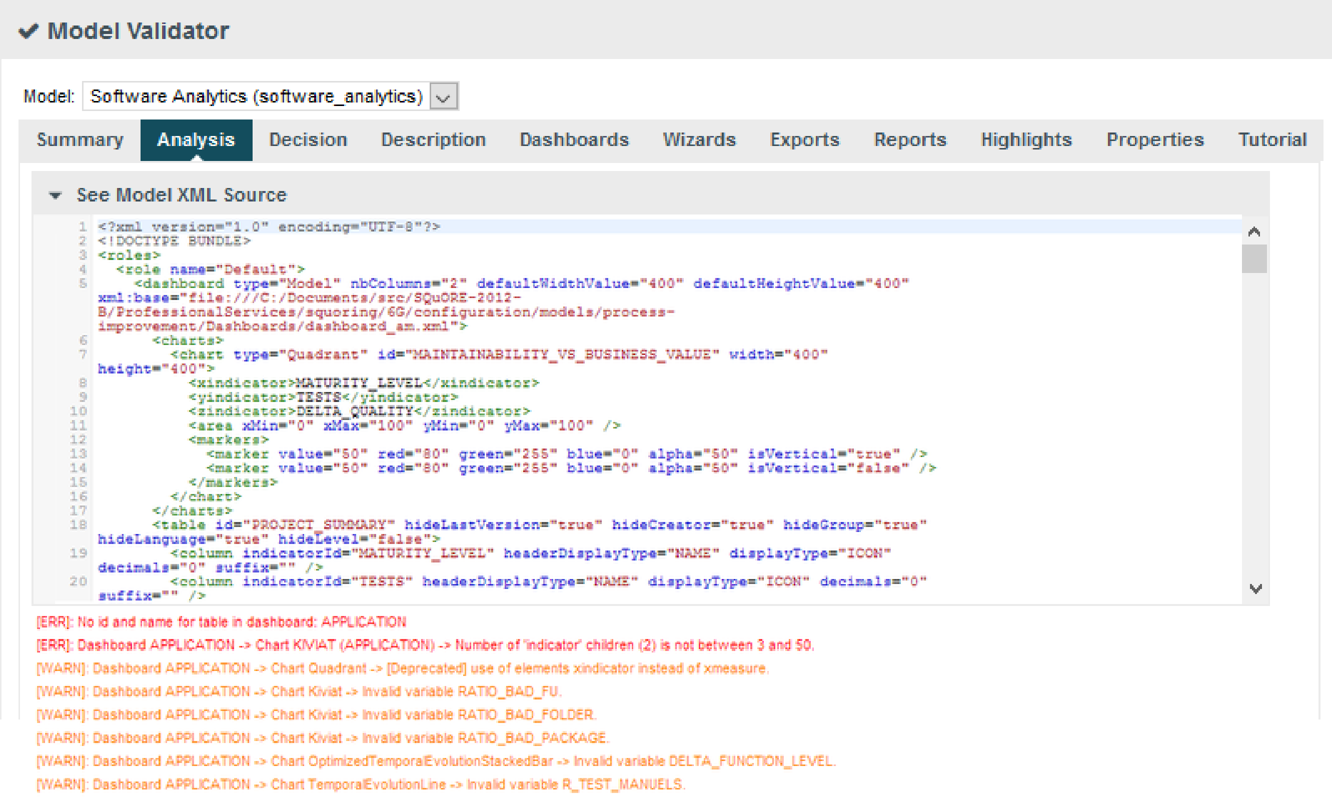
If your work involves adjusting the model's metrics or dashboard, you can use the Validator to verify its integrity during as you make changes. Click **Models > Validator** to display the diagnostics organised by category, as shown below:



```
[INFO]: Analysis: INFO (537)
[INFO]: Decision: OK
[INFO]: Description: INFO (335)
[INFO]: Dashboards: INFO (51)
[INFO]: Reports: INFO (20)
[INFO]: Wizards: INFO (1)
[INFO]: Exports: OK
[INFO]: Highlights: OK
[INFO]: Properties: OK
[INFO]: Tutorials: OK
```

The software_analytics model in the Validator

The Summary tab displays a summary of all the diagnostics run for each category. By clicking any of the other tabs, more details are shown about potential problems found in your model. You can also show the complete XML of the model to understand the errors reported. The XML can be searched by using the Ctrl+F key combination to bring up the search dialogue, and then Ctrl+G to search for the next occurrence of the search term:



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE BUNDLE>
<roles>
  <role name="Default">
    <dashboard type="Model" nbColumns="2" defaultWidthValue="400" defaultHeightValue="400"
xml:base="file:///C:/Documents/src/SQuORE-2012-
B/ProfessionalServices/squoring/6G/configuration/models/process-
improvement/Dashboards/dashboard_am.xml">
      <charts>
        <chart type="Quadrant" id="MAINTAINABILITY_VS_BUSINESS_VALUE" width="400"
height="400">
          <xindicator>MATURITY_LEVEL</xindicator>
          <yindicator>TESTS</yindicator>
          <zindicator>DELTA_QUALITY</zindicator>
          <area xMin="0" xMax="100" yMin="0" yMax="100" />
          <markers>
            <marker value="50" red="80" green="255" blue="0" alpha="50" isVertical="true" />
            <marker value="50" red="80" green="255" blue="0" alpha="50" isVertical="false" />
          </markers>
        </chart>
      </charts>
      <table id="PROJECT_SUMMARY" hideLastVersion="true" hideCreator="true" hideGroup="true"
hideLanguage="true" hideLevel="false">
        <column indicatorId="MATURITY_LEVEL" headerDisplayType="NAME" displayType="ICON"
decimals="0" suffix="" />
        <column indicatorId="TESTS" headerDisplayType="NAME" displayType="ICON" decimals="0"
suffix="" />
```

```
[ERR]: No id and name for table in dashboard: APPLICATION
[ERR]: Dashboard APPLICATION -> Chart KIVIAT (APPLICATION) -> Number of 'indicator' children (2) is not between 3 and 50.
[WARN]: Dashboard APPLICATION -> Chart Quadrant -> [Deprecated] use of elements xindicator instead of xmeasure.
[WARN]: Dashboard APPLICATION -> Chart Kiviat -> Invalid variable RATIO_BAD_FU.
[WARN]: Dashboard APPLICATION -> Chart Kiviat -> Invalid variable RATIO_BAD_FOLDER.
[WARN]: Dashboard APPLICATION -> Chart Kiviat -> Invalid variable RATIO_BAD_PACKAGE.
[WARN]: Dashboard APPLICATION -> Chart OptimizedTemporalEvolutionStackedBar -> Invalid variable DELTA_FUNCTION_LEVEL.
[WARN]: Dashboard APPLICATION -> Chart TemporalEvolutionLine -> Invalid variable R_TEST_MANUELS.
```

The Validator reporting errors

Your Squore administrator can help you get more information model development. You can also refer to the Squore Configuration Guide for a complete reference.

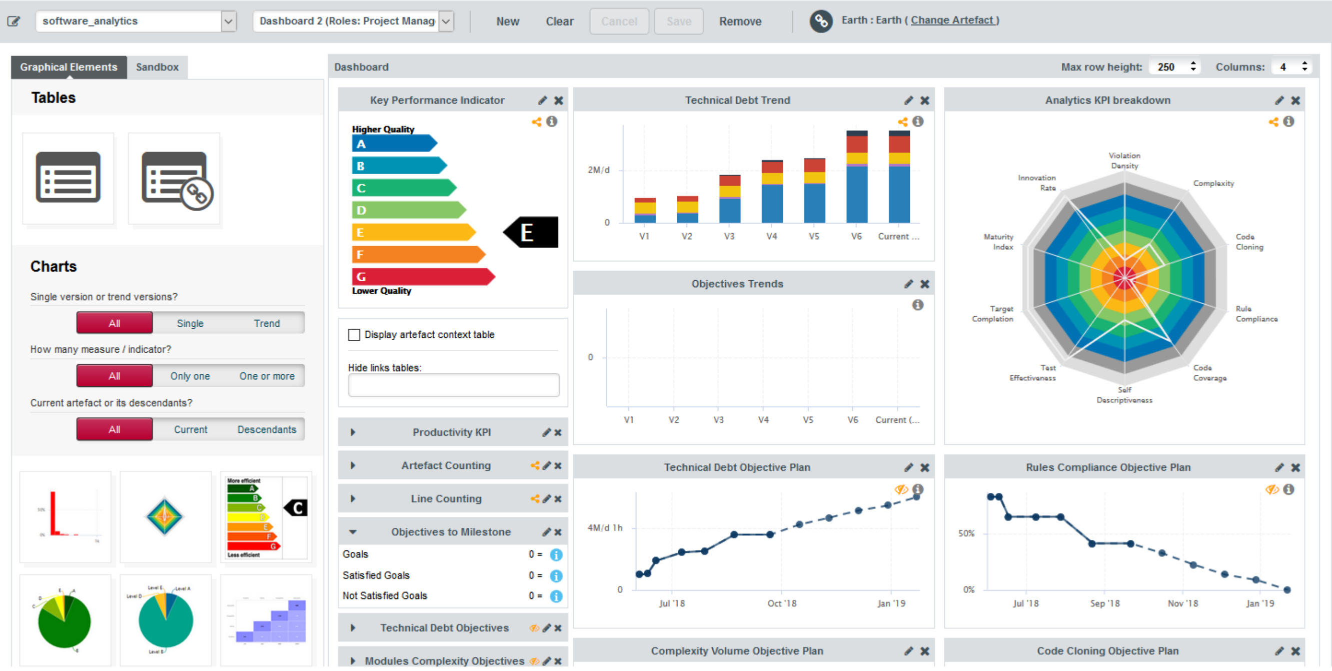
Dashboard Editor

The Dashboard Editor is a graphical editor for the dashboards of a particular model. Dashboards consist of a key performance indicator, a list of tables and one or more columns of predefined charts. With the Dashboard Editor, you can rearrange the information shown on the dashboard for all users, or create a completely new dashboard for a new project role or new group in your project.

In order to use the Dashboard Editor:

1. Click **Models > Dashboard Editor**
2. Select a model and load an existing dashboard

The current dashboard skeleton is loaded in the editor, as shown below:

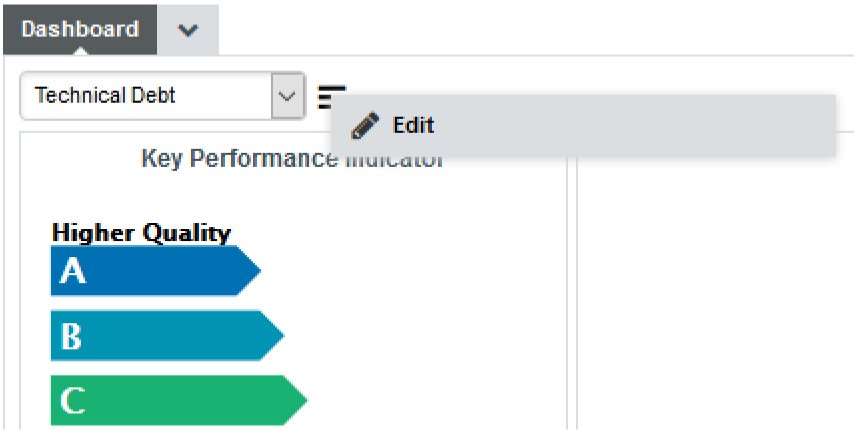

software_analytics
Dashboard 2 (Roles: Project Manag
New
Clear
Cancel
Save
Remove
Earth : Earth (Change Artefact)
Graphical Elements
Sandbox
Tables
Charts
Single version or trend versions?
All
Single
Trend
How many measure / indicator?
All
Only one
One or more
Current artefact or its descendants?
All
Current
Descendants
Dashboard
Max row height: 250
Columns: 4
Key Performance Indicator
Higher Quality
Lower Quality
Technical Debt Trend
Analytics KPI breakdown
Objectives Trends
Display artefact context table
Hide links tables:
Productivity KPI
Artefact Counting
Line Counting
Objectives to Milestone
Goals
Satisfied Goals
Not Satisfied Goals
Technical Debt Objectives
Modules Complexity Objectives
Technical Debt Objective Plan
Rules Compliance Objective Plan
Complexity Volume Objective Plan
Code Cloning Objective Plan

The Software Analytics model in the Dashboard Editor

The right panel displays your current dashboard, where every chart can be resized or edited, while the left panel allows you to add new charts or tables by dragging them to your current dashboard. When you are satisfied with your changes, you can save your modifications. You can also create a new dashboard, using an existing one as a basis for the new one, or start from a blank canvas.

Your changes are saved in Squore's workspace folder on the server and are made available to all other Squore users when you click **Save**.

You can edit a dashboard directly from the Explorer by clicking the Edit Dashboard... entry in the Explorer Settings.



The Edit Dashboard... menu loads the current dashboard in the editor so you can change it

More detailed explanations about the Dashboard Editor can be found in Squore's Online Help.

Analysis Model Editor

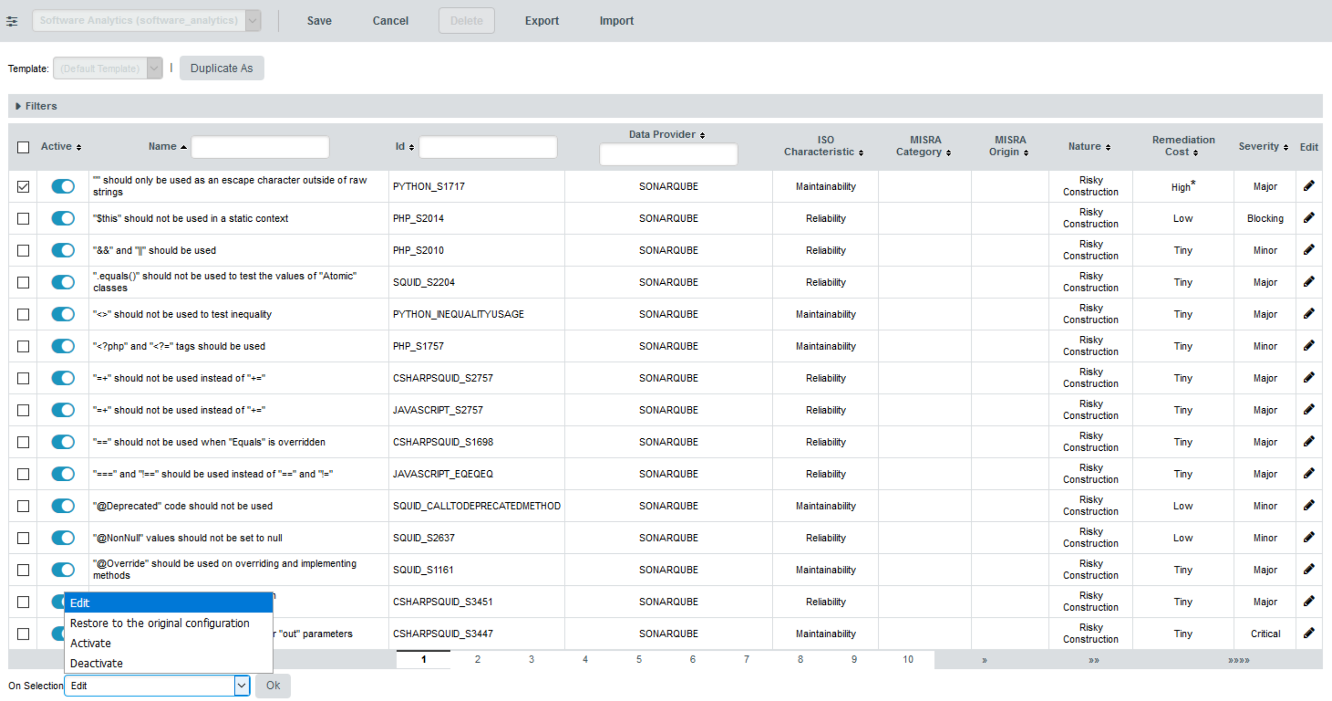
The Analysis Model Editor is a graphical ruleset editor where you can turn rules on and off, or adjust the categories associated with each rule in your model.

It also allows you to save ruleset templates so that you can use a different set of rules for each project you create

In order to use the Analysis Model Editor:

1. Click **Models > Analysis Model Editor**
2. Select a model to load its ruleset

The entire ruleset for the current model is displayed in table form, as shown below:


Software Analytics (software_analytics)
Save
Cancel
Delete
Export
Import
Duplicate As
Filters
Edit
Restore to the original configuration
Activate
Deactivate
Ok

Template: (Default Template)

☐	Active	Name	Id	Data Provider	ISO Characteristic	MISRA Category	MISRA Origin	Nature	Remediation Cost	Severity	Edit
☑	●	"" should only be used as an escape character outside of raw strings	PYTHON_S1717	SONARQUBE	Maintainability			Risky Construction	High*	Major	✎
☐	●	"$this" should not be used in a static context	PHP_S2014	SONARQUBE	Reliability			Risky Construction	Low	Blocking	✎
☐	●	"&&" and "||" should be used	PHP_S2010	SONARQUBE	Reliability			Risky Construction	Tiny	Minor	✎
☐	●	".equals()" should not be used to test the values of "Atomic" classes	SQUID_S2204	SONARQUBE	Reliability			Risky Construction	Tiny	Major	✎
☐	●	"<>" should not be used to test inequality	PYTHON_INEQUALITYUSAGE	SONARQUBE	Maintainability			Risky Construction	Tiny	Major	✎
☐	●	"<?php" and "<?=" tags should be used	PHP_S1757	SONARQUBE	Maintainability			Risky Construction	Tiny	Minor	✎
☐	●	"=+" should not be used instead of "+="	CSHARPSQUID_S2757	SONARQUBE	Reliability			Risky Construction	Tiny	Major	✎
☐	●	"=+" should not be used instead of "+="	JAVASCRIPT_S2757	SONARQUBE	Reliability			Risky Construction	Tiny	Major	✎
☐	●	"==" should not be used when "Equals" is overridden	CSHARPSQUID_S1698	SONARQUBE	Reliability			Risky Construction	Tiny	Major	✎
☐	●	"===" and "!==" should be used instead of "==" and "!="	JAVASCRIPT_EQEQEQ	SONARQUBE	Reliability			Risky Construction	Tiny	Major	✎
☐	●	"@Deprecated" code should not be used	SQUID_CALLTODEPRECATEDMETHOD	SONARQUBE	Maintainability			Risky Construction	Low	Minor	✎
☐	●	"@NonNull" values should not be set to null	SQUID_S2637	SONARQUBE	Reliability			Risky Construction	Low	Minor	✎
☐	●	"@Override" should be used on overriding and implementing methods	SQUID_S1161	SONARQUBE	Maintainability			Risky Construction	Tiny	Major	✎
☐	●	n	CSHARPSQUID_S3451	SONARQUBE	Reliability			Risky Construction	Tiny	Major	✎
☐	●	r "out" parameters	CSHARPSQUID_S3447	SONARQUBE	Maintainability			Risky Construction	Tiny	Critical	✎

1 2 3 4 5 6 7 8 9 10 » »» »»»»

On Selection: Edit

The Analysis Model Editor displaying the ruleset of the Software Analytics model

Use the filter panel and the table headers to find the rule you want to modify. You can activate or deactivate a rule by clicking the on/off switch in the table. If you want to make more modifications, click the **Edit** icon for this rule.

You can edit multiple rules at once by checking several rules and using the actions list at the bottom of the page. When you save your changes, the configuration is reloaded and every new analysis for this model will use the new settings.

Changes made in the web interface are saved in a workspace folder on the server.

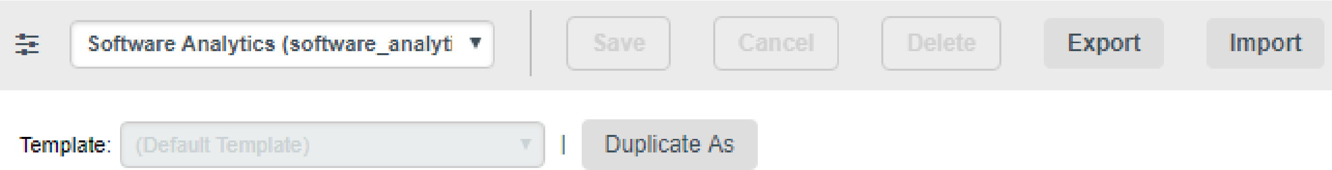
Using Ruleset Templates

Using the Analysis Model Editor, you can set up various ruleset templates to modify or ignore rules that do not apply for certain departments or project teams within your organisation.

Users with model edition privileges (see the **Modify Models** permission in User Global Roles) can define templates right from the Analysis Model Editor. Project managers can decide to modify existing templates or create new ones from the project wizard. In order to ensure that projects are analysed using company-wide standards, templates can be marked as approved, which prevents them from being modified by other users.

In order to create a ruleset template:

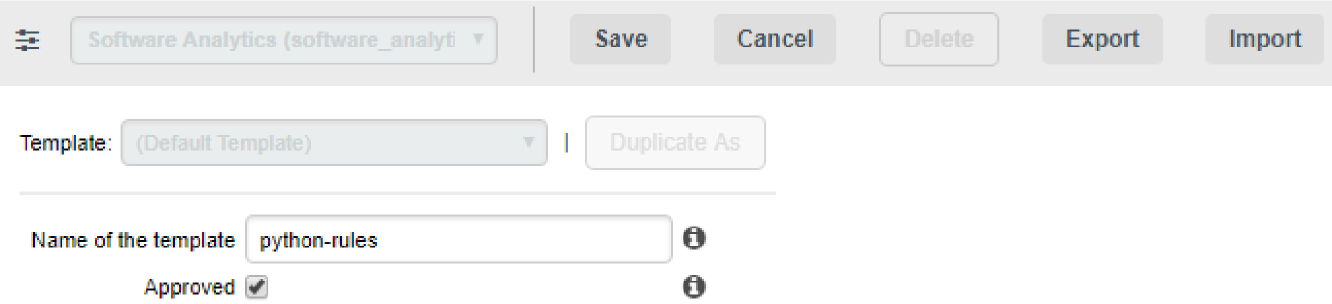
1. Click on **Models > Analysis Model Editor**
2. Select an analysis model and locate the **Template** selection list above the filter tools. For a model where no templates exist yet, only the Duplicate As button is available so can create a new template from the default one.



Template: (Default Template)

The ruleset template tools for a model with no custom templates yet

1. Click Duplicate As to create a new template and enter edit mode. In this example, we are creating a ruleset that contains only rules that apply to the Python programming language. By checking the **Approved** box, we are defining this ruleset template as read-only for project managers.



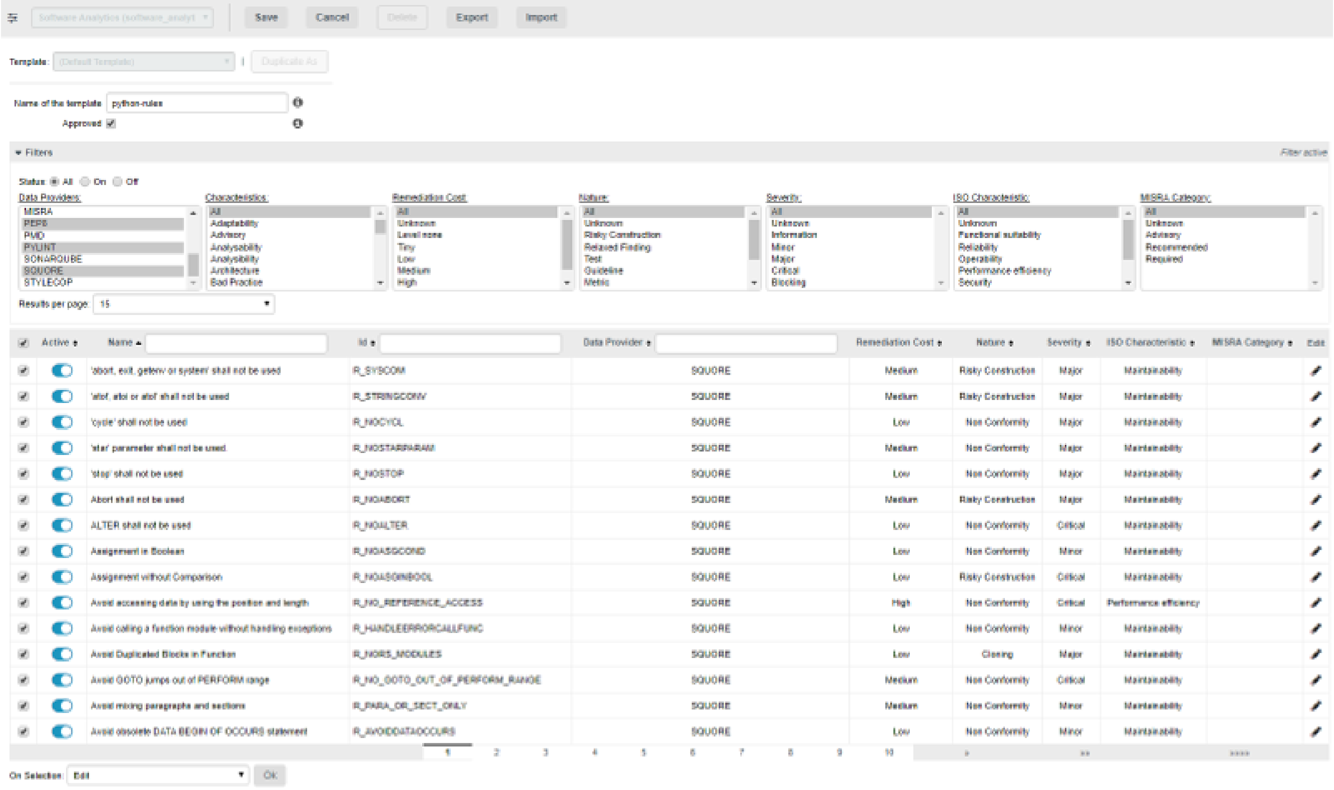
Template: (Default Template)

Name of the template: python-rules

Approved ☑

Creating a new python-rules ruleset template

1. Activate, deactivate or modify any rule you want for the template. In this example, we use the filter tools to select all Data Providers, turn off all the rules, and then select the Python-related Data Providers to activate all their rules



Template: (Default Template)

Name of the template: python-rules

Approved ☑

Status: ● All ○ On ○ Off

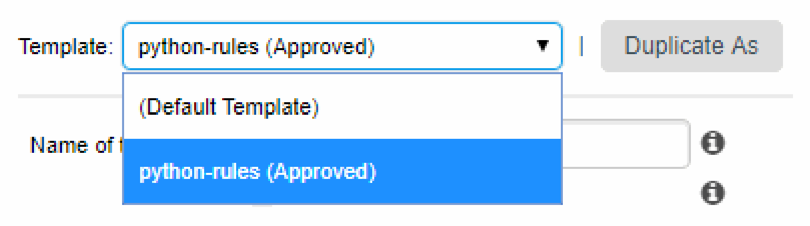
Results per page: 15

☑	Active	Name	Id	Data Provider	Remediation Cost	Nature	Severity	ISO Characteristic	MISRA Category	Edit
☑	●	'abort, exit, getenv or system' shall not be used	R_SYSCOM	SQUORE	Medium	Risky Construction	Major	Maintainability		✎
☑	●	'atof, atoi or atol' shall not be used	R_STRINGCONV	SQUORE	Medium	Risky Construction	Major	Maintainability		✎
☑	●	'cycle' shall not be used	R_NOCYCL	SQUORE	Low	Non Conformity	Major	Maintainability		✎
☑	●	'star' parameter shall not be used.	R_NOSTARPARAM	SQUORE	Medium	Non Conformity	Major	Maintainability		✎
☑	●	'stop' shall not be used	R_NOSTOP	SQUORE	Low	Non Conformity	Major	Maintainability		✎
☑	●	Abort shall not be used	R_NOABORT	SQUORE	Medium	Risky Construction	Major	Maintainability		✎
☑	●	ALTER shall not be used	R_NOALTER	SQUORE	Low	Non Conformity	Critical	Maintainability		✎
☑	●	Assignement in Boolean	R_NOASGCOND	SQUORE	Low	Non Conformity	Minor	Maintainability		✎
☑	●	Assignement without Comparison	R_NOASGINBOOL	SQUORE	Low	Risky Construction	Critical	Maintainability		✎
☑	●	Avoid accessing data by using the position and length	R_NO_REFERENCE_ACCESS	SQUORE	High	Non Conformity	Critical	Performance efficiency		✎
☑	●	Avoid calling a function module without handling exceptions	R_HANDLEERRORCALLFUNC	SQUORE	Low	Non Conformity	Minor	Maintainability		✎
☑	●	Avoid Duplicated Blocks in Function	R_NORS_MODULES	SQUORE	Low	Cloning	Major	Maintainability		✎
☑	●	Avoid GOTO jumps out of PERFORM range	R_NO_GOTO_OUT_OF_PERFORM_RANGE	SQUORE	Medium	Non Conformity	Critical	Maintainability		✎
☑	●	Avoid mixing paragraphs and sections	R_PARA_OR_SECT_ONLY	SQUORE	Medium	Non Conformity	Minor	Maintainability		✎
☑	●	Avoid obsolete DATA BEGIN OF OCCURS statement	R_AVOIDDATAOCCURS	SQUORE	Low	Non Conformity	Minor	Maintainability		✎

On Selection: Edit

The python ruleset template includes the Pylint, pep8 and Squan Sources rulesets

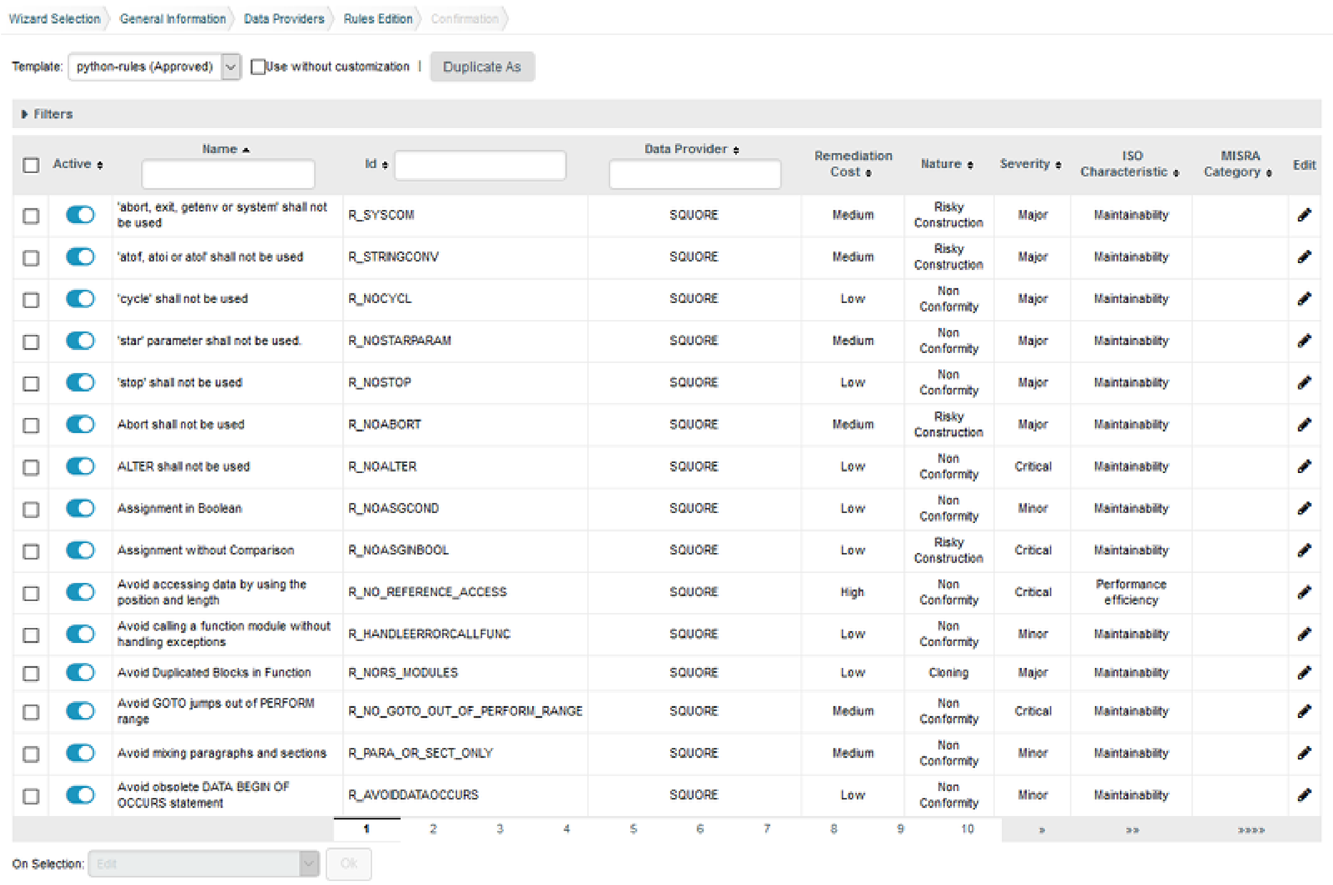
1. When you are satisfied with your rule selection, click **Save** to save the template. It now appears in the template selection list. You can still modify it as needed, or click Duplicate As to start creating a new template based on your first template.



Template: python-rules (Approved)

The saved python ruleset

Project managers can start using your template immediately by selecting it in the Ruleset Edition page of the project wizard, which is displayed after the Data Provider selection screen:



Template: python-rules (Approved) □ Use without customization

□	Active	Name	Id	Data Provider	Remediation Cost	Nature	Severity	ISO Characteristic	MISRA Category	Edit
□		'abort, exit, getenv or system' shall not be used	R_SYSCOM	SQUORE	Medium	Risky Construction	Major	Maintainability		✎
□		'atof, atoi or atol' shall not be used	R_STRINGCONV	SQUORE	Medium	Risky Construction	Major	Maintainability		✎
□		'cycle' shall not be used	R_NOCYCL	SQUORE	Low	Non Conformity	Major	Maintainability		✎
□		'star' parameter shall not be used.	R_NOSTARPARAM	SQUORE	Medium	Non Conformity	Major	Maintainability		✎
□		'stop' shall not be used	R_NOSTOP	SQUORE	Low	Non Conformity	Major	Maintainability		✎
□		Abort shall not be used	R_NOABORT	SQUORE	Medium	Risky Construction	Major	Maintainability		✎
□		ALTER shall not be used	R_NOALTER	SQUORE	Low	Non Conformity	Critical	Maintainability		✎
□		Assignment in Boolean	R_NOASGCOND	SQUORE	Low	Non Conformity	Minor	Maintainability		✎
□		Assignment without Comparison	R_NOASGINBOOL	SQUORE	Low	Risky Construction	Critical	Maintainability		✎
□		Avoid accessing data by using the position and length	R_NO_REFERENCE_ACCESS	SQUORE	High	Non Conformity	Critical	Performance efficiency		✎
□		Avoid calling a function module without handling exceptions	R_HANDLEERRORCALLFUNC	SQUORE	Low	Non Conformity	Minor	Maintainability		✎
□		Avoid Duplicated Blocks in Function	R_NORS_MODULES	SQUORE	Low	Cloning	Major	Maintainability		✎
□		Avoid GOTO jumps out of PERFORM range	R_NO_GOTO_OUT_OF_PERFORM_RANGE	SQUORE	Medium	Non Conformity	Critical	Maintainability		✎
□		Avoid mixing paragraphs and sections	R_PARA_OR_SECT_ONLY	SQUORE	Medium	Non Conformity	Minor	Maintainability		✎
□		Avoid obsolete DATA BEGIN OF OCCURS statement	R_AVOIDDATAOCCURS	SQUORE	Low	Non Conformity	Minor	Maintainability		✎

1 2 3 4 5 6 7 8 9 10 » »» »»»»

On Selection: Edit

The template selection in the project wizard

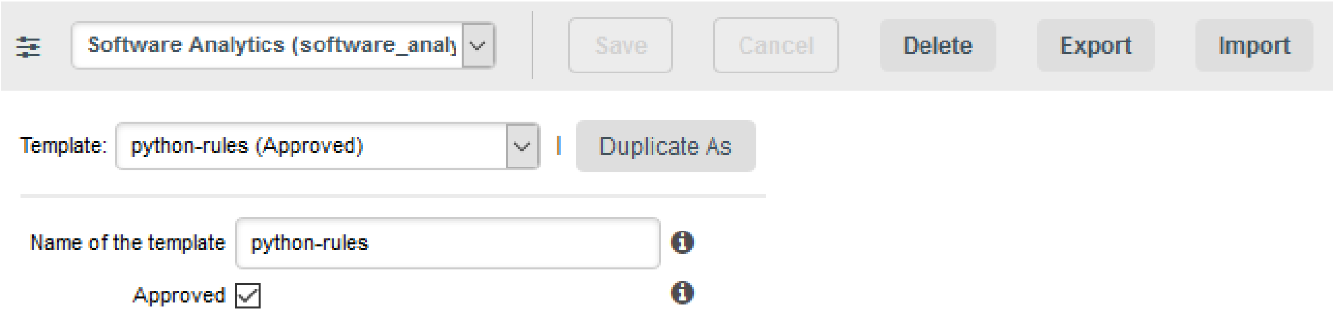
Templates can also be applied to projects from the command line using the **--rulesetTemplate** parameter:

```
--rulesetTemplate="python-rules"
```

Managing Ruleset Templates

Export and Import

Aside from creating and deleting ruleset templates, the Analysis Model Editor also allows you to export them to XML and import them again. This is useful if you want to copy your templates to another Squore Server or back them up before resetting your server.



Software Analytics (software_analy

Template: python-rules (Approved)

Name of the template: python-rules

Approved ☑

The Export / Import tools in the Analysis Model Editor

The exported XML file contains the entire ruleset.

Editing the XML template file manually before importing it as a new template is not recommended, however, it can be useful so you can modify it to determine what happens to the rules not contained in the XML, using the disableOtherRules attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateRules disableOtherRules="false">
  <!--This file is auto-generated, please don't modify.-->
  <UpdateRule measureId="R_MISRA_SAME_DEFPARAMS" disabled="false" categories=
"SCALE_SEVERITY.MINOR;SCALE_NATURE.NON_CONFORMITY;CHARACTERISTIC.PORTABILITY;SCALE_REM
EDIATION.LOW" />
...
</UpdateRules>
```

Setting disableOtherRules to true deactivates all the other rules in the model that are not specified in the template. By default, or if the attribute is not specified, its value is set to **false**.

Handling Model Upgrades

Your existing templates will be retained when you upgrade your model to add new rules or upgrade to a new version of Squore:

- Existing templates will have all the new rules as disabled (off) by default
- The default template will have all the new rules as enabled (on) by default
- Projects analysed using project-specific templates (i.e. not using a saved template in the Analysis Model Editor) will have all the new rules as enabled (on) by default

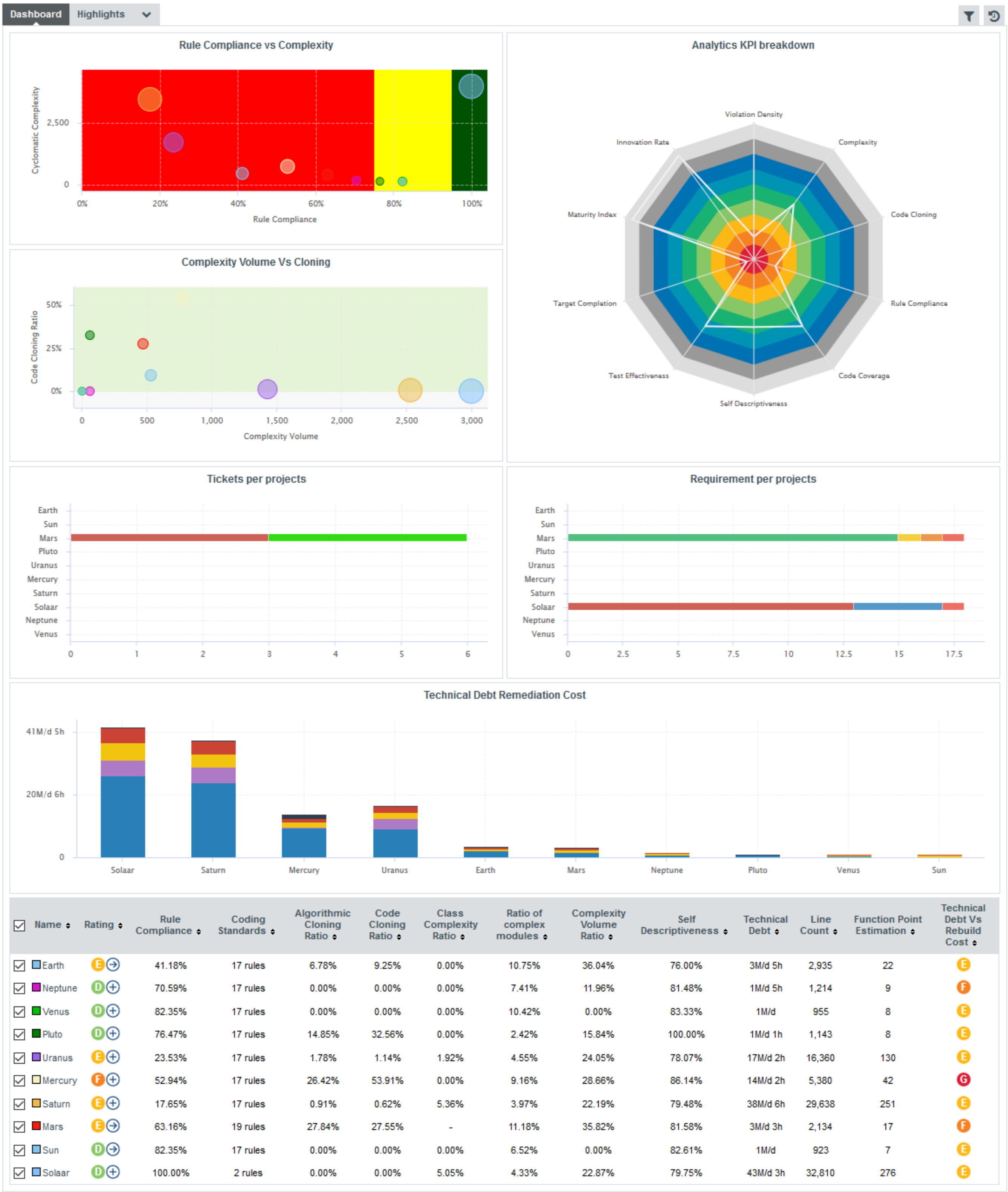
Reviewing Multiple Projects

Project Managers may be interested in monitoring several projects as a whole. Squore provides a special dashboard view which compounds information about several projects into an Model/Group Dashboard, which can help you prioritise projects according to their current status.

In order to view the Model/Group Dashboard:

1. Log into Squore with the demo user.
2. Click the model name "Software Analytics" in the Project Portfolios.

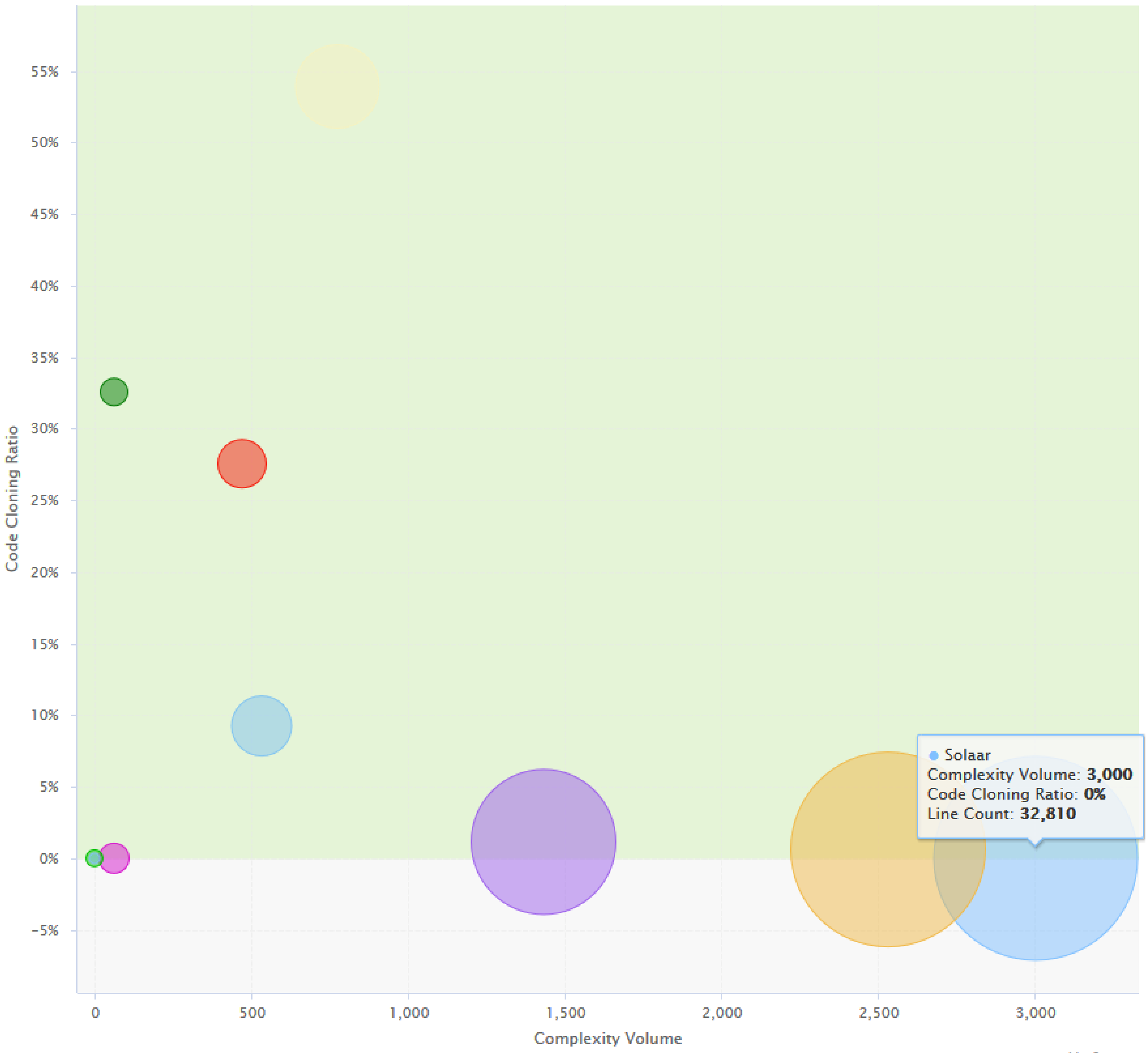
The dashboard refreshes to show the compounded information for all projects analysed with this model using charts and a summary table of the main indicators, the rating and the trend of each project.



☑ Name ⬍	Rating ⬍	Rule Compliance ⬍	Coding Standards ⬍	Algorithmic Cloning Ratio ⬍	Code Cloning Ratio ⬍	Class Complexity Ratio ⬍	Ratio of complex modules ⬍	Complexity Volume Ratio ⬍	Self Descriptiveness ⬍	Technical Debt ⬍	Line Count ⬍	Function Point Estimation ⬍	Technical Debt Vs Rebuild Cost ⬍
☑ 🟦 Earth	E →	41.18%	17 rules	6.78%	9.25%	0.00%	10.75%	36.04%	76.00%	3M/d 5h	2,935	22	E
☑ 🟪 Neptune	D ⊕	70.59%	17 rules	0.00%	0.00%	0.00%	7.41%	11.96%	81.48%	1M/d 5h	1,214	9	F
☑ 🟩 Venus	D ⊕	82.35%	17 rules	0.00%	0.00%	0.00%	10.42%	0.00%	83.33%	1M/d	955	8	E
☑ 🟩 Pluto	D ⊕	76.47%	17 rules	14.85%	32.56%	0.00%	2.42%	15.84%	100.00%	1M/d 1h	1,143	8	E
☑ 🟪 Uranus	E ⊕	23.53%	17 rules	1.78%	1.14%	1.92%	4.55%	24.05%	78.07%	17M/d 2h	16,360	130	E
☑ 🟨 Mercury	F ⊕	52.94%	17 rules	26.42%	53.91%	0.00%	9.16%	28.66%	86.14%	14M/d 2h	5,380	42	G
☑ 🟧 Saturn	E ⊕	17.65%	17 rules	0.91%	0.62%	5.36%	3.97%	22.19%	79.48%	38M/d 6h	29,638	251	E
☑ 🟥 Mars	E →	63.16%	19 rules	27.84%	27.55%	-	11.18%	35.82%	81.58%	3M/d 3h	2,134	17	F
☑ 🟦 Sun	D →	82.35%	17 rules	0.00%	0.00%	0.00%	6.52%	0.00%	82.61%	1M/d	923	7	E
☑ 🟦 Solaar	D ⊕	100.00%	2 rules	0.00%	0.00%	5.05%	4.33%	22.87%	79.75%	43M/d 3h	32,810	276	E

The Model/Group Dashboard for Software Analytics projects

In the quadrants, each project is represented as a bubble . Two indicators define the horizontal and vertical position of the bubble along the axes, while a third indicator defines the bubble size. Let's see how you should prioritise maintenance work on your project portfolio for the sample projects. Click on the **Complexity Volume Vs Cloning** quadrant to view the full version:



Complexity Volume Vs Cloning for current Software Analytics projects

In this chart, projects with a high code cloning ratio appear higher, while more complex projects appear more to the right. The size of each bubble indicates the size of the project in terms of source lines of code. Therefore, it may be easier to improve the quality of a project with more cloning but less complexity like Pluto (dark green) than a project with less cloning but more complexity (Mars, in red) As a project manager, you know that as a general rule you need to focus on moving projects towards the bottom-left corner of the chart for a healthy portfolio of projects.

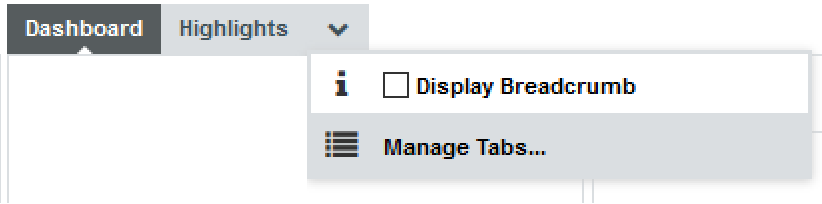
Below the quadrants, Squore displays tables with the values used in the charts so you can refine the information read in the charts. All the information shown in the analysis model dashboard can be configured by a Squore administrator. Refer to the Squore Configuration Guide for more information.

Chapter 6. Managing Your To-Do List With Squore

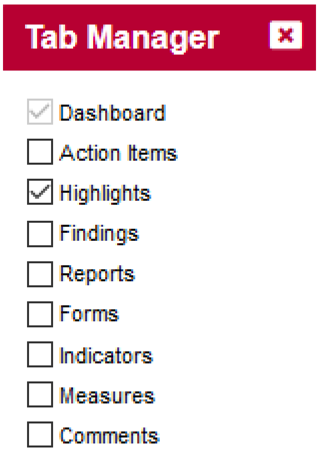
The analysis results you obtained by creating your first projects in Creating Projects and Versions and observed in Understanding Analysis Results can be drilled down further by looking at the other tabs available in the Explorer. In this chapter, you will learn how to use the information contained in Indicators, Measures, Findings and Action Items to better understand and reuse the information provided by Squore in your development workflow.

How do I understand and Improve My Ratings?

If you need more background information about the measures and indicators used in the charts and tables in the dashboard, the **Indicators**, **Measures** and **Findings** tabs can provide more details about the statistics recorded for the current artefact. Note that these tabs are not displayed by default. If you want to show them in Squore, click the Explorer Settings menu and then Manage Tabs to display the Tab Manager to enable these tabs, as shown below:



The Manage Tabs option in the Explorer Settings allows to display the Tab Manager

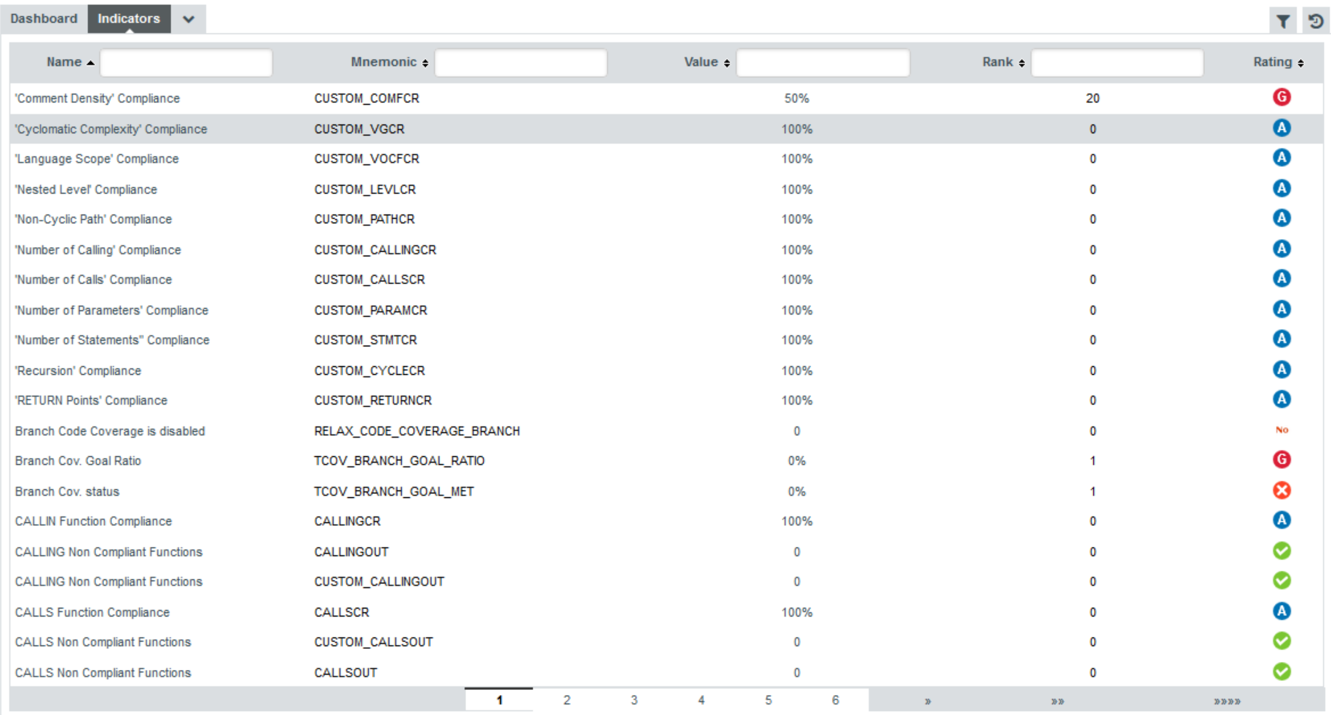

Tab Manager

☑ Dashboard
□ Action Items
☑ Highlights
□ Findings
□ Reports
□ Forms
□ Indicators
□ Measures
□ Comments

The Tab Manager allows displaying tabs hidden by default by checking them. Note that not according to your configuration, some tabs may not be removeable

If you want to understand the scale used for a particular indicator, to see for example how close you are to moving up the rating scale, you can check the scale used for this indicator in the **Indicators** tab of the dashboard.

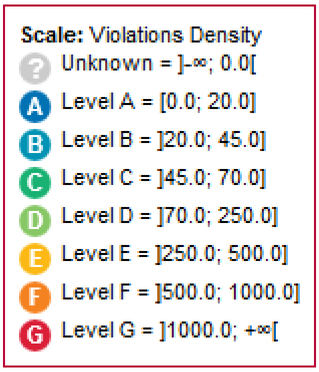
Log in and search for the artefact **DB5_backup.c** in the Neptune project, where the indicator **Maintainability Non-Conformity** is rated E. While this tells you about the current rating for this artefact, this does not tell you how to improve it. In order to learn how to improve this score, let's first take a look at the scale used for this indicator. Click the **Indicators** tab of the Explorer. The table of indicators opens, as shown below:



Name ▲	Mnemonic ⬍	Value ⬍	Rank ⬍	Rating ⬍
'Comment Density' Compliance	CUSTOM_COMFCR	50%	20	G
'Cyclomatic Complexity' Compliance	CUSTOM_VGCR	100%	0	A
'Language Scope' Compliance	CUSTOM_VOCFCR	100%	0	A
'Nested Level' Compliance	CUSTOM_LEVLCR	100%	0	A
'Non-Cyclic Path' Compliance	CUSTOM_PATHCR	100%	0	A
'Number of Calling' Compliance	CUSTOM_CALLINGCR	100%	0	A
'Number of Calls' Compliance	CUSTOM_CALLSCR	100%	0	A
'Number of Parameters' Compliance	CUSTOM_PARAMCR	100%	0	A
'Number of Statements" Compliance	CUSTOM_STMTCR	100%	0	A
'Recursion' Compliance	CUSTOM_CYCLECR	100%	0	A
'RETURN Points' Compliance	CUSTOM_RETURNCR	100%	0	A
Branch Code Coverage is disabled	RELAX_CODE_COVERAGE_BRANCH	0	0	No
Branch Cov. Goal Ratio	TCOV_BRANCH_GOAL_RATIO	0%	1	G
Branch Cov. status	TCOV_BRANCH_GOAL_MET	0%	1	✗
CALLIN Function Compliance	CALLINGCR	100%	0	A
CALLING Non Compliant Functions	CALLINGOUT	0	0	✓
CALLING Non Compliant Functions	CUSTOM_CALLINGOUT	0	0	✓
CALLS Function Compliance	CALLSCR	100%	0	A
CALLS Non Compliant Functions	CUSTOM_CALLSOUT	0	0	✓
CALLS Non Compliant Functions	CALLSOUT	0	0	✓

The indicators table for DB5_backup.c

The table lists all the indicators available for the artefact over several pages. The scale and levels available for an indicator can be viewed in a tooltip by placing your mouse over a rating. Using the filter above the "name" column, look for the entry named **Maintainability Non-Conformity**, then click its value in the rating column. The scale for the indicator indicates that the artefact is rated E because the value of the indicator is 472.09. In order to improve the score, the value would need to decrease to under 250 to be rated D, as shown below:



Scale: Violations Density
? Unknown =]-∞; 0.0[
A Level A = [0.0; 20.0]
B Level B =]20.0; 45.0]
C Level C =]45.0; 70.0]
D Level D =]70.0; 250.0]
E Level E =]250.0; 500.0]
F Level F =]500.0; 1000.0]
G Level G =]1000.0; +∞[

The scale used for the Maintainability Non-Conformity indicator

To understand how to improve the rating , you need to know how the indicator's value is computed. Clicking the indicator name in the Indicator Tree shows the following explanation in the indicator popup:

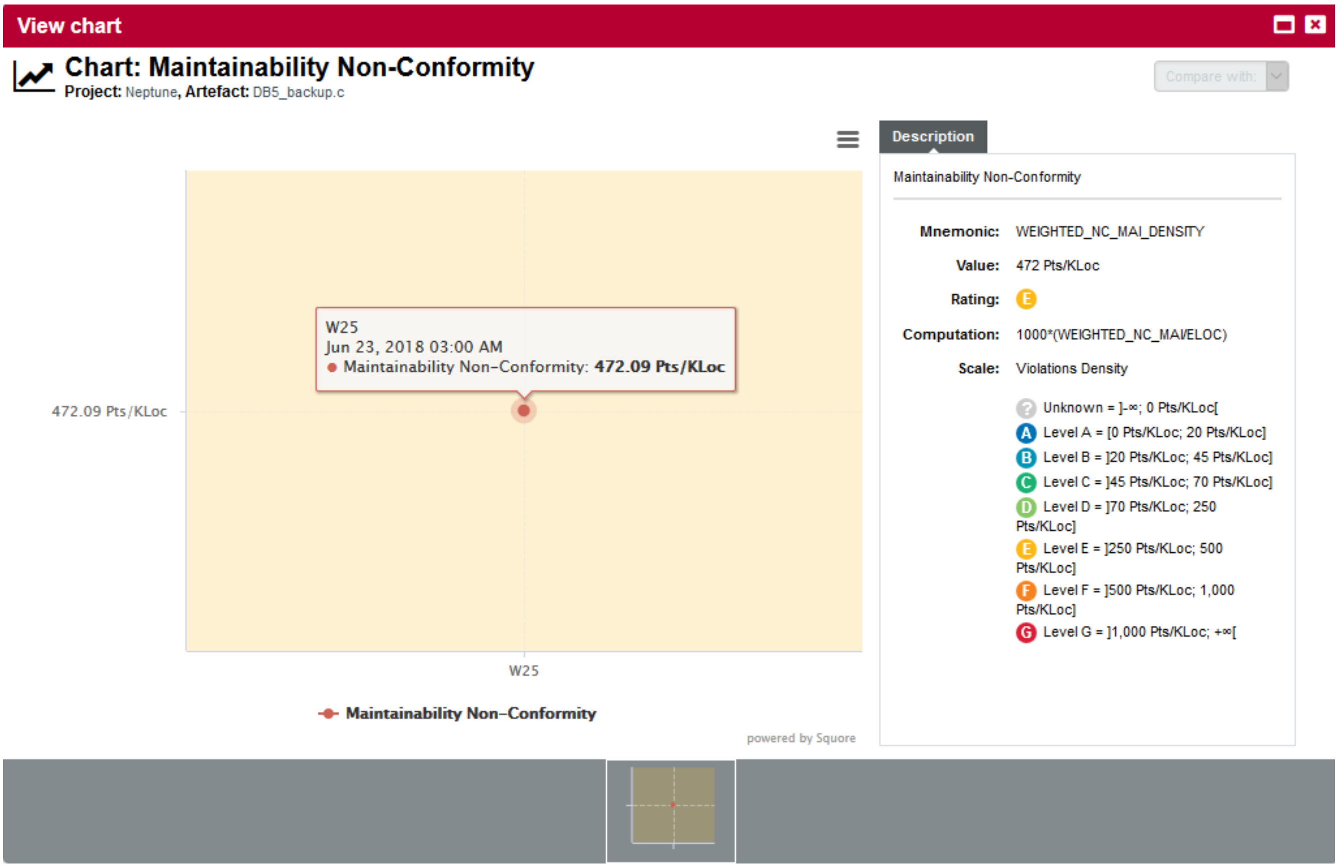


Chart: Maintainability Non-Conformity
Project: Neptune, **Artefact:** DB5_backup.c

Maintainability Non-Conformity

Mnemonic: WEIGHTED_NC_MAI_DENSITY
Value: 472 Pts/KLoc
Rating: E
Computation: 1000*(WEIGHTED_NC_MAI/ELOC)
Scale: Violations Density

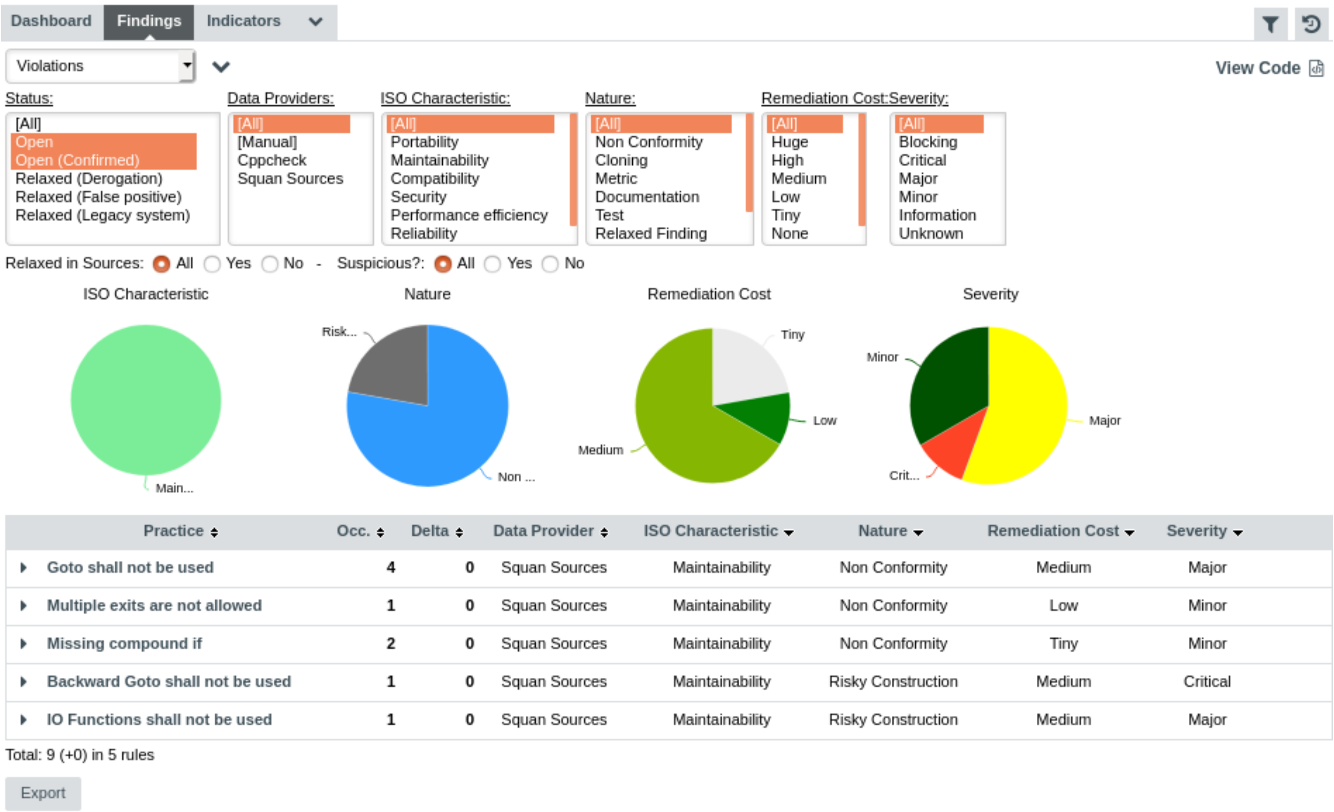
- ? Unknown =]-∞; 0 Pts/KLoc[
- A Level A = [0 Pts/KLoc; 20 Pts/KLoc]
- B Level B =]20 Pts/KLoc; 45 Pts/KLoc]
- C Level C =]45 Pts/KLoc; 70 Pts/KLoc]
- D Level D =]70 Pts/KLoc; 250 Pts/KLoc]
- E Level E =]250 Pts/KLoc; 500 Pts/KLoc]
- F Level F =]500 Pts/KLoc; 1,000 Pts/KLoc]
- G Level G =]1,000 Pts/KLoc; +∞[

The indicator popup for the Maintainability Non-Conformity indicator

The computation , i.e. the formula used to calculate the rating is *1000*(WEIGHTED_NC_MAI/ELOC)*, meaning that the indicator computes a ratio of broken Maintainability rules. To find out what these rules are, click the **Findings** tab.

Squore displays all the findings for a particular artefact in a table in the Findings tab. Next to the finding's label is a number of occurrences followed by a colour-coded delta value (red for more occurrences, green for less) compared to a previous analysis.

If you want to find out which rules are taken into account by the Maintainability Non-Conformity indicator, expand the filter to show the advanced filtering options. Highlight **Maintainability** in the *ISO Characteristics* filter to see the corresponding rules, as shown in the picture below:



Violations

Status: [All], Open, Open (Confirmed), Relaxed (Derogation), Relaxed (False positive), Relaxed (Legacy system)

Data Providers: [All], [Manual], Cppcheck, Squan Sources

ISO Characteristic: [All], Portability, Maintainability, Compatibility, Security, Performance efficiency, Reliability

Nature: [All], Non Conformity, Cloning, Metric, Documentation, Test, Relaxed Finding

Remediation Cost: [All], Huge, High, Medium, Low, Tiny, None

Severity: [All], Blocking, Critical, Major, Minor, Information, Unknown

Relaxed in Sources: ● All ○ Yes ○ No - Suspicious?: ● All ○ Yes ○ No

Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Goto shall not be used	4	0	Squan Sources	Maintainability	Non Conformity	Medium	Major
Multiple exits are not allowed	1	0	Squan Sources	Maintainability	Non Conformity	Low	Minor
Missing compound if	2	0	Squan Sources	Maintainability	Non Conformity	Tiny	Minor
Backward Goto shall not be used	1	0	Squan Sources	Maintainability	Risky Construction	Medium	Critical
IO Functions shall not be used	1	0	Squan Sources	Maintainability	Risky Construction	Medium	Major

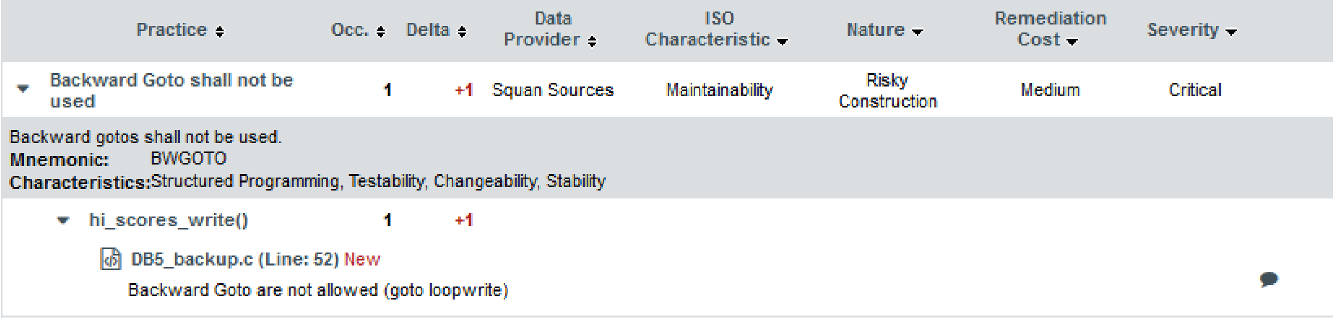
Total: 9 (+0) in 5 rules

The findings table for DB5_backup.c

You can filter violations according to many criteria, including relaxation status, origin, artefact type and other characteristics from the analysis model

The rules **BWGOTO, STDIO, NOGOTO, RETURN and COMPOUNDIF** are the rules that should be fixed in order to improve the Maintainability rating of DB5_backup.c.

You can expand the **BWGOTO** rule to show each occurrence of the rule being broken, and also review the location in the source code that breaks the rule, as shown below:



Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Backward Goto shall not be used	1	+1	Squan Sources	Maintainability	Risky Construction	Medium	Critical

Backward gotos shall not be used.
Mnemonic: BWGOTO
Characteristics: Structured Programming, Testability, Changeability, Stability

hi_scores_write() 1 +1

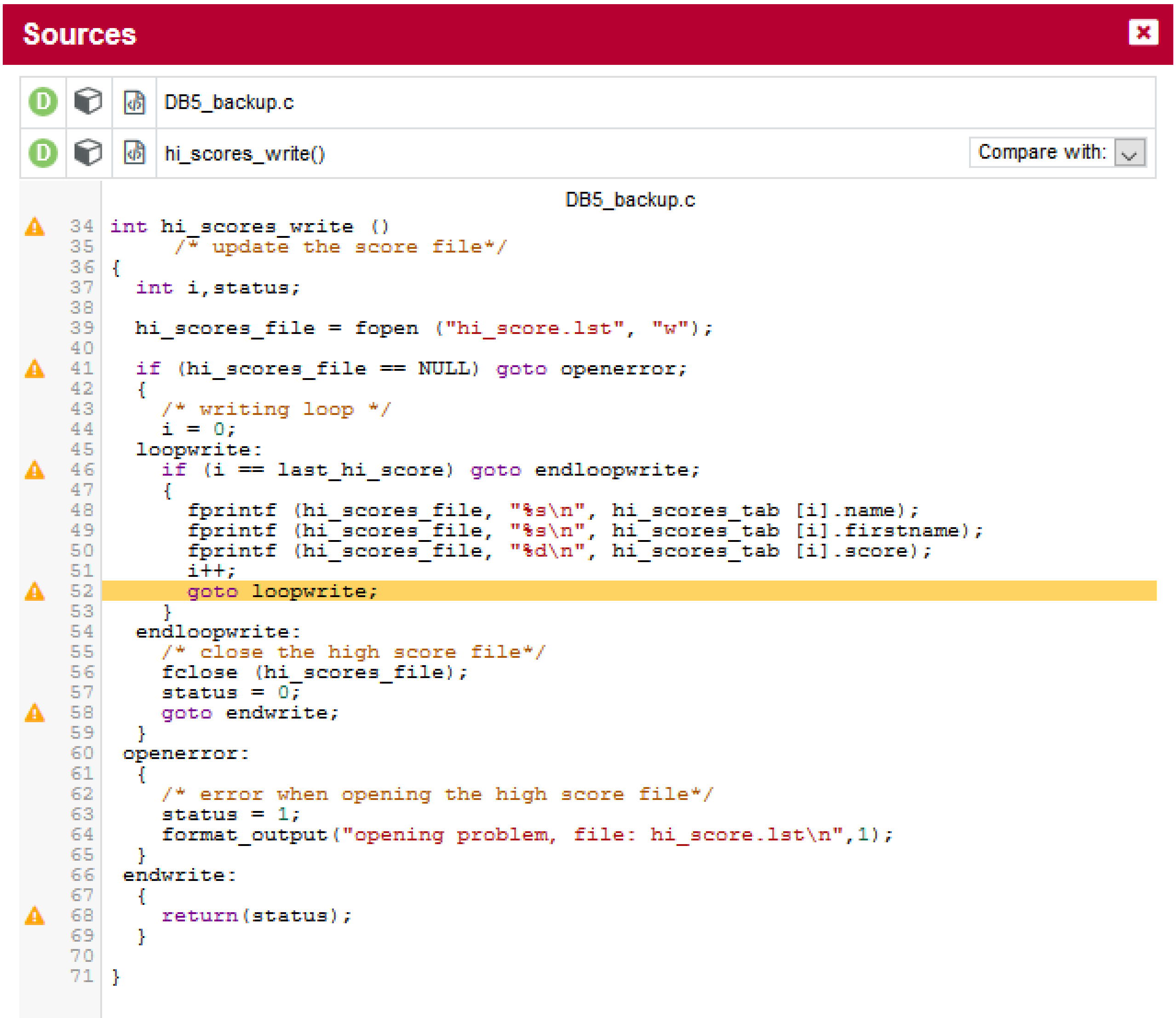
DB5_backup.c (Line: 52) New

Backward Goto are not allowed (goto loopwrite)

The location of the broken occurrences of the ***BWGOTO*** *rule*

The list of findings indicates if a finding is **New**, **Closed** or **Modified** since the reference version. Findings are traceable through time, so even if your code is modified, you can to go back to the version in which it was first detected.

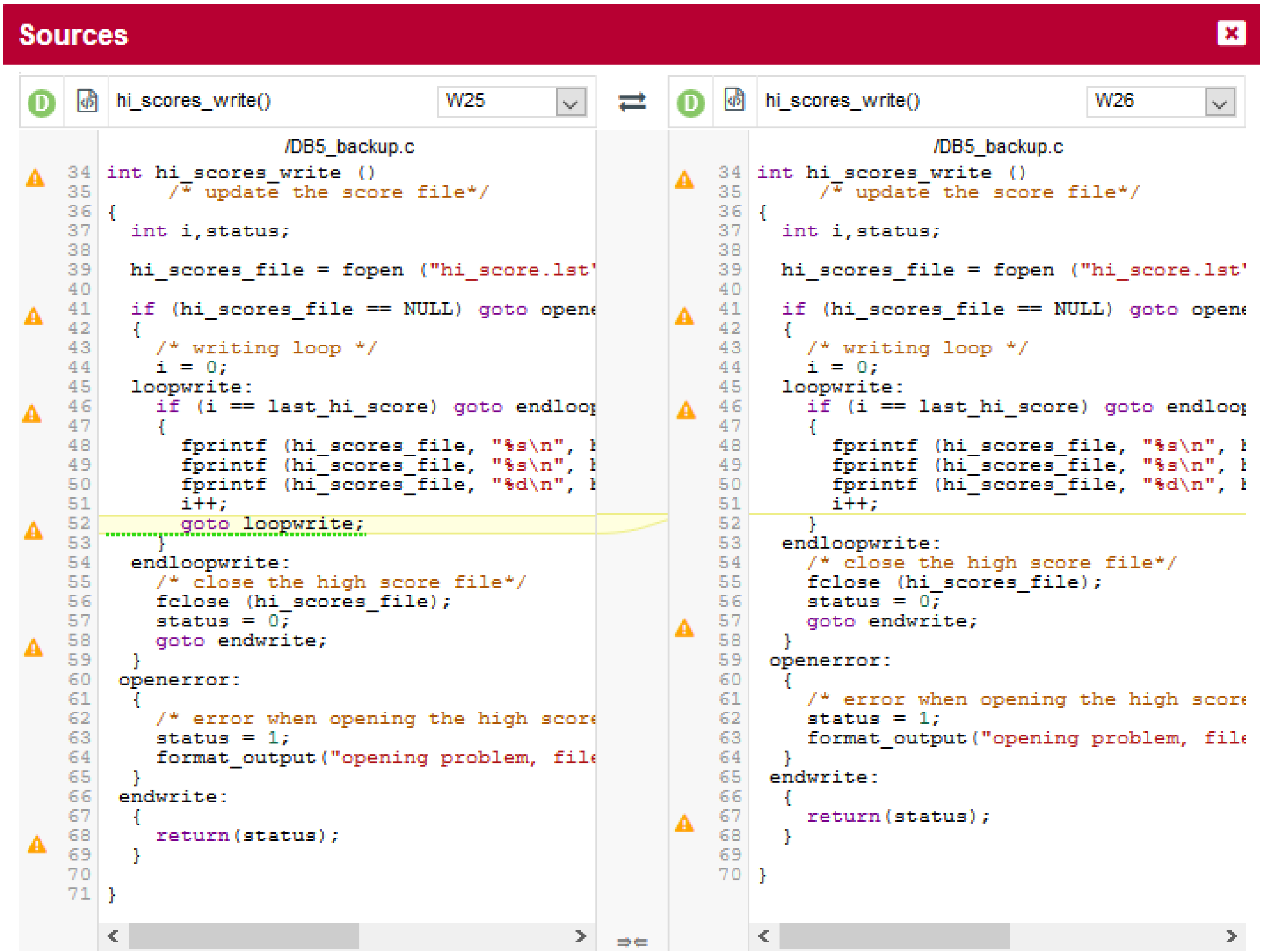
Finally, clicking on the line number for each rule-breaking occurrence opens the source code viewer in full screen so you can carry out your code review:



```
int hi_scores_write ()
     /* update the score file*/
{
  int i,status;

  hi_scores_file = fopen ("hi_score.lst", "w");

  if (hi_scores_file == NULL) goto openerror;
  {
    /* writing loop */
    i = 0;
  loopwrite:
    if (i == last_hi_score) goto endloopwrite;
    {
      fprintf (hi_scores_file, "%s\n", hi_scores_tab [i].name);
      fprintf (hi_scores_file, "%s\n", hi_scores_tab [i].firstname);
      fprintf (hi_scores_file, "%d\n", hi_scores_tab [i].score);
      i++;
      goto loopwrite;
    }
  endloopwrite:
    /* close the high score file*/
    fclose (hi_scores_file);
    status = 0;
    goto endwrite;
  }
 openerror:
  {
    /* error when opening the high score file*/
    status = 1;
    format_output("opening problem, file: hi_score.lst\n",1);
  }
 endwrite:
  {
    return(status);
  }

}
```

The source code viewer highlighting the first occurrence of ***BWGOTO***

The source code viewer allows comparing the code against another version of the code. Select a version name in the **Compare with:** list to switch to diff mode, as shown below:



/DB5_backup.c

```
int hi_scores_write ()
     /* update the score file*/
{
  int i,status;

  hi_scores_file = fopen ("hi_score.lst'

  if (hi_scores_file == NULL) goto opene
  {
    /* writing loop */
    i = 0;
  loopwrite:
    if (i == last_hi_score) goto endloo
    {
      fprintf (hi_scores_file, "%s\n",
      fprintf (hi_scores_file, "%s\n",
      fprintf (hi_scores_file, "%d\n",
      i++;
      goto loopwrite;
    }
  endloopwrite:
    /* close the high score file*/
    fclose (hi_scores_file);
    status = 0;
    goto endwrite;
  }
 openerror:
  {
    /* error when opening the high score
    status = 1;
    format_output("opening problem, file
  }
 endwrite:
  {
    return(status);
  }
}
```

/DB5_backup.c

```
int hi_scores_write ()
     /* update the score file*/
{
  int i,status;

  hi_scores_file = fopen ("hi_score.lst'

  if (hi_scores_file == NULL) goto opene
  {
    /* writing loop */
    i = 0;
  loopwrite:
    if (i == last_hi_score) goto endloo
    {
      fprintf (hi_scores_file, "%s\n",
      fprintf (hi_scores_file, "%s\n",
      fprintf (hi_scores_file, "%d\n",
      i++;
    }
  endloopwrite:
    /* close the high score file*/
    fclose (hi_scores_file);
    status = 0;
    goto endwrite;
  }
 openerror:
  {
    /* error when opening the high score
    status = 1;
    format_output("opening problem, file
  }
 endwrite:
  {
    return(status);
  }
}
```

The source code viewer in diff mode

In diff mode, use the top arrows to switch the left and right panes, and the bottom arrows to turn synchronised scrolling on or off. Characters that were removed are underlined in green, while characters that were added are underlined in red.

Analysing findings helps to improve the quality of the code in your project. There is much more you can do with the Findings tab by using the built-in filters to detect regressions and improvements:

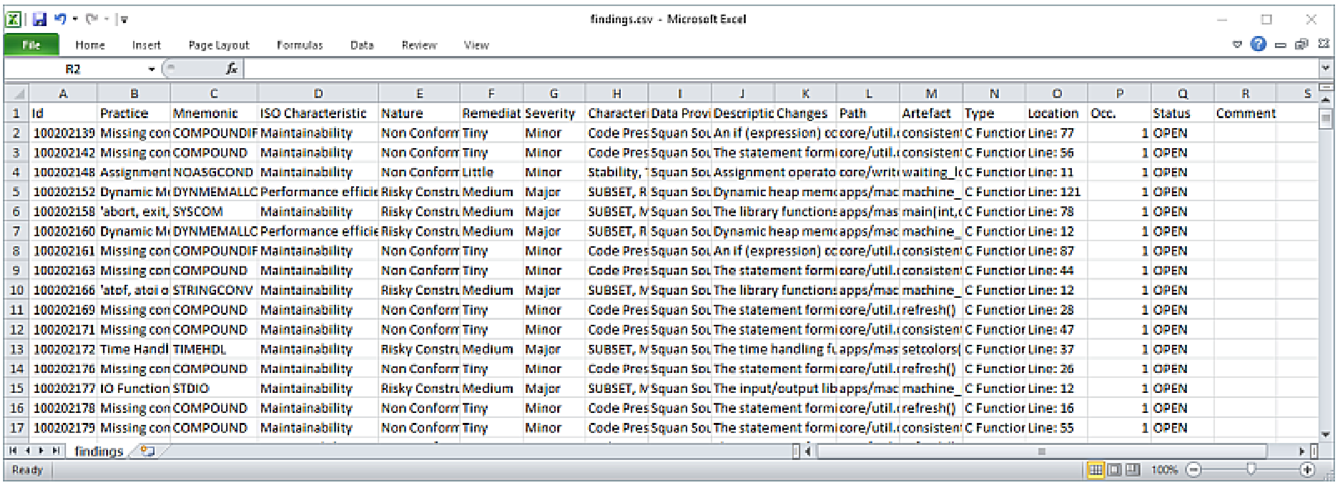
- **Violations:** displays all the rules violated in this version
- **Lost Practices:** displays violations that are new in this version since a specified version
- **Acquired Practices:** displays all violations not occurring anymore in this version since a previous version
- **Deteriorated Practices:** displays all violations with more occurrences in this version than in a previous version
- **Improved Practices:** displays all violations with fewer occurrences in this version than in a previous version
- **New violations:** displays all the new violations since a previous version
- **Fixed violations:** displays all the violations fixed since a previous version
- **All changed violations:** displays all the rules where a change in the number of violations was detected, essentially providing the combination of New violations and Fixed violations in one list
- **Project ruleset:** displays all the rules checked by the project, i.e. the violated ones as well as the

ones that are not

By default, the Findings tab displays violations compared to the previous analysis, but you can refine the search by adjusting the **Reference** drop-down list (under the Explorer Settings menu) that contains all the versions analysed in your project.

You can learn about more automated ways to review and fix code in How Do I Review And Manage Action Items Flagged by Squore?.

You can click the Export button at the bottom of the list of findings, to generate a CSV file of the findings displayed in the user interface. The contents of the file reflect your current filter selections on the page. The following is a CSV export for the Findings of the Earth project, which you can download in full here.



findings.csv - Microsoft Excel

R2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Id	Practice	Mnemonic	ISO Characteristic	Nature	Remediat	Severity	Character	Data Provi	Descriptic	Changes	Path	Artefact	Type	Location	Occ.	Status	Comment	
2	100202139	Missing con	COMPOUNDIF	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	An if (expression) cc		core/util.	consistent	C Functior	Line: 77	1	OPEN		
3	100202142	Missing con	COMPOUND	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	The statement formi		core/util.	consistent	C Functior	Line: 56	1	OPEN		
4	100202148	Assignment	NOASGCOND	Maintainability	Non Conform	Little	Minor	Stability, T	Squan Sou	Assignment operato		core/writ	waiting_lc	C Functior	Line: 11	1	OPEN		
5	100202152	Dynamic M	DYNMEMALLC	Performance efficie	Risky Constru	Medium	Major	SUBSET, R	Squan Sou	Dynamic heap mem		apps/mac	machine_	C Functior	Line: 121	1	OPEN		
6	100202158	'abort, exit,	SYSCOM	Maintainability	Risky Constru	Medium	Major	SUBSET, M	Squan Sou	The library function		apps/mas	main(int,c	C Functior	Line: 78	1	OPEN		
7	100202160	Dynamic M	DYNMEMALLC	Performance efficie	Risky Constru	Medium	Major	SUBSET, R	Squan Sou	Dynamic heap mem		apps/mac	machine_	C Functior	Line: 12	1	OPEN		
8	100202161	Missing con	COMPOUNDIF	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	An if (expression) cc		core/util.	consistent	C Functior	Line: 87	1	OPEN		
9	100202163	Missing con	COMPOUND	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	The statement formi		core/util.	consistent	C Functior	Line: 44	1	OPEN		
10	100202166	'atof, atoi a	STRINGCONV	Maintainability	Risky Constru	Medium	Major	SUBSET, M	Squan Sou	The library function		apps/mac	machine_	C Functior	Line: 12	1	OPEN		
11	100202169	Missing con	COMPOUND	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	The statement formi		core/util.	refresh()	C Functior	Line: 28	1	OPEN		
12	100202171	Missing con	COMPOUND	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	The statement formi		core/util.	consistent	C Functior	Line: 47	1	OPEN		
13	100202172	Time Handl	TIMEHDL	Maintainability	Risky Constru	Medium	Major	SUBSET, M	Squan Sou	The time handling fu		apps/mas	setcolors(	C Functior	Line: 37	1	OPEN		
14	100202176	Missing con	COMPOUND	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	The statement formi		core/util.	refresh()	C Functior	Line: 26	1	OPEN		
15	100202177	IO Function	STDIO	Maintainability	Risky Constru	Medium	Major	SUBSET, M	Squan Sou	The input/output lib		apps/mac	machine_	C Functior	Line: 12	1	OPEN		
16	100202178	Missing con	COMPOUND	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	The statement formi		core/util.	refresh()	C Functior	Line: 16	1	OPEN		
17	100202179	Missing con	COMPOUND	Maintainability	Non Conform	Tiny	Minor	Code Pres	Squan Sou	The statement formi		core/util.	consistent	C Functior	Line: 55	1	OPEN		

A CSV export of the findings of the Earth project

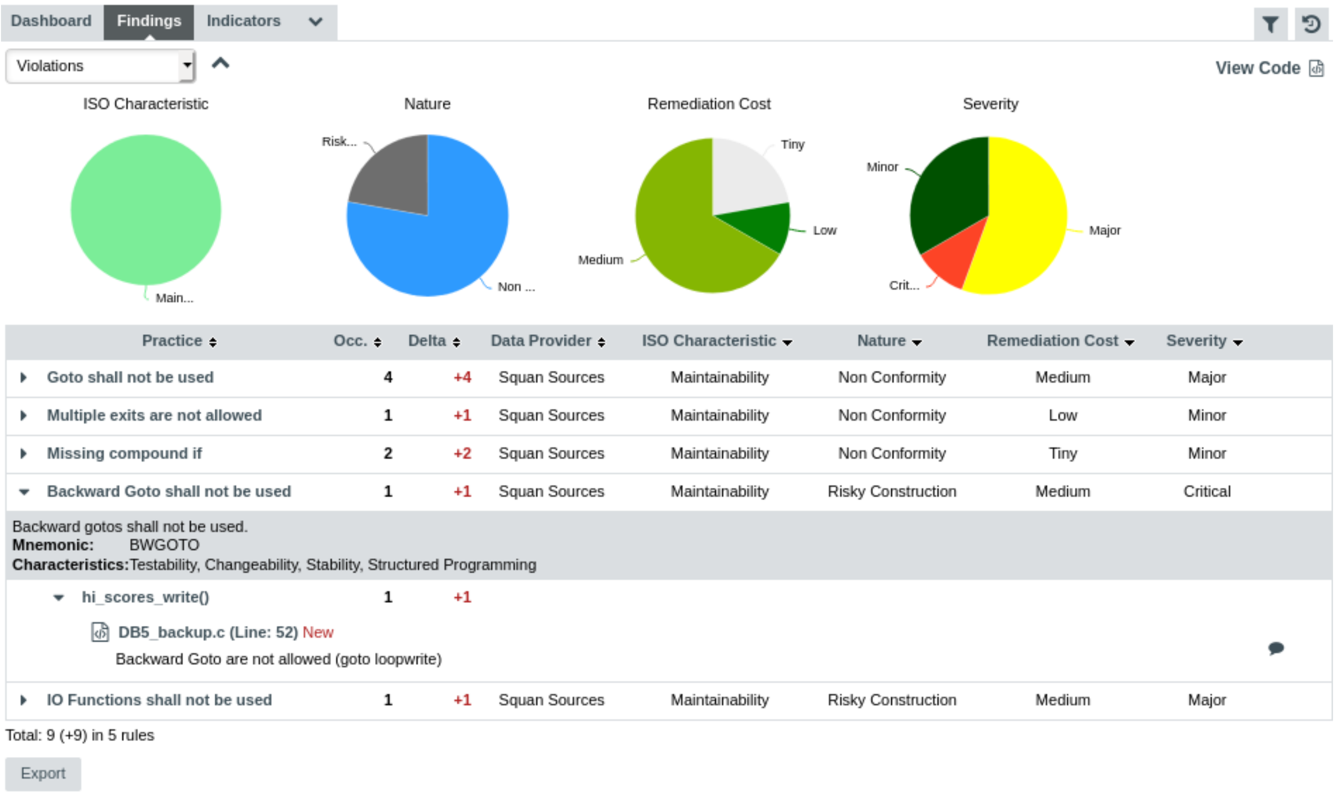
If the Export button is greyed out, your licence does not include the option to export data to CSV files.

Relaxing Findings

Relaxing from the Findings Tab of Squore

If you realise that a violation found during an analysis is not justified, you can relax it from the Findings tab of the Explorer.

In the example below, a we consider that a **Backward goto** violation should not be reported, because it is a **Relaxed (False positive)**. Let's start by locating the violation in the Findings tab:



	Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
▸	Goto shall not be used	4	+4	Squan Sources	Maintainability	Non Conformity	Medium	Major
▸	Multiple exits are not allowed	1	+1	Squan Sources	Maintainability	Non Conformity	Low	Minor
▸	Missing compound if	2	+2	Squan Sources	Maintainability	Non Conformity	Tiny	Minor
▾	Backward Goto shall not be used	1	+1	Squan Sources	Maintainability	Risky Construction	Medium	Critical

Backward gotos shall not be used.
Mnemonic: BWGOTO
Characteristics: Testability, Changeability, Stability, Structured Programming

▾ hi_scores_write() 1 +1

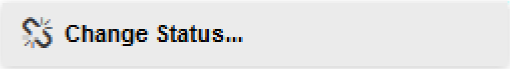
DB5_backup.c (Line: 52) New
Backward Goto are not allowed (goto loopwrite)

	Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
▸	IO Functions shall not be used	1	+1	Squan Sources	Maintainability	Risky Construction	Medium	Major

Total: 9 (+9) in 5 rules

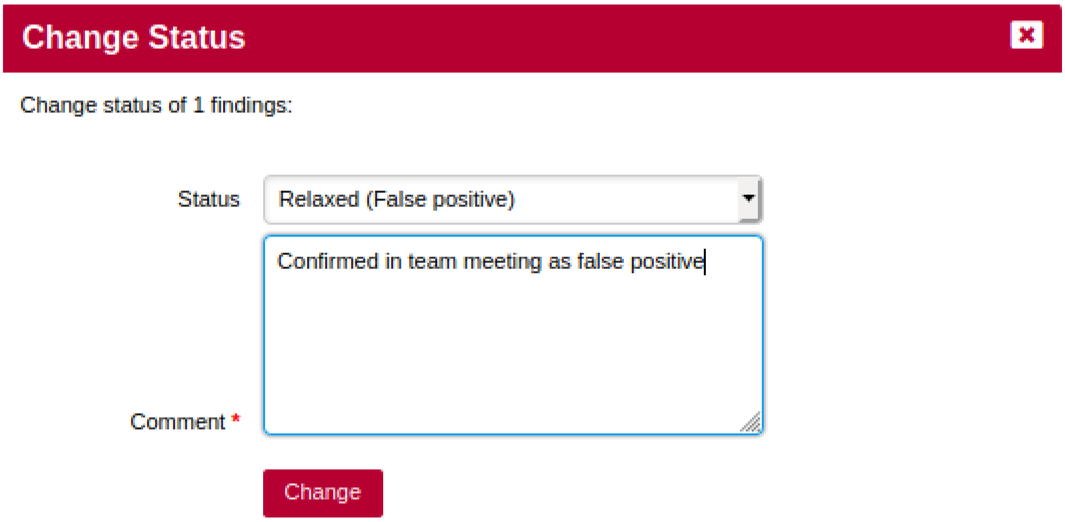
The backward goto violation we want to relax

When you hover over the menu icon for the violation, you can display a context menu that allows you to change the status of the finding:



The finding context menu

Click **Change Status...** to view the available statuses for the violation.



Change Status

Change status of 1 findings:

Status: Relaxed (False positive)

Comment *: Confirmed in team meeting as false positive

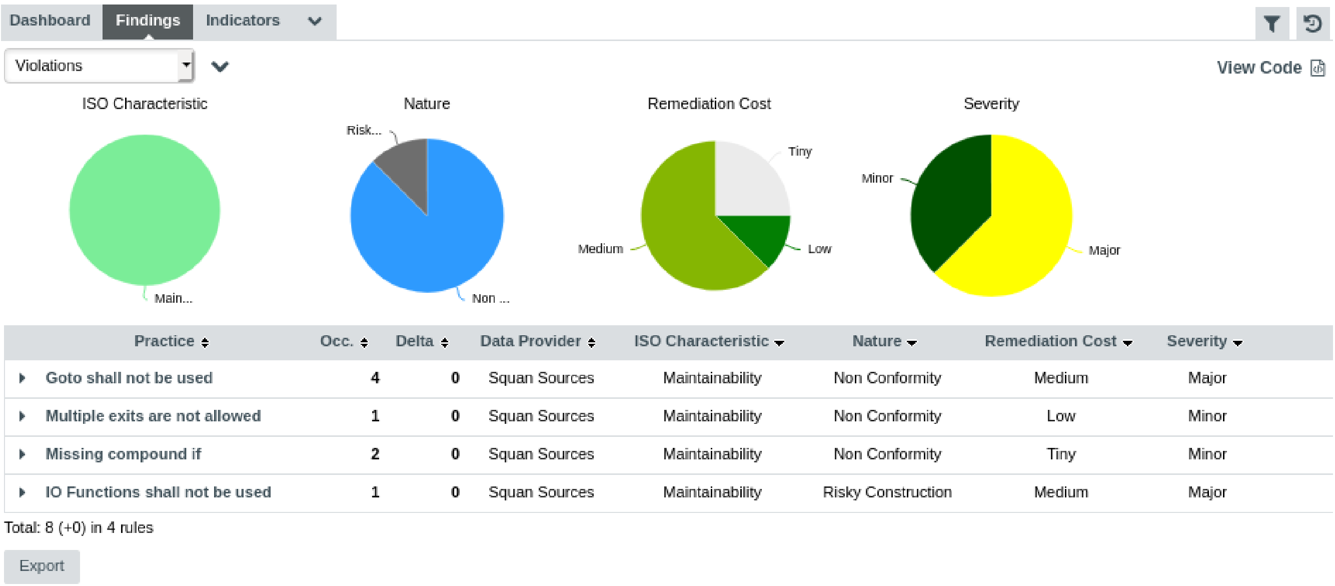
The Change Status Popup

Type a justification or comment for the relaxation and choose from one of the reasons for relaxing the violation:

- **Open** is the default status for new findings, which means no relaxation
- **Open (Confirmed)** means no relaxation, plus indicates that this findings has been reviewed and is status is validated as open.

- **Relaxed (Derogation)** means that you are relaxing a true violation for an exceptional reason
- **Relaxed (False positive)** can be used to work around a violation that was falsely detected by a data provider
- **Relaxed (Legacy system)** is used when a violation is detected in a piece of code that was analysed but cannot or will not be fixed.

In our example, select **Relaxed (False positive)**, enter a comment and click **Change**. The Findings page will reload and the violation will be gone from the list.



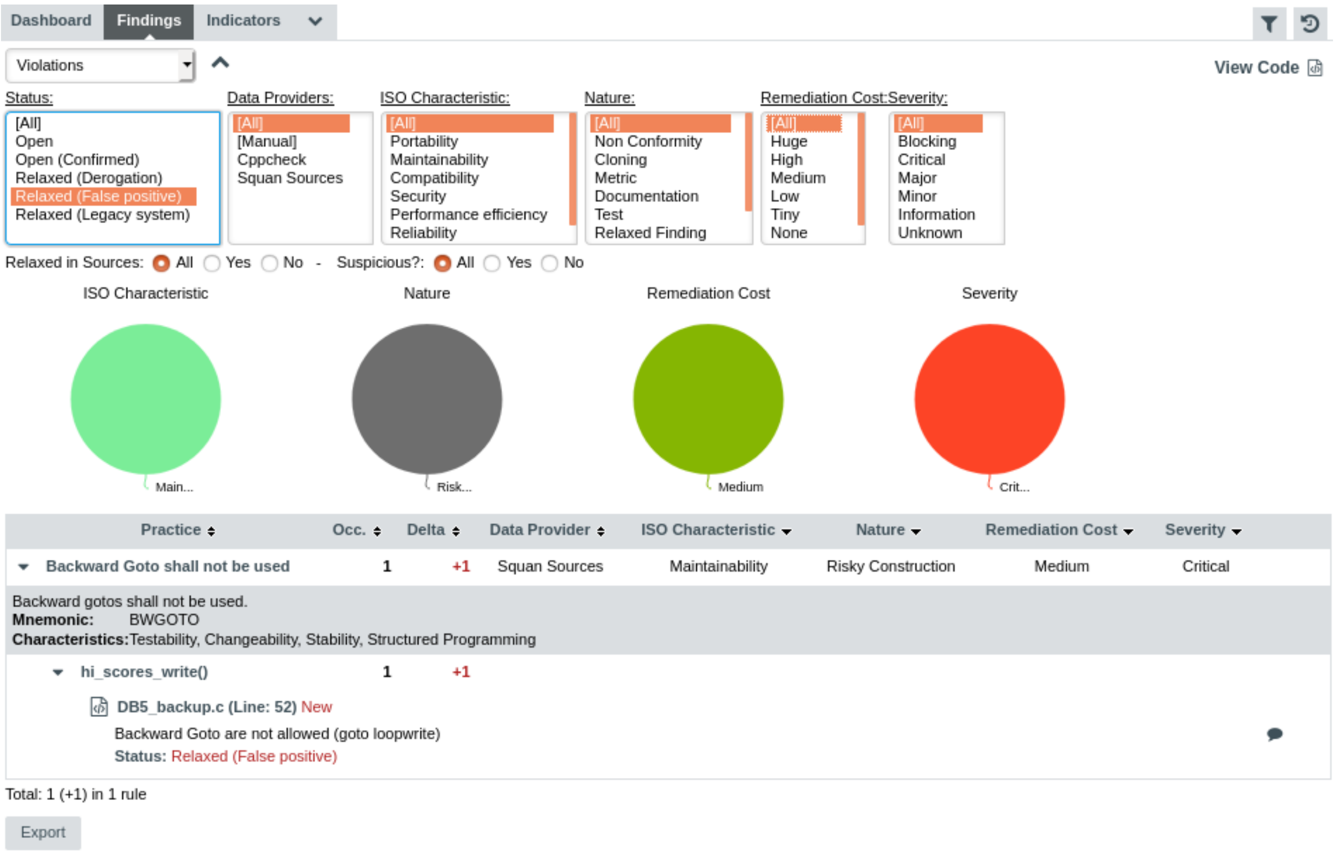
Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Goto shall not be used	4	0	Squan Sources	Maintainability	Non Conformity	Medium	Major
Multiple exits are not allowed	1	0	Squan Sources	Maintainability	Non Conformity	Low	Minor
Missing compound if	2	0	Squan Sources	Maintainability	Non Conformity	Tiny	Minor
IO Functions shall not be used	1	0	Squan Sources	Maintainability	Risky Construction	Medium	Major

Total: 8 (+0) in 4 rules

Export

The updated findings list after relaxing the backwards goto

Relaxed findings are never deleted. If you want to review the list of findings that were relaxed in your project, adjust the filter on the Findings tab to display relaxed findings, as shown below;



Status:	Data Providers:	ISO Characteristic:	Nature:	Remediation Cost:	Severity:
[All]	[All]	[All]	[All]	[All]	[All]
Open	[Manual]	Portability	Non Conformity	Huge	Blocking
Open (Confirmed)	Cppcheck	Maintainability	Cloning	High	Critical
Relaxed (Derogation)	Squan Sources	Compatibility	Metric	Medium	Major
Relaxed (False positive)		Security	Documentation	Low	Minor
Relaxed (Legacy system)		Performance efficiency	Test	Tiny	Information
		Reliability	Relaxed Finding	None	Unknown

Relaxed in Sources: ● All ○ Yes ○ No - Suspicious?: ● All ○ Yes ○ No

Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Backward Goto shall not be used	1	+1	Squan Sources	Maintainability	Risky Construction	Medium	Critical

Backward gotos shall not be used.
Mnemonic: BWGOTO
Characteristics: Testability, Changeability, Stability, Structured Programming

hi_scores_write() 1 +1

DB5_backup.c (Line: 52) New

Backward Goto are not allowed (goto loopwrite)

Status: Relaxed (False positive)

Total: 1 (+1) in 1 rule

Export

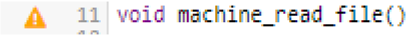
The filtered list of findings for the project, including the backwards goto ***Relaxed (False positive)***

You can relax an individual finding, all findings for an artefact, or an entire rule at once. Note that instead of relaxing a rule, you can deactivate rules by using the Analysis Model Editor (see Analysis Model Editor).

If you are looking to relax entire artefacts instead of findings, you can do that from the Artefact Tree , as described in Relaxing and Excluding Artefacts).

Relaxing from Source Code View

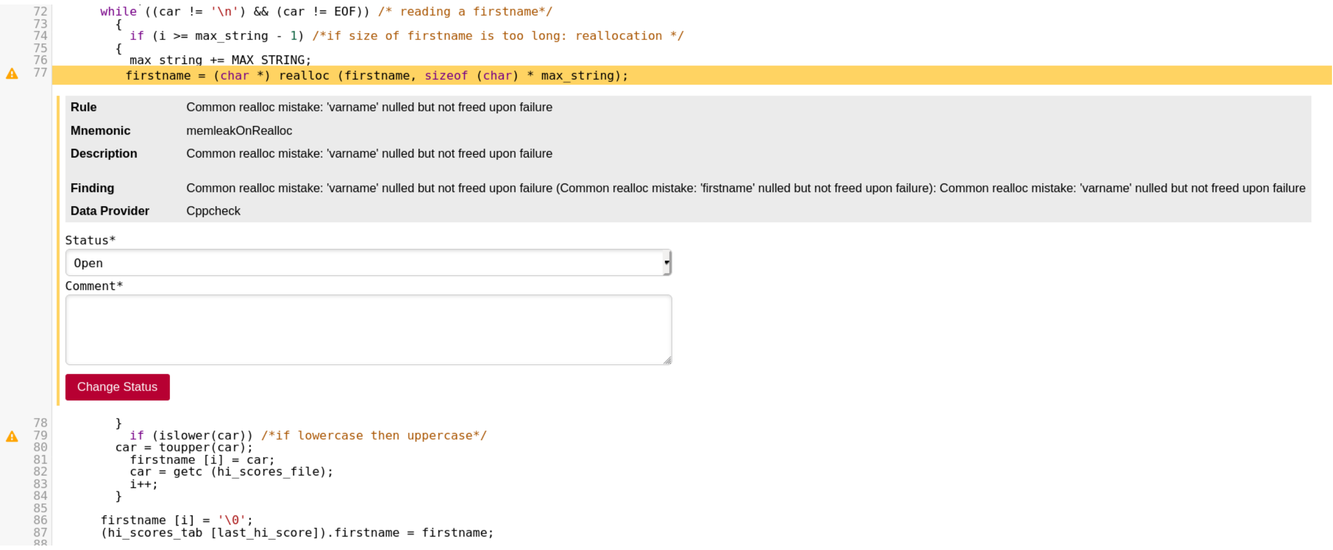
You can also relax Findings from the Source Code view. In View Source Code, Some Findings are directly indicated in the Source Code view with a Warning sign



```
void machine_read_file()
```

The warning sign indicates the presence of a Finding

If clicking on one Finding Warning sign, you can open a Finding shortcut menu which allows you to do some things



```
        while ((car != '\n') && (car != EOF)) /* reading a firstname*/
          {
            if (i >= max_string - 1) /*if size of firstname is too long: reallocation */
          {
            max_string += MAX_STRING;
           firstname = (char *) realloc (firstname, sizeof (char) * max_string);
```

Rule	Common realloc mistake: 'varname' nulled but not freed upon failure
Mnemonic	memleakOnRealloc
Description	Common realloc mistake: 'varname' nulled but not freed upon failure
Finding	Common realloc mistake: 'varname' nulled but not freed upon failure (Common realloc mistake: 'firstname' nulled but not freed upon failure): Common realloc mistake: 'varname' nulled but not freed upon failure
Data Provider	Cppcheck

Status*: Open

Comment*:

```
          }
            if (islower(car)) /*if lowercase then uppercase*/
          car = toupper(car);
            firstname [i] = car;
            car = getc (hi_scores_file);
            i++;
          }

        firstname [i] = '\0';
        (hi_scores_tab [last_hi_score]).firstname = firstname;
```

Finding shortcut menu

From here, you can see basic informations about your Finding: Rule, Mnemonic, Rule description, Finding description and the Data Provider which emitted it. You can also relax your findings from there:

As in Findings tab, type a justification or comment for the relaxation and choose from one of the reasons for relaxing the violation:

- **Open** is the default status for new findings, which means no relaxation
- **Open (Confirmed)** means no relaxation, plus indicates that this findings has been reviewed and is status is validated as open.
- **Relaxed (Derogation)** means that you are relaxing a true violation for an exceptional reason
- **Relaxed (False positive)** can be used to work around a violation that was falsely detected by a data provider
- **Relaxed (Legacy system)** is used when a violation is detected in a piece of code that was analysed but cannot or will not be fixed.

Then, click on the button to validate the relaxation.

Relaxing Violations in Code

Squore provides a violation relaxation mechanism that is triggered via comments found in the source code itself. There are two pre-requisites for relaxation to work:

- The model used to analyse your source code must implement a special rule called **R_RELAX** for relaxation to take place.
- You need to know the mnemonic of the violated rule you want to relax, in order to use it as a key in your comment.

Squore interprets comments formatted in one of these three ways:

1. **Inline Relaxation**

   This syntax is used to relax violations on the current line.

   ```
   some code; /* %RELAX<keys> : Text to justify the relaxation */
   ```

2. **Relax Next Line**

   This syntax is used to relax a violation on the first following line that is not a comment. In the example the text of the justification will be: "Text to justify the relaxation the text of the justification continues while lines are made of comments only"

   ```
   /* >RELAX<keys> : Text to justify the relaxation */
   /* the text of the justification continues while */
   /* lines are made of comments only */
   some code;
   ```

3. **Block Relaxation**

   This syntax is used to relax violations in an entire block of code.

   ```
   /* {{ RELAX<keys> : Text to justify the relaxation */
   /* like for format 2 text can be on more than one line */
   int my_func() {
      /* contains many violations */
      ...
   }
   /* }} RELAX<keys> */
   ```

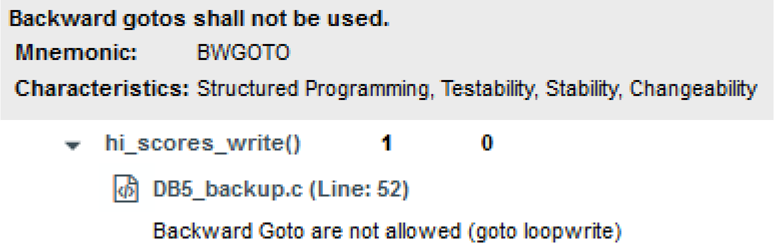
<keys> can be one of the following:

- **<*>:** relax all violations
- **<MNEMO>:** relax violations of the rule MNEMO
- **<MNEMO1,MNEMO2,...,MNEMOn>:** relax violations of rules MNEMO1 and MNEMO2 ... and MNEMOn

The relaxed violations are still shown in the Findings page after the next analysis, but they appear under the rule R_RELAX, showing the mnemonic of the relaxed violation and the justification text.

As an example, this is how you would relax the violations of the rule Backward goto for Maintainability Non-Conformity in Neptune:

1. click the violation of **Backward goto** on the Findings page to find the rule's mnemonic (BWGOTO) and the location of the finding (DB5_backup.c line 52).



Backward gotos shall not be used.
Mnemonic: BWGOTO
Characteristics: Structured Programming, Testability, Stability, Changeability

hi_scores_write() 1 0

DB5_backup.c (Line: 52)

Backward Goto are not allowed (goto loopwrite)

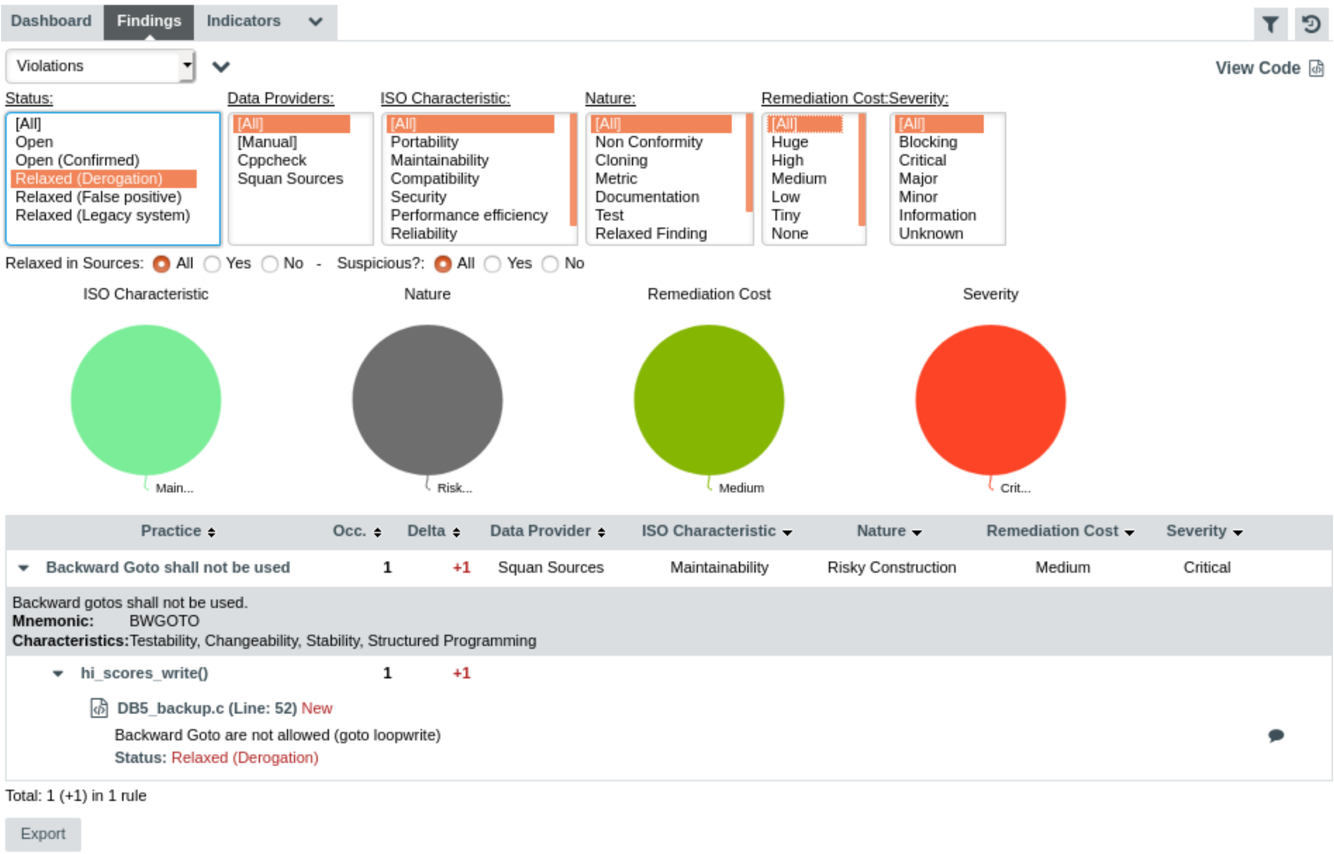
The details of the Backward goto violation

1. Edit the code of the sample project to relax the violation as shown below.

```
goto loopwrite; /* %RELAX<BWGOTO> : This backward goto is acceptable in our code.
*/
```

2. Create a new version of the project.

On the Findings page, the violation now visible if you select to display derogations in the filter:



Status:	Data Providers:	ISO Characteristic:	Nature:	Remediation Cost:	Severity:
[All]	[All]	[All]	[All]	[All]	[All]
Open	[Manual]	Portability	Non Conformity	Huge	Blocking
Open (Confirmed)	Cppcheck	Maintainability	Cloning	High	Critical
Relaxed (Derogation)	Squan Sources	Compatibility	Metric	Medium	Major
Relaxed (False positive)		Security	Documentation	Low	Minor
Relaxed (Legacy system)		Performance efficiency	Test	Tiny	Information
		Reliability	Relaxed Finding	None	Unknown

Relaxed in Sources: ● All ○ Yes ○ No - Suspicious?: ● All ○ Yes ○ No

Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Backward Goto shall not be used	1	+1	Squan Sources	Maintainability	Risky Construction	Medium	Critical

Backward gotos shall not be used.
Mnemonic: BWGOTO
Characteristics: Testability, Changeability, Stability, Structured Programming

hi_scores_write() 1 +1

DB5_backup.c (Line: 52) New

Backward Goto are not allowed (goto loopwrite)
Status: Relaxed (Derogation)

Total: 1 (+1) in 1 rule

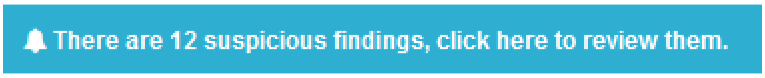
Export

The relaxed violation is visible when displaying derogations

Using the **Relaxed in Sources** toggle in the filter options, you can choose to show or hide violations that were relaxed in source code.

Suspicious Findings

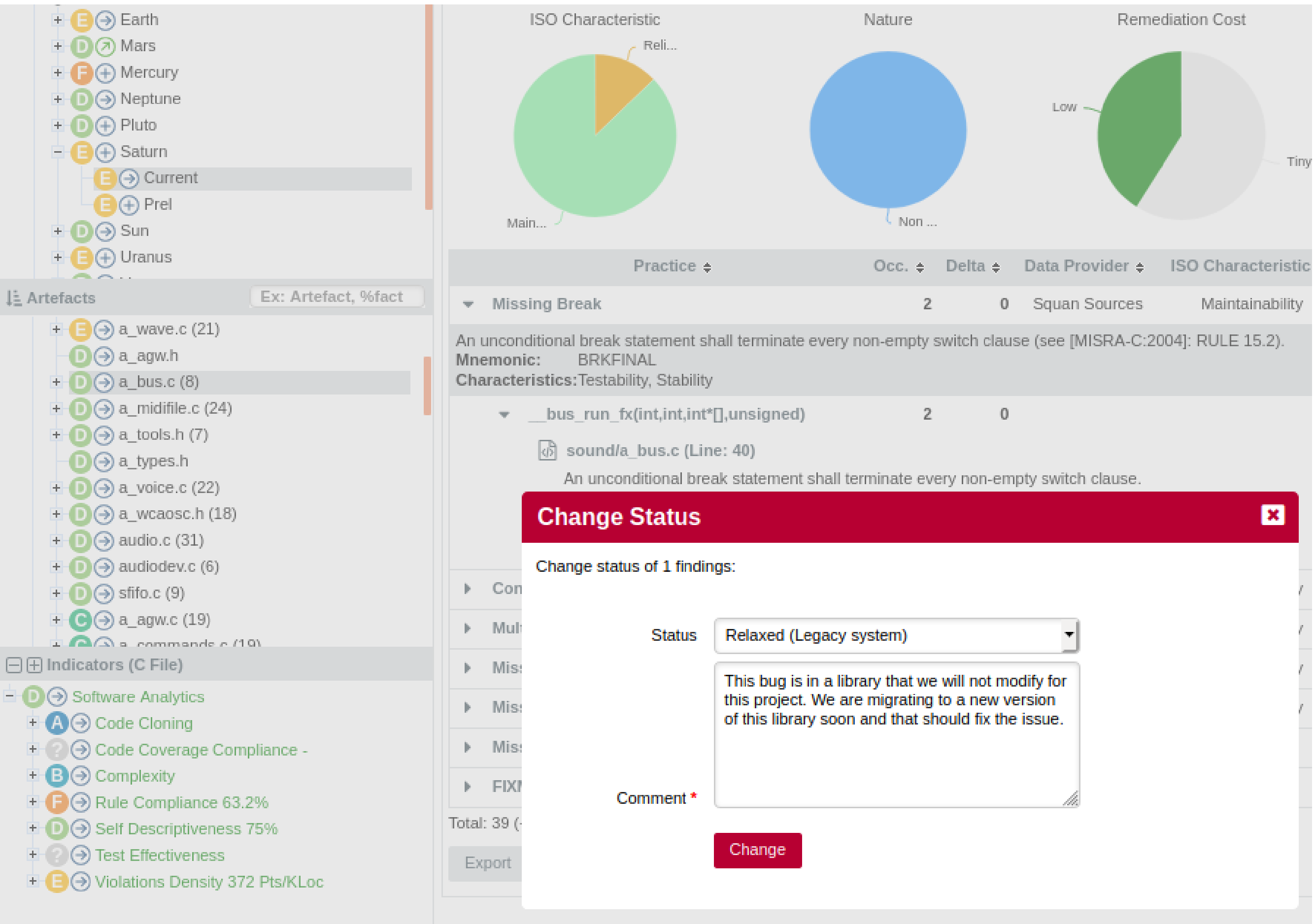
After you have relaxed or confirmed findings, Squore will check for source code changes around the location of the finding and will alert you if it should be re-examined by flagging it as suspicious. The suspicious state is a flag that is automatically added to state of the findings (the state is not affected).



There are 12 suspicious findings, click here to review them.

The warning banner for projects containing suspicious findings

To see suspicious findings in action, we will relax the Missing Break at line 40 in *audio/a_bus.c* in the project called **Saturn**. Click the Current node in the portfolio, find the artefact in the project and enter a relaxation comment for the Missing Break at line 40:



An unconditional break statement shall terminate every non-empty switch clause (see [MISRA-C:2004]: RULE 15.2).
Mnemonic: BRKFINAL
Characteristics: Testability, Stability

An unconditional break statement shall terminate every non-empty switch clause.

Change Status

Change status of 1 findings:

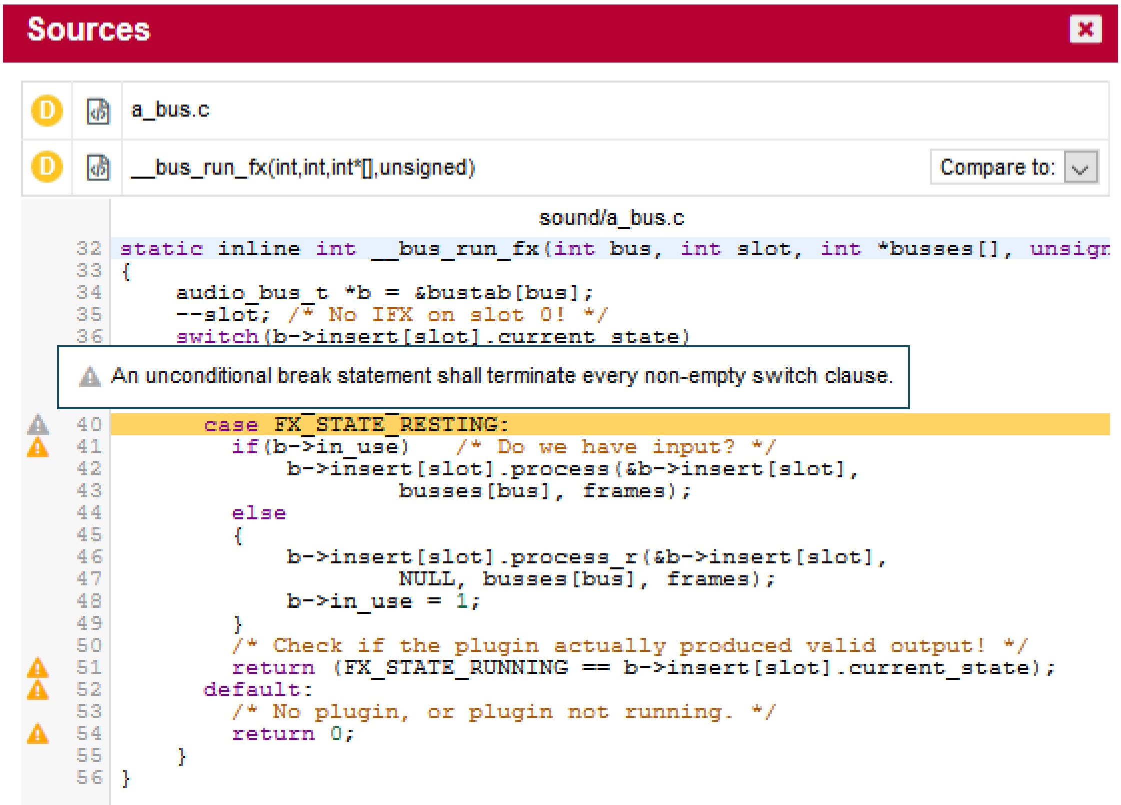
Status: Relaxed (Legacy system)

Comment *: This bug is in a library that we will not modify for this project. We are migrating to a new version of this library soon and that should fix the issue.

Change

Relaxing the Missing Break at line 40

When saving the finding relaxation, the source code viewer changes the marker for the finding to a grey warning sign to indicate that the violation was relaxed.



```
static inline int __bus_run_fx(int bus, int slot, int *busses[], unsigr
{
    audio_bus_t *b = &bustab[bus];
    --slot; /* No IFX on slot 0! */
    switch(b->insert[slot].current_state)
```

An unconditional break statement shall terminate every non-empty switch clause.

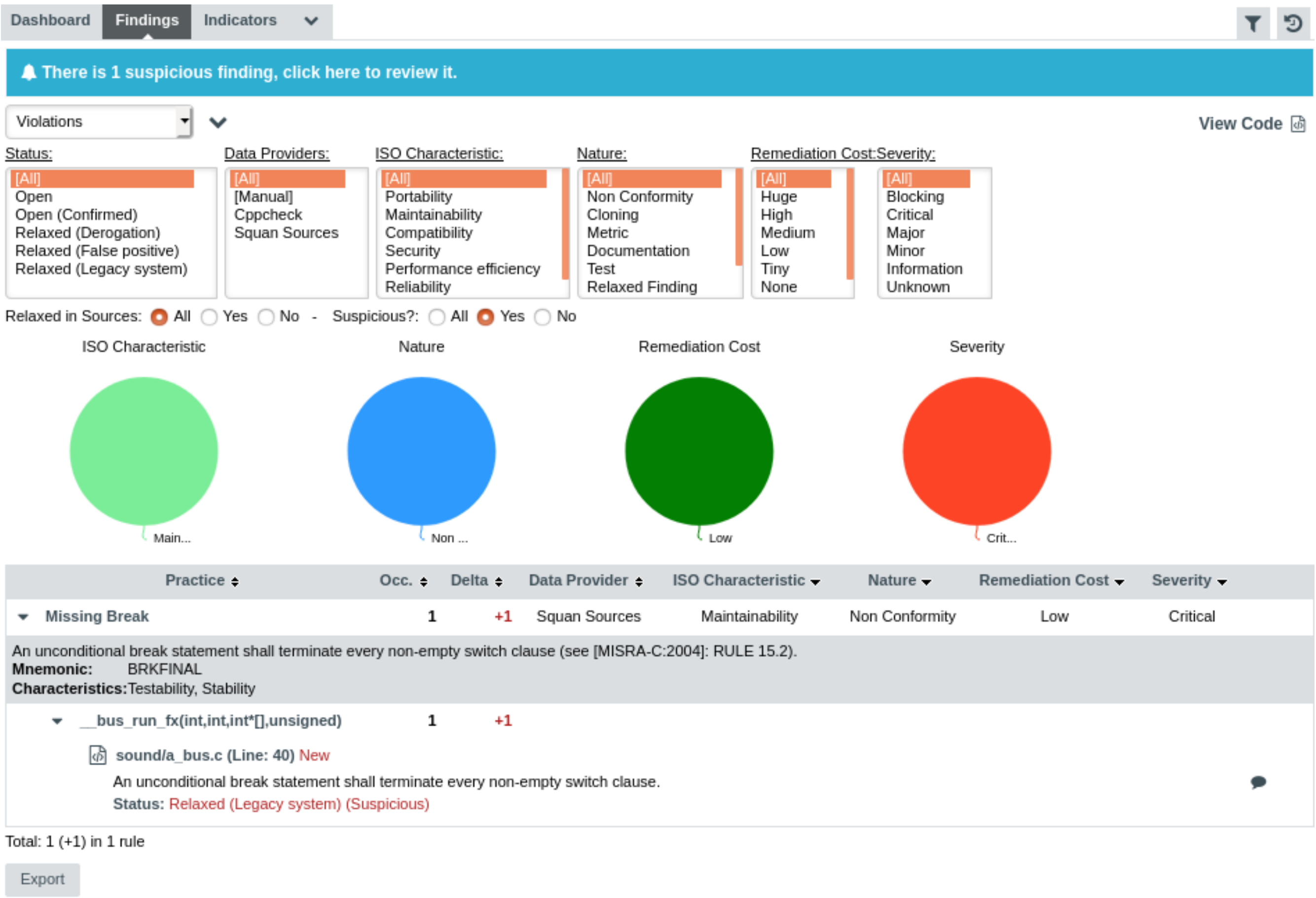
```
      case FX_STATE_RESTING:
        if(b->in_use)    /* Do we have input? */
            b->insert[slot].process(&b->insert[slot],
                    busses[bus], frames);
        else
        {
            b->insert[slot].process_r(&b->insert[slot],
                    NULL, busses[bus], frames);
            b->in_use = 1;
        }
        /* Check if the plugin actually produced valid output! */
        return (FX_STATE_RUNNING == b->insert[slot].current_state);
      default:
        /* No plugin, or plugin not running. */
        return 0;
    }
}
```

Grey and yellow markers in the source code viewer for relaxed and normal findings

The next version of the file included in the library does not seem to include any fix for the specific violation, but instead uses a renamed variable in the artefact where the finding is:

```
static inline int __bus_run_fx(int bus, int in_slot, int *busses[], unsigned frames)
{
    audio_bus_t *b = {bustab}[bus];
    --in_slot;  /* No IFX on in_slot 0! */
    switch(b->insert[in_slot].current_state)
    {
      case FX_STATE_RUNNING:
      case FX_STATE_SILENT:
      case FX_STATE_RESTING:
        if(b->in_use)   /* Do we have input? */
            b->insert[in_slot].process({b}->insert[in_slot],
                    busses[bus], frames);
        else
        {
            b->insert[in_slot].process_r({b}->insert[in_slot],
                    NULL, busses[bus], frames);
            b->in_use = 1;
        }
        /* Check if the plugin actually produced valid output! */
        return (FX_STATE_RUNNING == b->insert[in_slot].current_state);
      default:
        /* No plugin, or plugin not running. */
        return 0;
    }
}
```

Since the relaxed violation is still there, analysing this code triggers the suspicious finding warning in the web interface to alert you in case you need to revise the finding status now that the code has changed. Click the banner to reveal the suspicious finding, as shown below:



There is 1 suspicious finding, click here to review it.

Relaxed in Sources: ● All ○ Yes ○ No - Suspicious?: ○ All ● Yes ○ No

Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
Missing Break	1	+1	Squan Sources	Maintainability	Non Conformity	Low	Critical

An unconditional break statement shall terminate every non-empty switch clause (see [MISRA-C:2004]: RULE 15.2).
Mnemonic: BRKFINAL
Characteristics: Testability, Stability

__bus_run_fx(int,int,int*[],unsigned) 1 +1

sound/a_bus.c (Line: 40) New

An unconditional break statement shall terminate every non-empty switch clause.

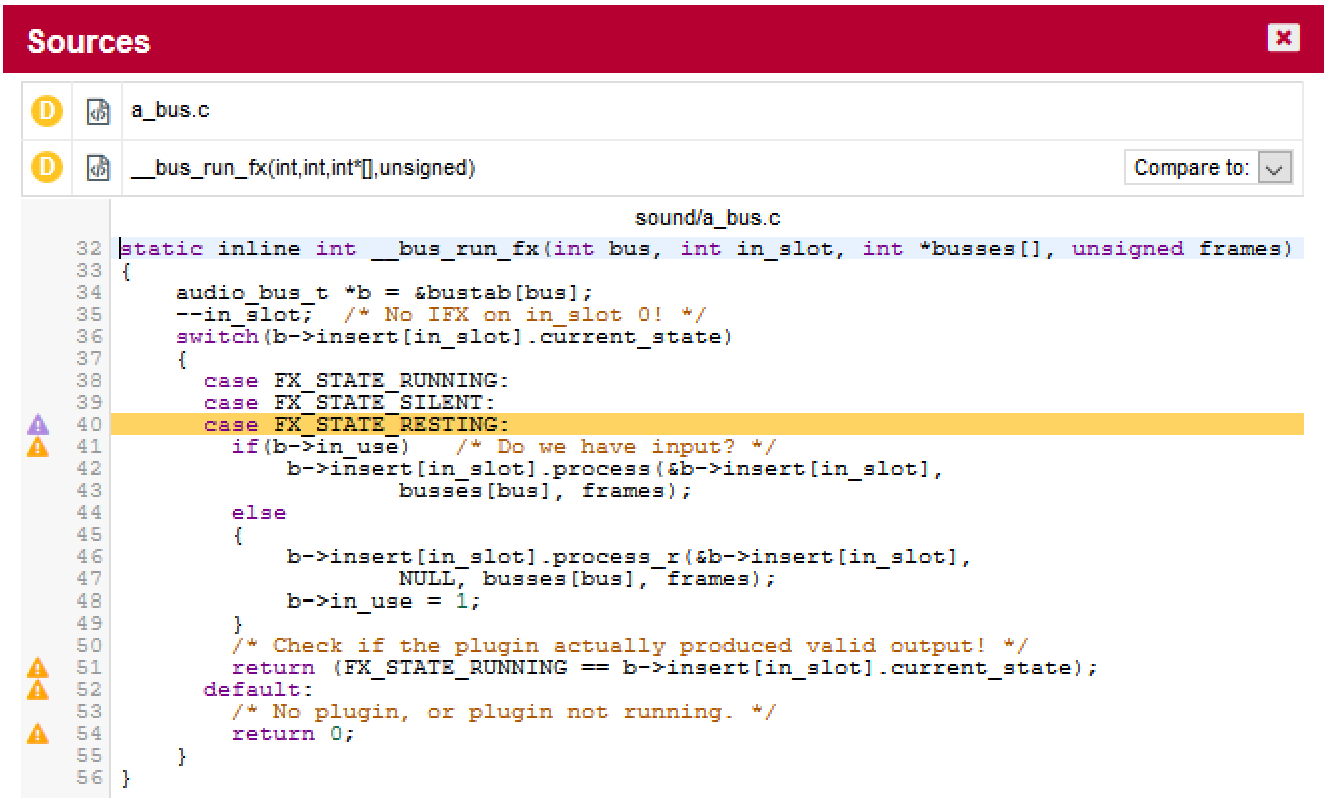
Status: Relaxed (Legacy system) (Suspicious)

Total: 1 (+1) in 1 rule

Revealed suspicious findings using the warning banner

Clicking the banner automatically applies a filter that lets you view the suspicious findings, but you can also set up this filter manually using the **Suspicious?** option on the Findings tab.

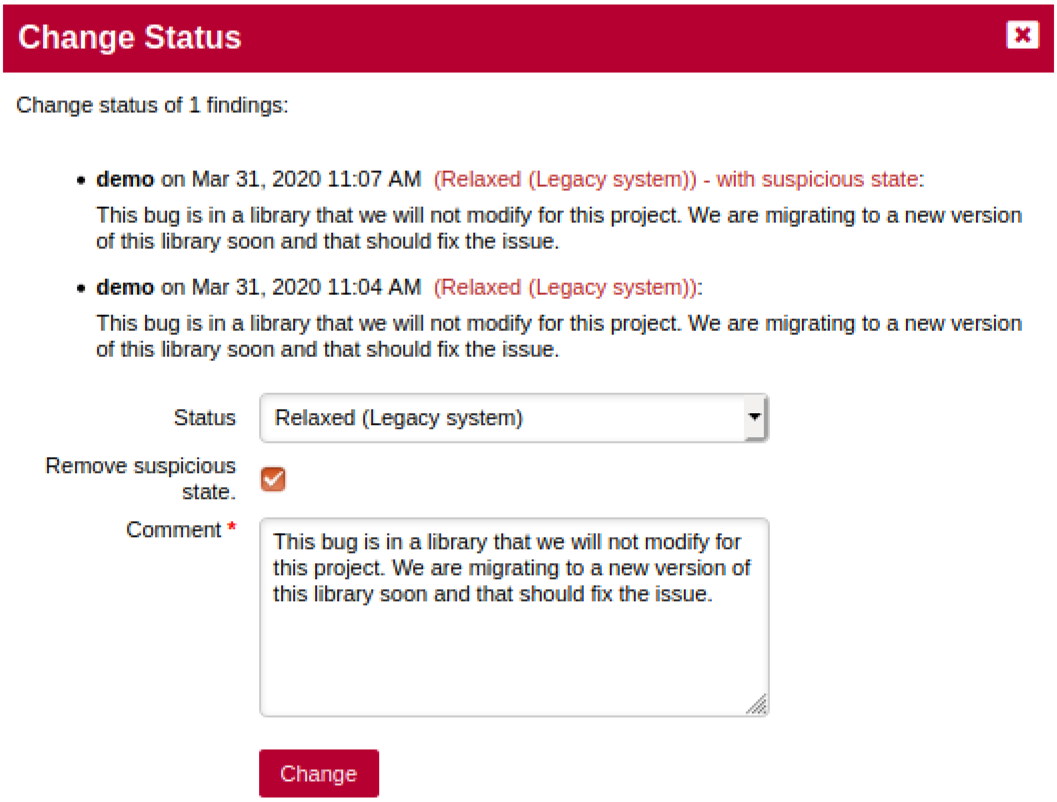
Note that the violation is still relaxed with the **Relaxed (Legacy system)** status, but the **Suspicious** flag was attached to it. Click the source code icon next to the finding to view the source code in this new version. The marker for the finding is now purple, which is the colour used to highlight suspicious findings:



```
static inline int __bus_run_fx(int bus, int in_slot, int *busses[], unsigned frames)
{
    audio_bus_t *b = &bustab[bus];
    --in_slot;  /* No IFX on in_slot 0! */
    switch(b->insert[in_slot].current_state)
    {
      case FX_STATE_RUNNING:
      case FX_STATE_SILENT:
      case FX_STATE_RESTING:
        if(b->in_use)    /* Do we have input? */
            b->insert[in_slot].process(&b->insert[in_slot],
                    busses[bus], frames);
        else
        {
            b->insert[in_slot].process_r(&b->insert[in_slot],
                    NULL, busses[bus], frames);
            b->in_use = 1;
        }
        /* Check if the plugin actually produced valid output! */
        return (FX_STATE_RUNNING == b->insert[in_slot].current_state);
      default:
        /* No plugin, or plugin not running. */
        return 0;
    }
}
```

Purple marker for suspicious findings in source code viewer

After reviewing the finding, you can remove the suspicious flag by opening the **Change Status...** dialog again and change the relaxation status or comment, or just remove the suspicious flag:



Change status of 1 findings:

- **demo** on Mar 31, 2020 11:07 AM (Relaxed (Legacy system)) - with suspicious state:

  This bug is in a library that we will not modify for this project. We are migrating to a new version of this library soon and that should fix the issue.

- **demo** on Mar 31, 2020 11:04 AM (Relaxed (Legacy system)):

  This bug is in a library that we will not modify for this project. We are migrating to a new version of this library soon and that should fix the issue.

Status: Relaxed (Legacy system)

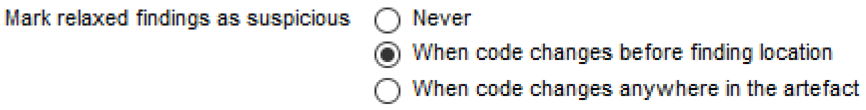
Remove suspicious state. ☑

Comment *: This bug is in a library that we will not modify for this project. We are migrating to a new version of this library soon and that should fix the issue.

Removing the suspicious flag of a finding

When you save your changes, the suspicious warning banner disappears, as there are no more suspicious findings to review in the project.

Detection of suspicious findings is activated by default and is a parameter of Squan Sources that can be tweaked by modifying the following parameters in the project wizard:



Mark relaxed findings as suspicious ○ Never
● When code changes before finding location
○ When code changes anywhere in the artefact

The settings for suspicious finding detection in Squan Sources

Relaxing and Excluding Artefacts

In this section, you will learn how to relax or exclude entire artefacts directly from the Artefact Tree instead of relaxing findings. Relaxing artefacts ensures that their metrics do not impact the rating of the project, however, data providers will still generate findings for the relaxed artefacts. Excluding artefacts hides them from the project, and findings and action items are no longer created for them.

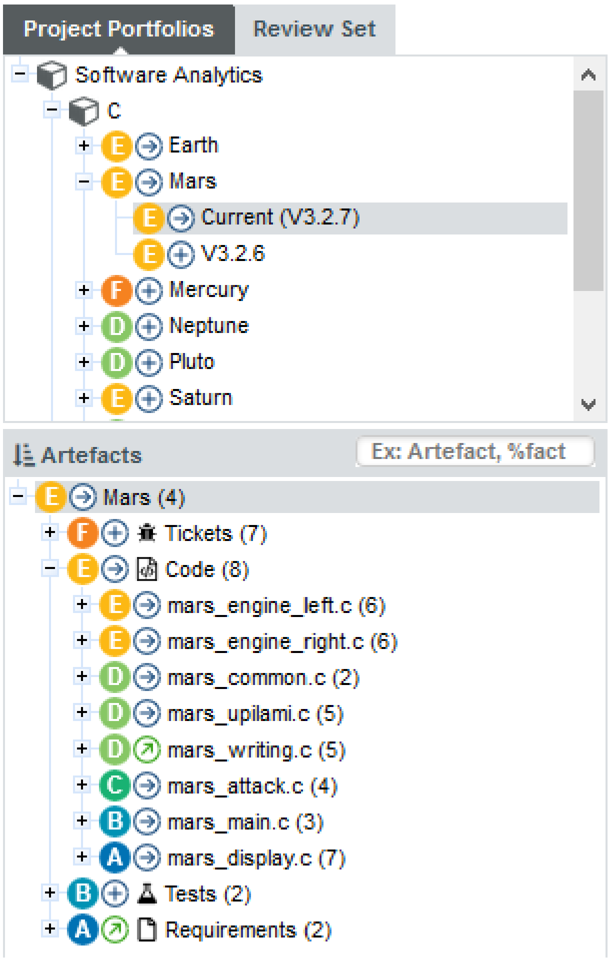
This example uses the Mars project from the samples folder. Ensure that you are a Project Manager in this project, or are part of a project role with the **View Drafts of Projects** and **Modify Artefacts** privileges before you begin.

The Mars is a sample project that includes ticket and test artefacts, in addition to the source code artefacts you have encountered up to this point. For more information about projects containing artefact types other than source code, refer to Going Beyond Source Code.

Expand the Project Portfolios to show all the versions of **Mars**. There are two versions in the tree (from bottom to top):

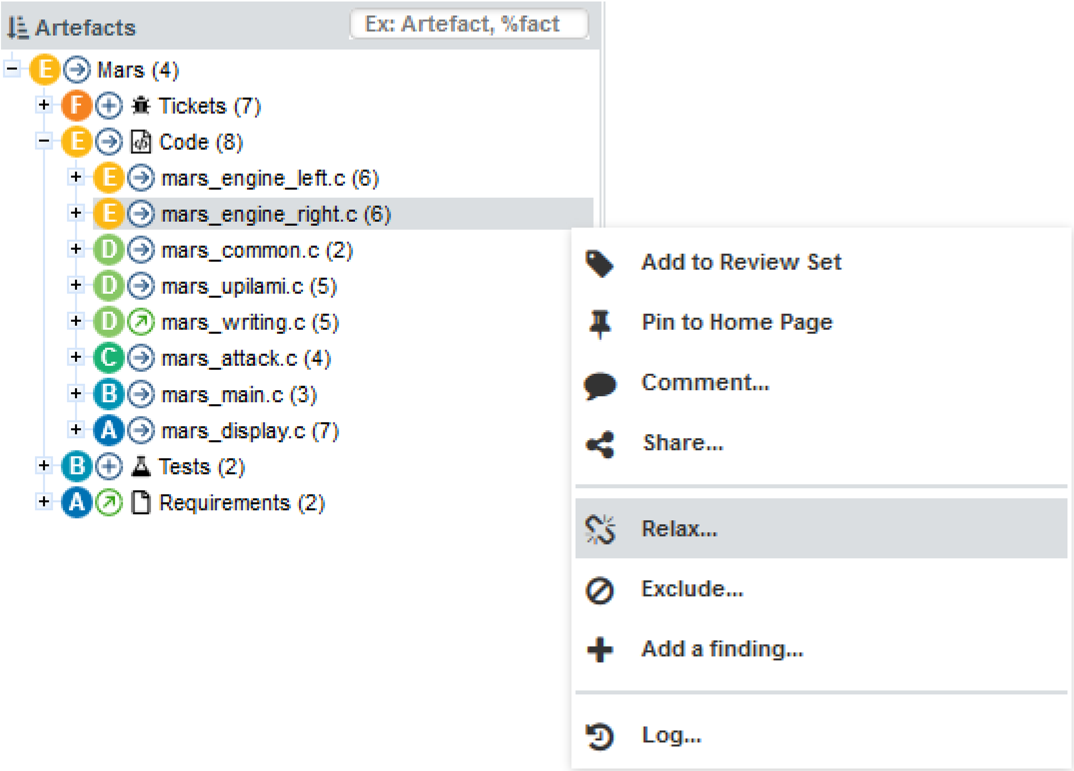
1. **V3.2.6** is the baseline version, whose results were computed during the analysis and cannot be changed.
2. **V3.2.7** is a version that was analysed as a draft so that you can edit form values, relax, exclude or add artefacts in preparation for the next analysis, as described in Working with Draft and Baseline Versions

Click on **Mars > V3.2.7** to see the artefacts in the Mars project as created by the demo script:



The artefacts in the V3.2.7 version of the Mars project and their rating

To relax an artefact and therefore tell Squore that its rating should not impact the rest of the project, display the context menu for this artefact. The relaxation options appear at the bottom of the menu if they are available for your model, as shown below:



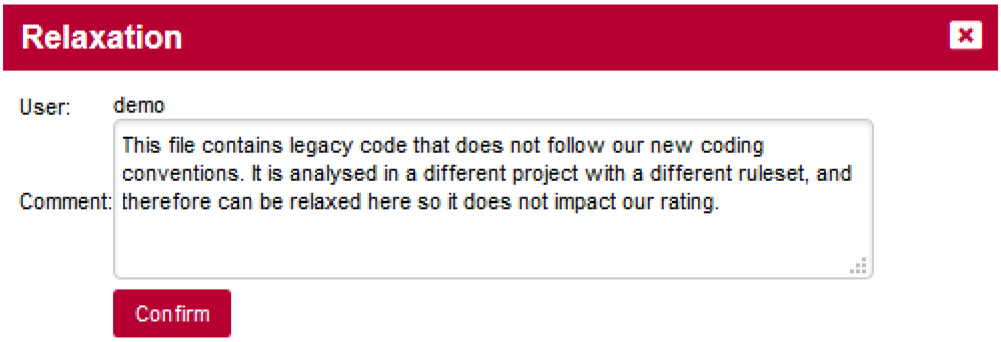
The artefact context menu

There are two actions that can be taken to relax an artefact:

- **Relax...** allows simply marking an artefact as relaxed, leaves it in the tree in a way that will not impact the overall rating of the project.
- **Exclude...** also relaxes the artefact but then removes it from the Artefact Tree so it will not be visible anymore in future analyses.

In both cases, the relaxation action is only made on a draft version and can be reversed by selecting the **Un-relax...** entry in the menu or the **Clear unapplied changes** option in the project portfolio.

Clicking **Relax...** or **Exclude...** brings up a pop-up menu where you can type a comment to explain the reason for the relaxation. Let's relax *mars_common.c* so it stops impacting the overall project rating. Click the **Relax...** option in the menu to display the relaxation popup and enter a relaxation comment:

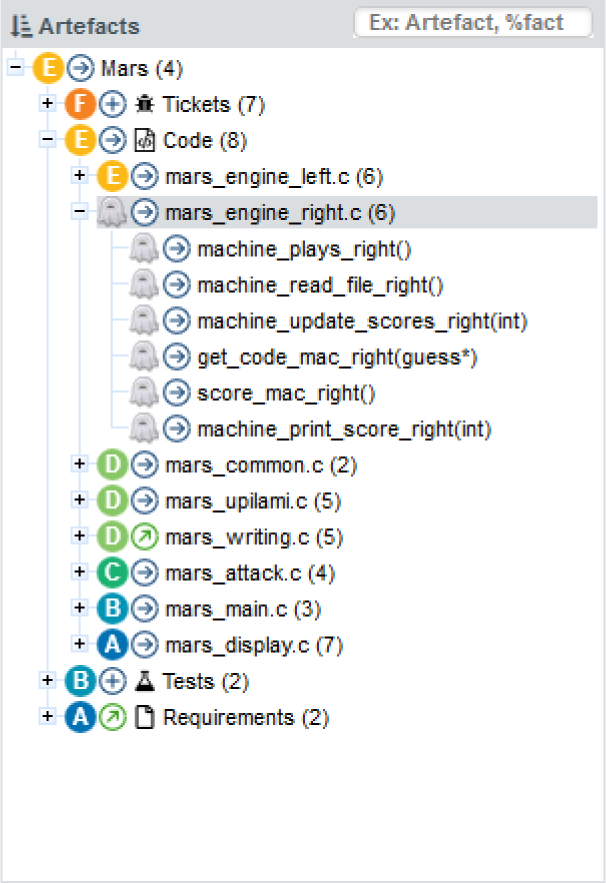


User: demo

Comment: This file contains legacy code that does not follow our new coding conventions. It is analysed in a different project with a different ruleset, and therefore can be relaxed here so it does not impact our rating.

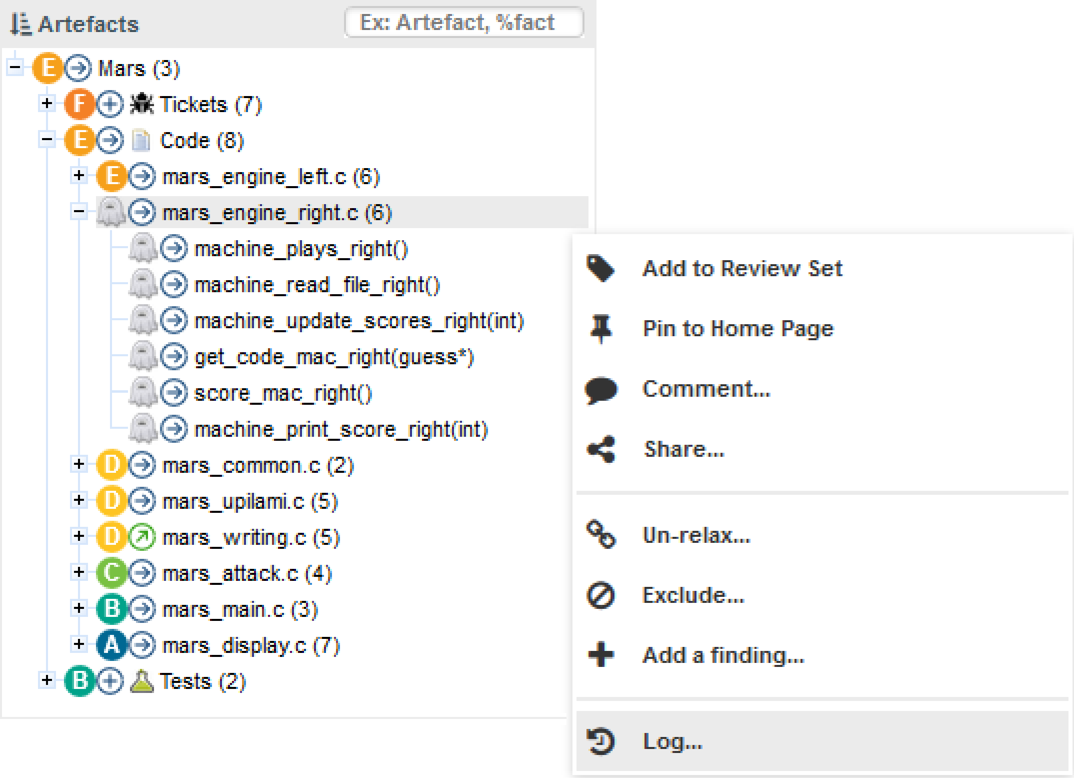
The relaxation justification

Click **Confirm** to save your comment, and notice how the Artefact Tree is updated to reflect the finding's status:



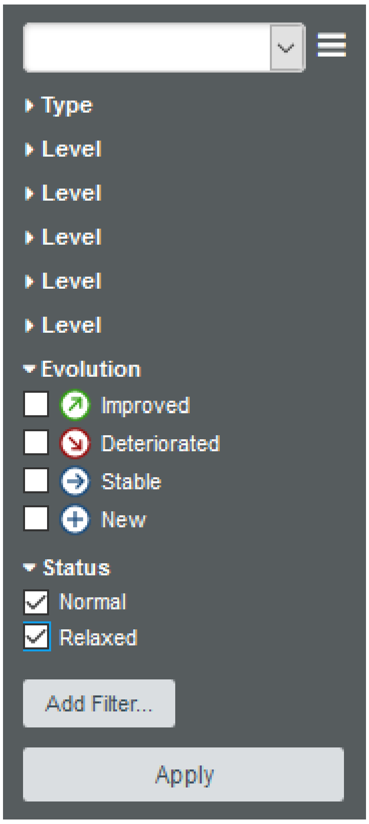
The relaxed mars_common.c in the Artefact Tree

Other users can review the justification for the relaxation by clicking on the Log... item in the artefact context menu:



The log of changes for the artefact mars_common.c

If you keep relaxing artefacts in this project and create a new draft build of the project, then you will end up seeing changes in the overall rating, When you relax an artefact, the action items and findings relevant to this artefact are hidden, except when you specifically click on the relaxed artefact. If you want to show them, you can create a filter that includes relaxed artefacts by checking the boxes with the appropriate status in the Filter Panel:



Type

Level

Level

Level

Level

Level

Evolution

□ Improved

□ Deteriorated

□ Stable

□ New

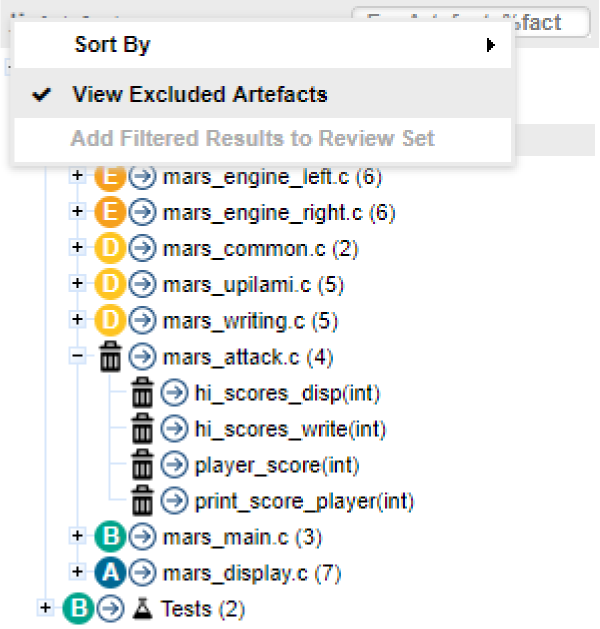
Status

☑ Normal

☑ Relaxed

The artefact statuses shown by default in the Artefact Tree

Excluded artefacts can be shown or hidden by clicking the View Excluded Artefacts menu option in the Artefact Tree



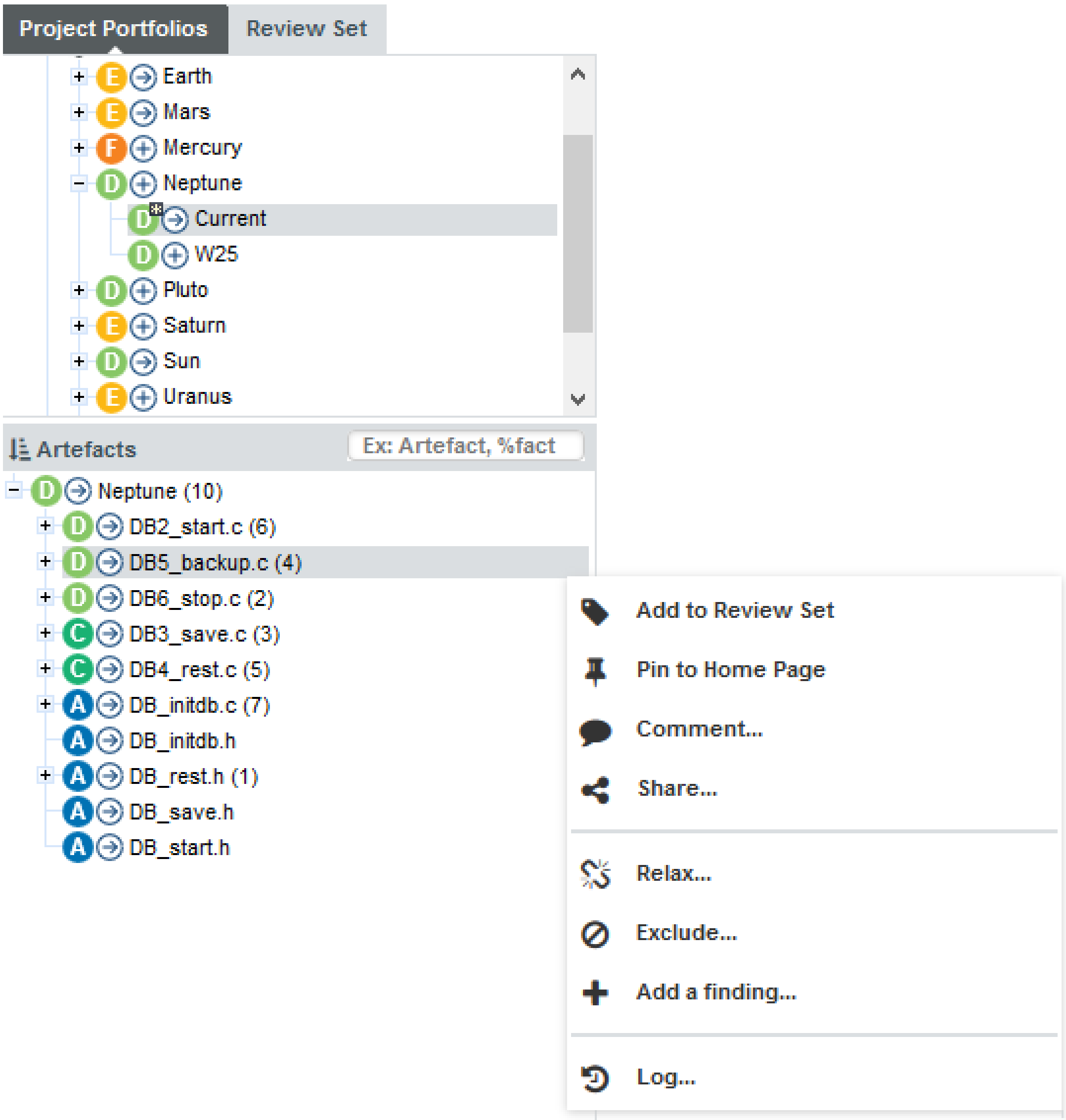
The View Excluded Artefacts menu option in the Artefact Tree

Adding Findings Manually

If you notice that a violation in the code or an issue in the project was not detected during an analysis, you can decide to create a finding manually from the Artefact Tree.

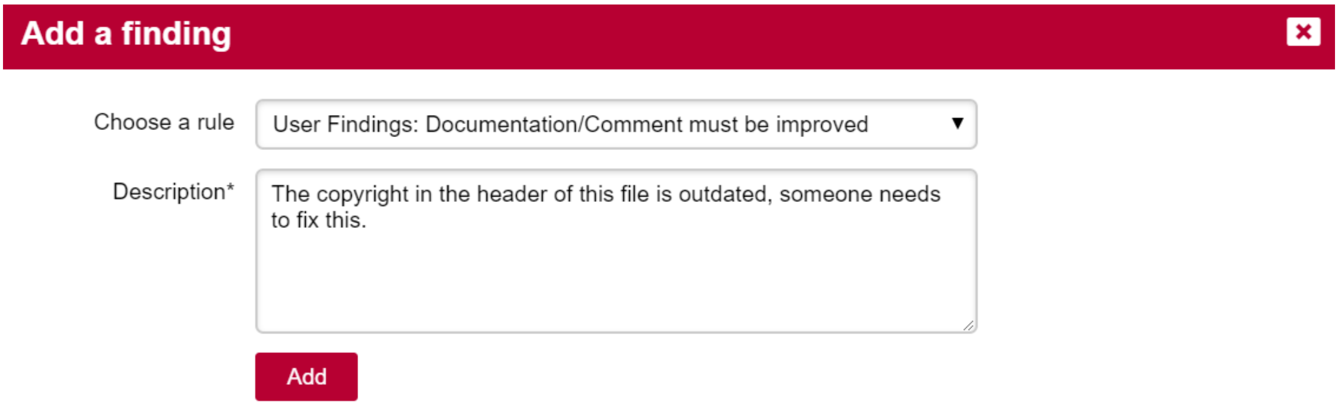
This feature, like the creation of manual artefacts (see Adding and Removing Artefacts Manually) is only available if your model was configured to support it. Consult your Squore administrator to verify if it is available in your configuration.

In this example, we add a finding to notify of a documentation issue in the Neptune project. Click on the Current version of the project, and display the context menu for the artefact where you consider that the documentation is wrong.



The artefact context menu with the ***Add a finding...*** *option highlighted*

When you click the **Add a finding...** option, a dialog appears and lets you select the type of finding to add, as well as a description of the issue:



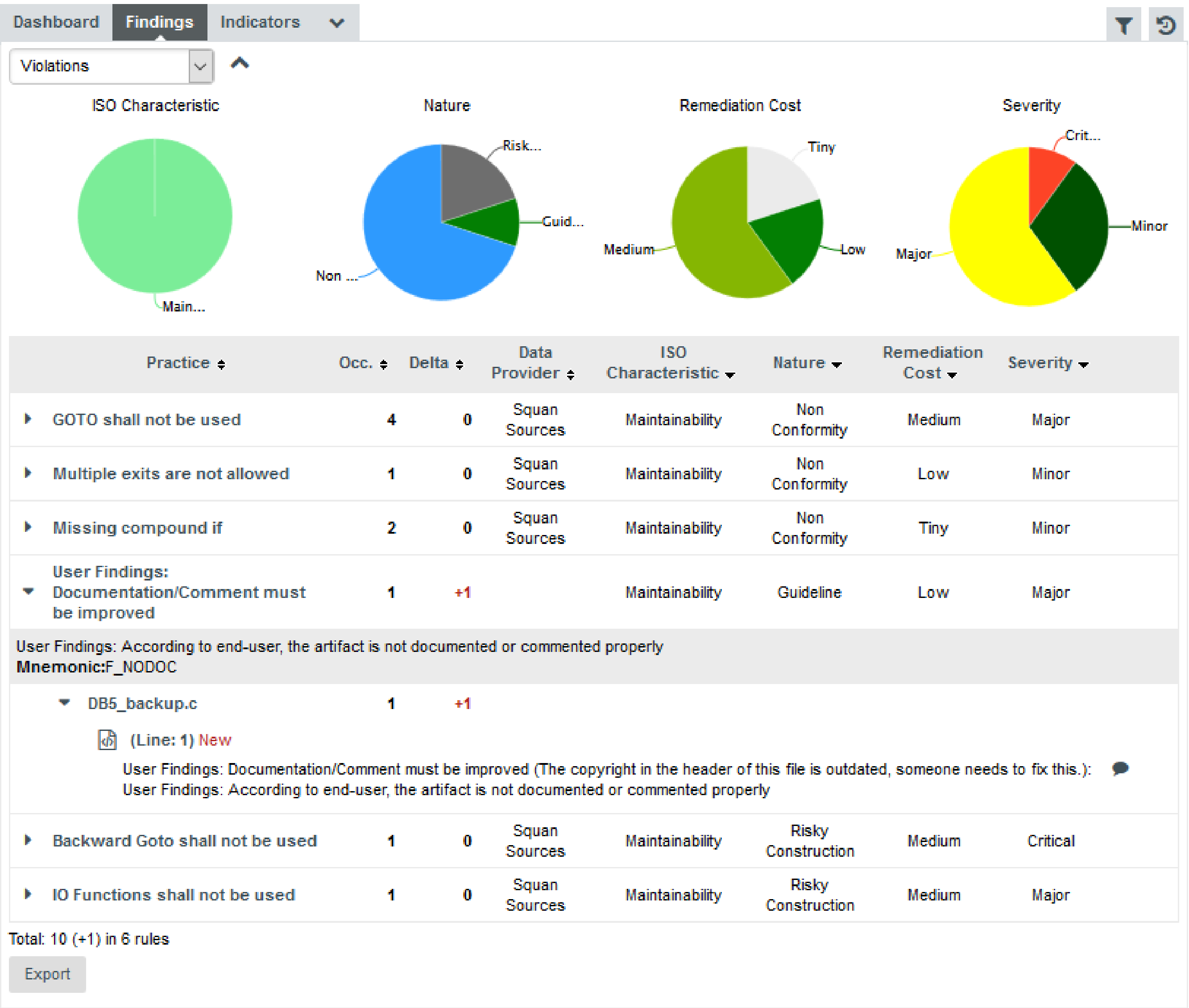
Add a finding

Choose a rule: User Findings: Documentation/Comment must be improved

Description*: The copyright in the header of this file is outdated, someone needs to fix this.

*The **Add a finding...** popup*

Click **Add** to save the finding. You can check that it was added successfully in the Findings tab of the Explorer:



	Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
▸	GOTO shall not be used	4	0	Squan Sources	Maintainability	Non Conformity	Medium	Major
▸	Multiple exits are not allowed	1	0	Squan Sources	Maintainability	Non Conformity	Low	Minor
▸	Missing compound if	2	0	Squan Sources	Maintainability	Non Conformity	Tiny	Minor
▾	User Findings: Documentation/Comment must be improved	1	+1		Maintainability	Guideline	Low	Major

User Findings: According to end-user, the artifact is not documented or commented properly
Mnemonic:F_NODOC

▾ DB5_backup.c 1 +1

(Line: 1) New

User Findings: Documentation/Comment must be improved (The copyright in the header of this file is outdated, someone needs to fix this.):
User Findings: According to end-user, the artifact is not documented or commented properly

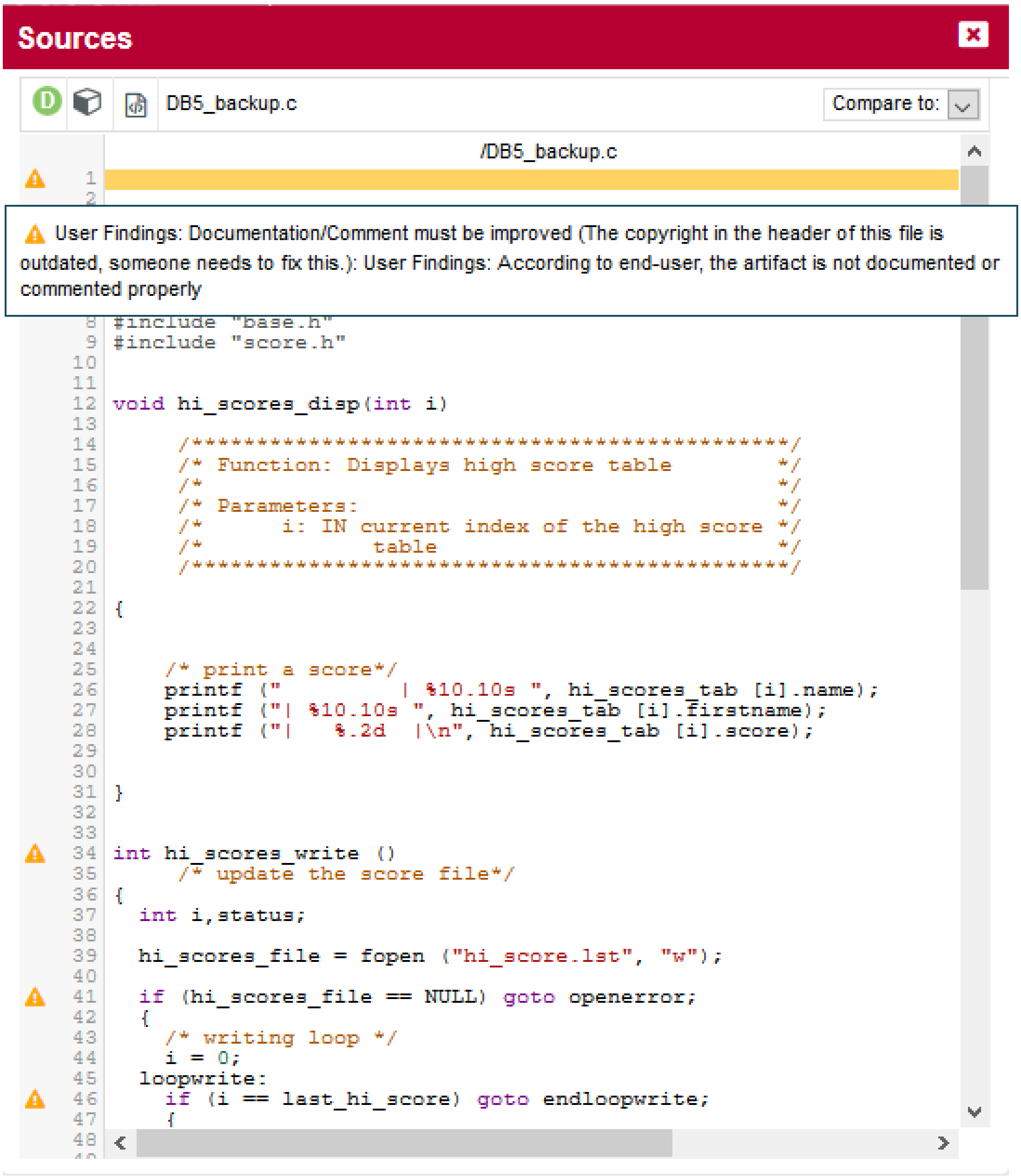
	Practice	Occ.	Delta	Data Provider	ISO Characteristic	Nature	Remediation Cost	Severity
▸	Backward Goto shall not be used	1	0	Squan Sources	Maintainability	Risky Construction	Medium	Critical
▸	IO Functions shall not be used	1	0	Squan Sources	Maintainability	Risky Construction	Medium	Major

Total: 10 (+1) in 6 rules

The Findings tab showing the manually added finding

Manual findings are displayed automatically in the Findings tab like other findings. If you want to filter them, use the advanced filter and select or exclude **[Manual]** in the Data Provider category.

Like regular findings, your finding also displays in the source code viewer, as shown below:

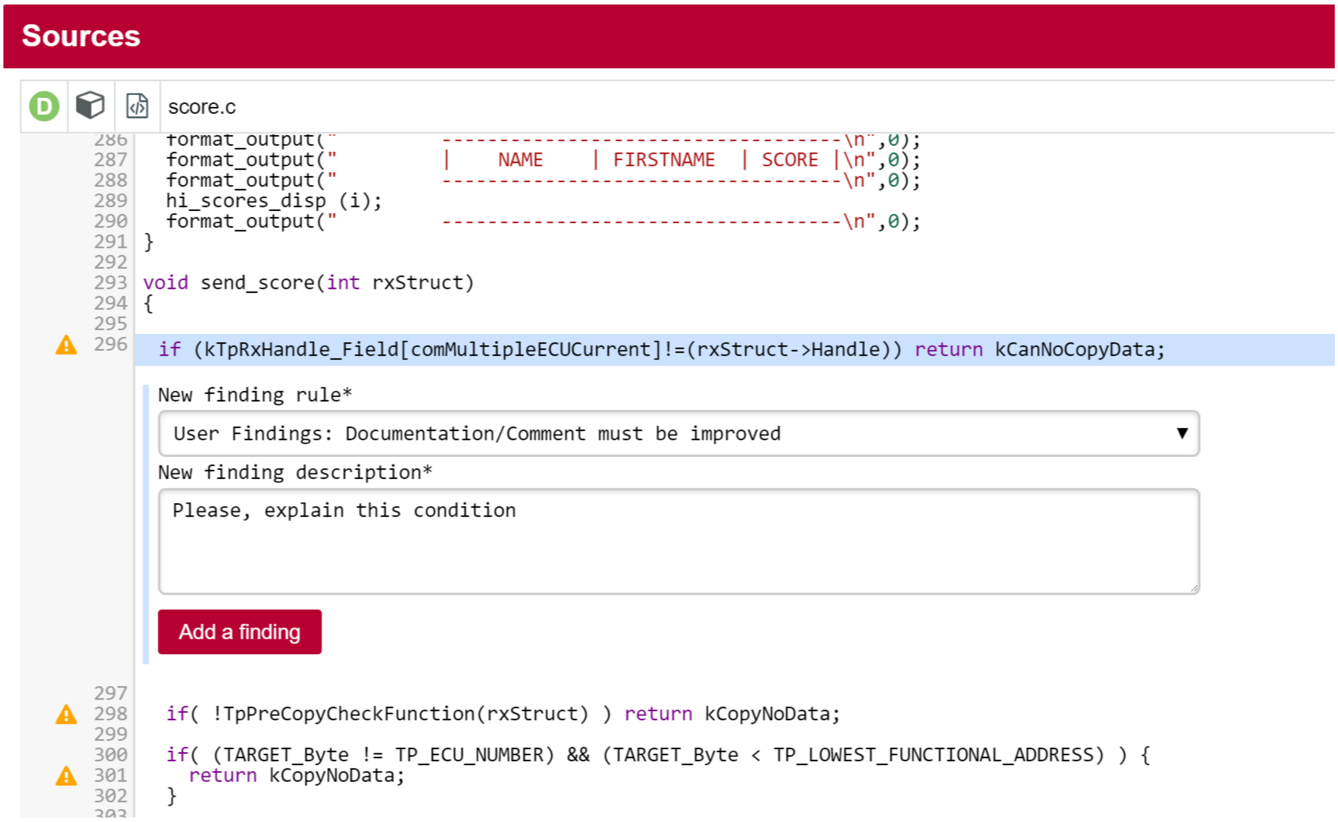


User Findings: Documentation/Comment must be improved (The copyright in the header of this file is outdated, someone needs to fix this.): User Findings: According to end-user, the artifact is not documented or commented properly

```
 1
 2
 8 #include "base.h"
 9 #include "score.h"
10
11
12 void hi_scores_disp(int i)
13
14      /************************************************/
15      /* Function: Displays high score table          */
16      /*                                              */
17      /* Parameters:                                  */
18      /*       i: IN current index of the high score */
19      /*              table                           */
20      /************************************************/
21
22 {
23
24
25     /* print a score*/
26     printf ("         | %10.10s ", hi_scores_tab [i].name);
27     printf ("| %10.10s ", hi_scores_tab [i].firstname);
28     printf ("|   %.2d  |\n", hi_scores_tab [i].score);
29
30
31 }
32
33
34 int hi_scores_write ()
35      /* update the score file*/
36 {
37   int i,status;
38
39   hi_scores_file = fopen ("hi_score.lst", "w");
40
41   if (hi_scores_file == NULL) goto openerror;
42   {
43     /* writing loop */
44     i = 0;
45   loopwrite:
46     if (i == last_hi_score) goto endloopwrite;
47     {
48
```

The documentation issue is visible on line 1 of the file it was added to

It is also possible to add a manual finding directly from the source code view. To do so, click on the code line number where you want to add the manual finding. The finding adding form is displayed. Firstly, you have to select the finding rule among the manual rules, then you have to fill the new finding description. Both fields are required. Finally, confirm the new manual finding addition thanks to the "Add a finding" button.



```
  format_output("          -----------------------------------\n",0);
  format_output("          |    NAME    | FIRSTNAME  | SCORE |\n",0);
  format_output("          -----------------------------------\n",0);
  hi_scores_disp (i);
  format_output("          -----------------------------------\n",0);
}

void send_score(int rxStruct)
{

 if (kTpRxHandle_Field[comMultipleECUCurrent]!=(rxStruct->Handle)) return kCanNoCopyData;
```

New finding rule*

User Findings: Documentation/Comment must be improved ▼

New finding description*

Please, explain this condition

```

  if( !TpPreCopyCheckFunction(rxStruct) ) return kCopyNoData;

  if( (TARGET_Byte != TP_ECU_NUMBER) && (TARGET_Byte < TP_LOWEST_FUNCTIONAL_ADDRESS) ) {
    return kCopyNoData;
  }
```

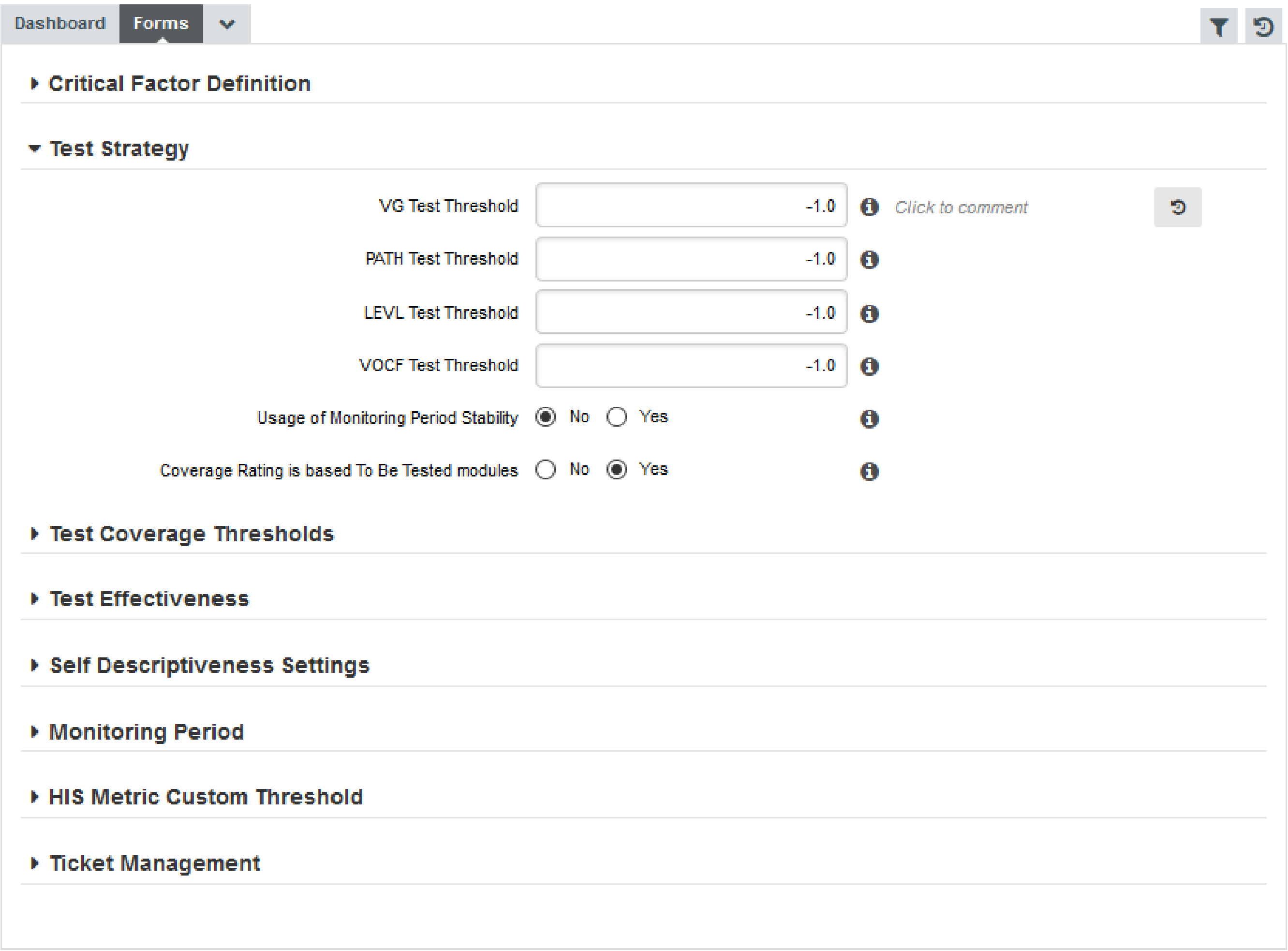
Adding a finding from the source code view

Working with Forms and Checklists

Squore lets you view and edit project attributes in a dedicated form tab of the explorer. You can therefore design your wizards to present checklists to a user. They can fill in the values manually after an analysis and they will be taken into account when creating the next version of the project. There are permissions associated with editing form values, so you can make them read-only for guests but read-write for project managers. The attributes displayed on the Forms tab depend on the type of the current artefact, and values are saved individually for each artefact in the project.

To begin working with forms, make sure you select the Current version of the project in the tree and that the **Forms** tab is visible in the Explorer.

When you click a project in the Project Portfolios and view the Forms tab of the Explorer, all the project attributes available at application level are displayed, as shown below:



▸ Critical Factor Definition

▾ Test Strategy

VG Test Threshold	-1.0
PATH Test Threshold	-1.0
LEVL Test Threshold	-1.0
VOCF Test Threshold	-1.0
Usage of Monitoring Period Stability	● No ○ Yes
Coverage Rating is based To Be Tested modules	○ No ● Yes

▸ Test Coverage Thresholds

▸ Test Effectiveness

▸ Self Descriptiveness Settings

▸ Monitoring Period

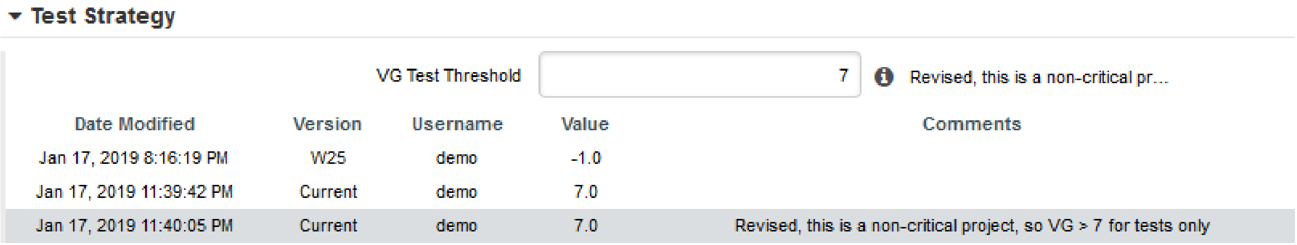
▸ HIS Metric Custom Threshold

▸ Ticket Management

The Forms tab for the Neptune project at application level

The values displayed correspond to the application attributes passed when the last version of Neptune was created. Users whose project role grants them the Modify Artefacts Attributes privilege can edit the current value of the form for any artefact, and the value will be taken into account during the next analysis.

When you modify the values in the form, you can use the comment field to justify the change you made. A history of the modifications can be displayed by expanding the attribute field, as shown below



▾ Test Strategy

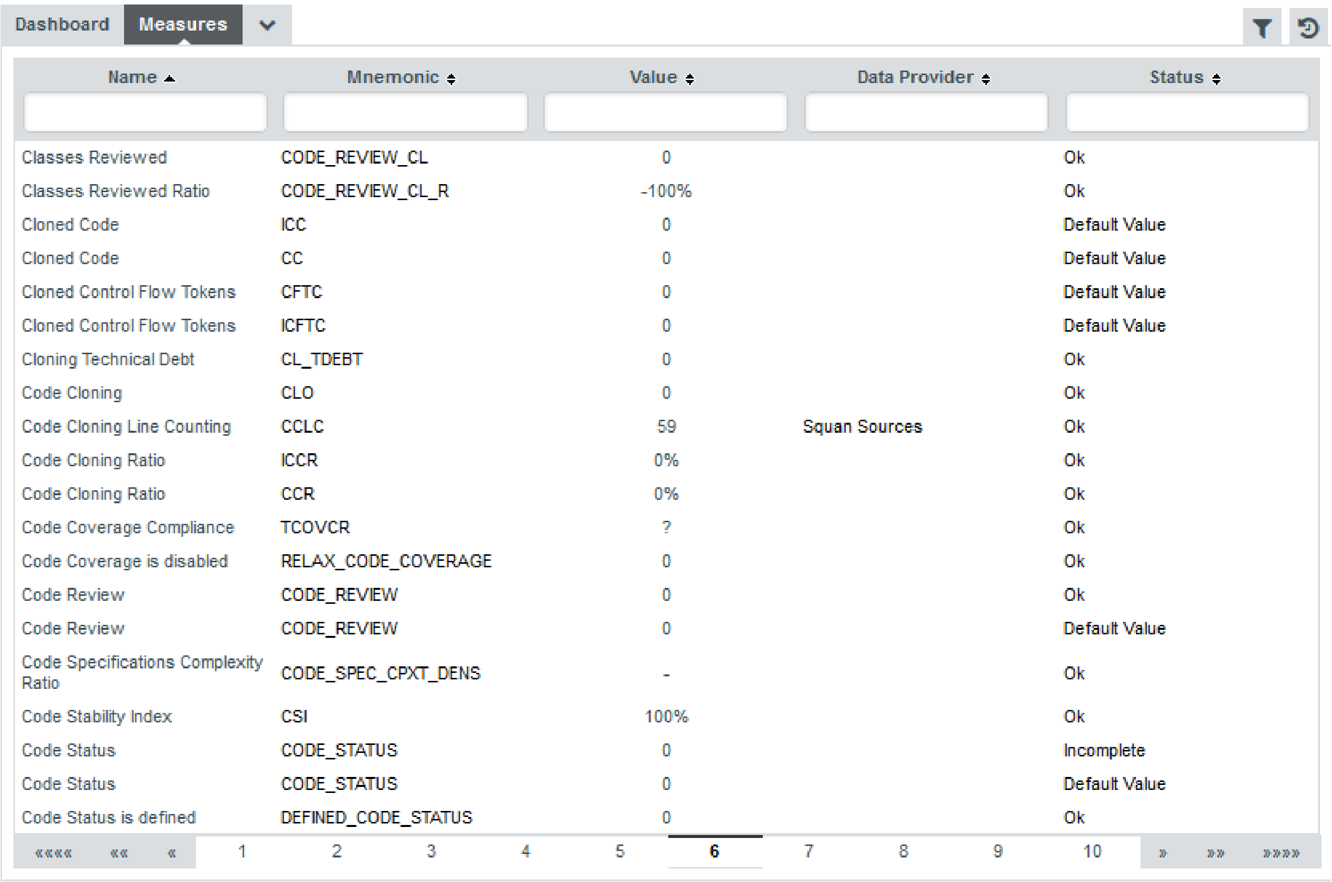
VG Test Threshold: 7 — Revised, this is a non-critical pr...

Date Modified	Version	Username	Value	Comments
Jan 17, 2019 8:16:19 PM	W25	demo	-1.0	
Jan 17, 2019 11:39:42 PM	Current	demo	7.0	
Jan 17, 2019 11:40:05 PM	Current	demo	7.0	Revised, this is a non-critical project, so VG > 7 for tests only

A history of modifications for the Test Coverage attributes

What Does This Measure Mean Exactly?

If you have doubts about the measures computed by Squore and their meaning, they can usually be solved by looking at the **Measures** tab of the Explorer. The content of the measures tab is also always refreshed to reflect the data for the current artefact, and is organised in a table displaying the measure's mnemonic, full name, description and value for the current selection, as shown in the picture below.



Name	Mnemonic	Value	Data Provider	Status
Classes Reviewed	CODE_REVIEW_CL	0		Ok
Classes Reviewed Ratio	CODE_REVIEW_CL_R	-100%		Ok
Cloned Code	ICC	0		Default Value
Cloned Code	CC	0		Default Value
Cloned Control Flow Tokens	CFTC	0		Default Value
Cloned Control Flow Tokens	ICFTC	0		Default Value
Cloning Technical Debt	CL_TDEBT	0		Ok
Code Cloning	CLO	0		Ok
Code Cloning Line Counting	CCLC	59	Squan Sources	Ok
Code Cloning Ratio	ICCR	0%		Ok
Code Cloning Ratio	CCR	0%		Ok
Code Coverage Compliance	TCOVCR	?		Ok
Code Coverage is disabled	RELAX_CODE_COVERAGE	0		Ok
Code Review	CODE_REVIEW	0		Ok
Code Review	CODE_REVIEW	0		Default Value
Code Specifications Complexity Ratio	CODE_SPEC_CPXT_DENS	-		Ok
Code Stability Index	CSI	100%		Ok
Code Status	CODE_STATUS	0		Incomplete
Code Status	CODE_STATUS	0		Default Value
Code Status is defined	DEFINED_CODE_STATUS	0		Ok

The table of measures for the DB5_backup.c

Measures can be sorted by mnemonic, name, description or value, and the sorting value is remembered when selecting another artefact in the tree so you can easily compare values.

The table also tells you which Data Provider reported the metric and its status in the latest analysis, so you can determine if a metric was computed or has its default value from the analysis model. The possible status values are:

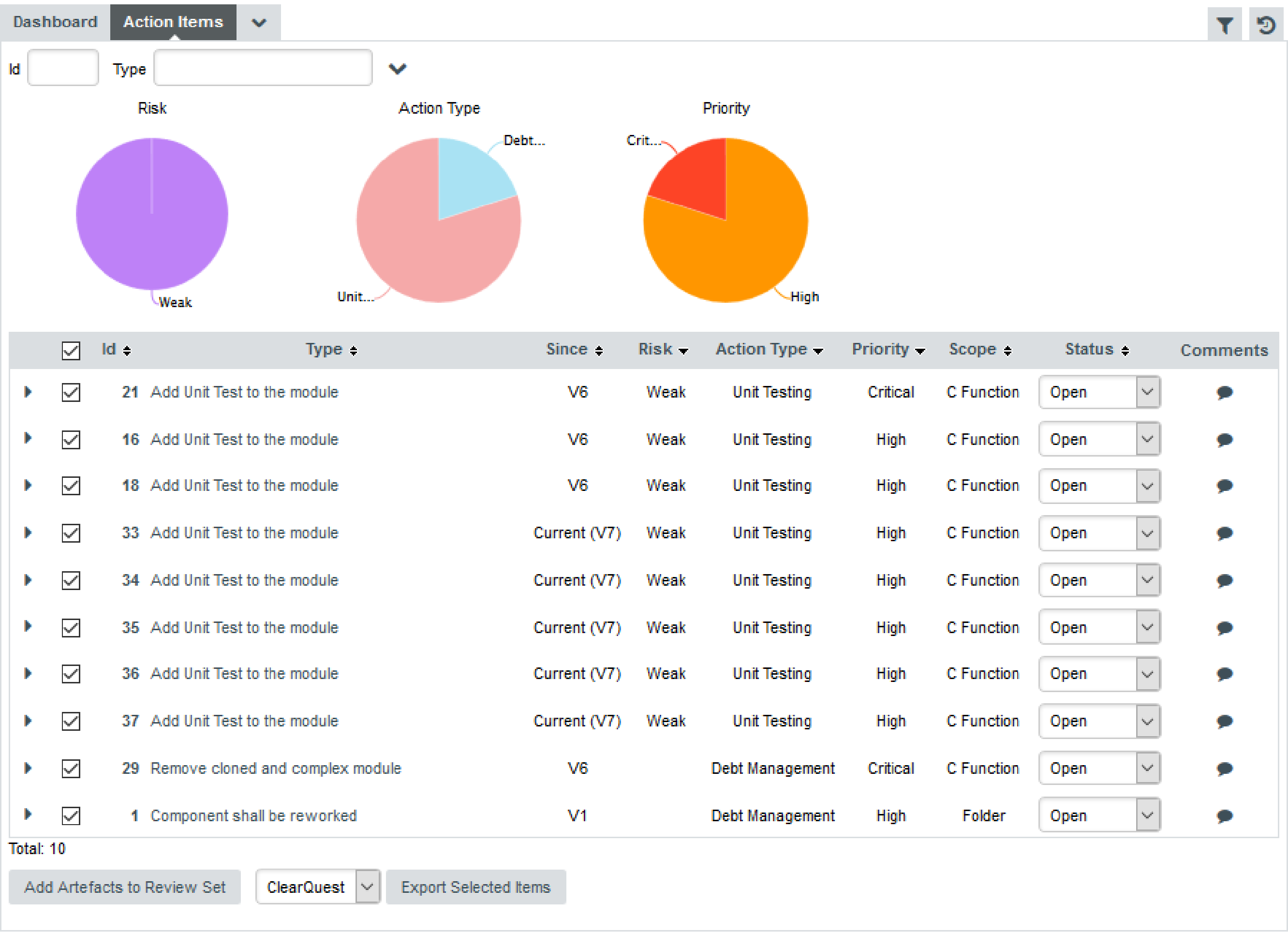
- **Default Value:** This measure has the default value defined in the analysis model
- **Ok:** A value was computed successfully for this measure
- **User-defined:** The value was set by the user (either via a tag on the command line or in the Forms tab of the web UI)
- **Definition error:** The value could be computed because of an error in the analysis model. Check the Model Validator to learn more.
- **Incomplete:** The value could not be computed because of an error (maybe a division by zero?). The analysis model should probably be updated to avoid this in the future. This error is also available in the project's build.log.
- **Warning:** The value could not be computed, but there is nothing wrong with the measure definition in the analysis model. Maybe you are trying to do a COUNT on descendants but there are no descendants? In such cases, the error is not serious, but you can improve your analysis model to handle the warning if needed.
- **-:** This measure was not found in the project. It did not exist at the time of the analysis.
- **Unknown:** An unexpected error happened while computing the measure's status

In all error statuses above, the metric is assigned the default value defined in the analysis model.

How Do I Review And Manage Action Items Flagged by Squore?

Searching for issues in your applications can be a manual process, as explained in How do I understand and Improve My Ratings?, but the analysis and decision models configured within Squore can automate this process by automatically suggesting items that require your attention after analysing the latest version of your code. This functionality can be accessed as part of the Explorer, in the **Action Items** tab. In this section, you will learn how to review Squore's suggestions and incorporate them into your own issue management tool.

Note that in order to change the status of the action item, you must be working with the current draft version of a project. In order to follow the steps below, ensure that you select the current version of the Earth project, click on the Action Items tab. A list of action items suggested by Squore appears in a table, as shown in the picture below:



	☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
▸	☑	21	Add Unit Test to the module	V6	Weak	Unit Testing	Critical	C Function	Open	💬
▸	☑	16	Add Unit Test to the module	V6	Weak	Unit Testing	High	C Function	Open	💬
▸	☑	18	Add Unit Test to the module	V6	Weak	Unit Testing	High	C Function	Open	💬
▸	☑	33	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▸	☑	34	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▸	☑	35	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▸	☑	36	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▸	☑	37	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▸	☑	29	Remove cloned and complex module	V6		Debt Management	Critical	C Function	Open	💬
▸	☑	1	Component shall be reworked	V1		Debt Management	High	Folder	Open	💬

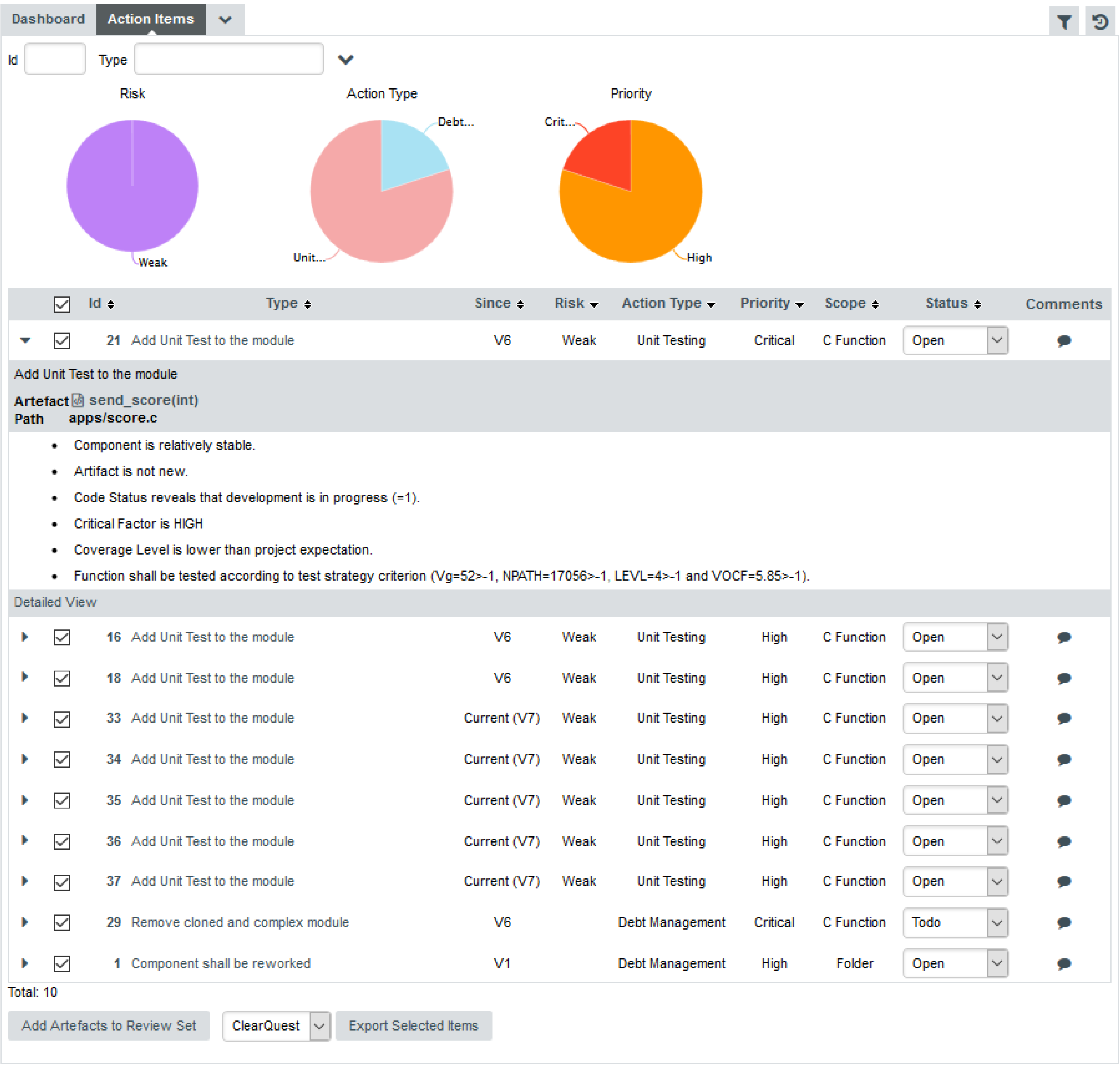
Total: 10

The action items table for the current version of the Earth project

You can filter action items if needed by using the filters above the table. The name given by Squore to the action item is the name defined in your analysis model for this alert. Priorities are also predefined, and your input is needed to validate or invalidate the reports based on your priorities.

In the action items list, *29* is critical, therefore its status should be changed to **Todo**.

If you are unsure about a report, you can click the action item ID to display the full details, which includes the location(s) in the source code that triggered the alert:



	☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
▼	☑	21	Add Unit Test to the module	V6	Weak	Unit Testing	Critical	C Function	Open	💬

Add Unit Test to the module

Artefact send_score(int)
Path apps/score.c

- Component is relatively stable.
- Artifact is not new.
- Code Status reveals that development is in progress (=1).
- Critical Factor is HIGH
- Coverage Level is lower than project expectation.
- Function shall be tested according to test strategy criterion (Vg=52>-1, NPATH=17056>-1, LEVL=4>-1 and VOCF=5.85>-1).

Detailed View

	☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
▶	☑	16	Add Unit Test to the module	V6	Weak	Unit Testing	High	C Function	Open	💬
▶	☑	18	Add Unit Test to the module	V6	Weak	Unit Testing	High	C Function	Open	💬
▶	☑	33	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▶	☑	34	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▶	☑	35	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▶	☑	36	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▶	☑	37	Add Unit Test to the module	Current (V7)	Weak	Unit Testing	High	C Function	Open	💬
▶	☑	29	Remove cloned and complex module	V6		Debt Management	Critical	C Function	Todo	💬
▶	☑	1	Component shall be reworked	V1		Debt Management	High	Folder	Open	💬

Total: 10

Action Item details for 21

You can review the code in a popup window before you decide to fix or relax the action item.

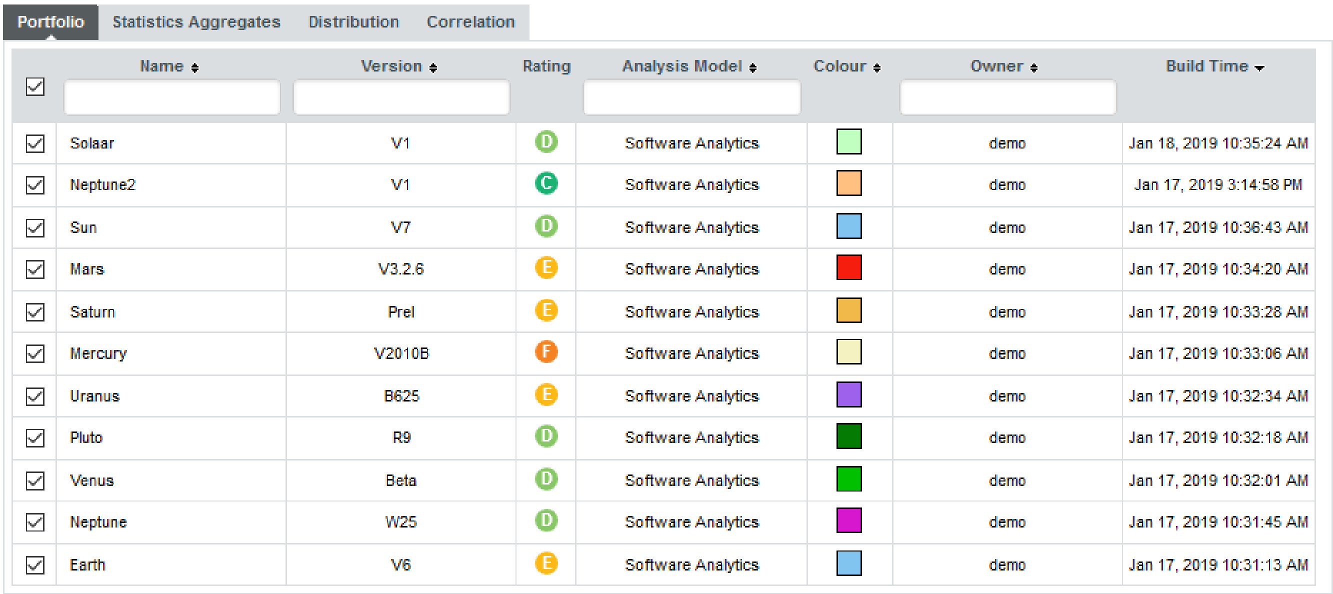
Finally, you can export the action items generated by Squore and feed them into your own issue tracker: Select the export format you want to use (CSV, ClearQuest, Mantis, XML, or any other custom format you defined in your configuration) and click the Export button to download the list to a file. You can also add all artefacts that triggered an action item to your Review Set by clicking the appropriate button.

If the Export button is greyed out, your licence does not include the option to export data to CSV files.

If you are looking for a way to present action items instead of exporting them, you should look into Squore reporting functionality, described in Reporting Project Status.

Can I Perform Advanced Data Mining?

The Capitalisation base provides statistics aggregates, distribution graphics and correlation coefficients across a portfolio of projects. To begin using the Capitalisation to understand historical trends about your projects and find out if your analysis models are suited to your development style, click the **Capitalisation** menu item in the Squore toolbar.

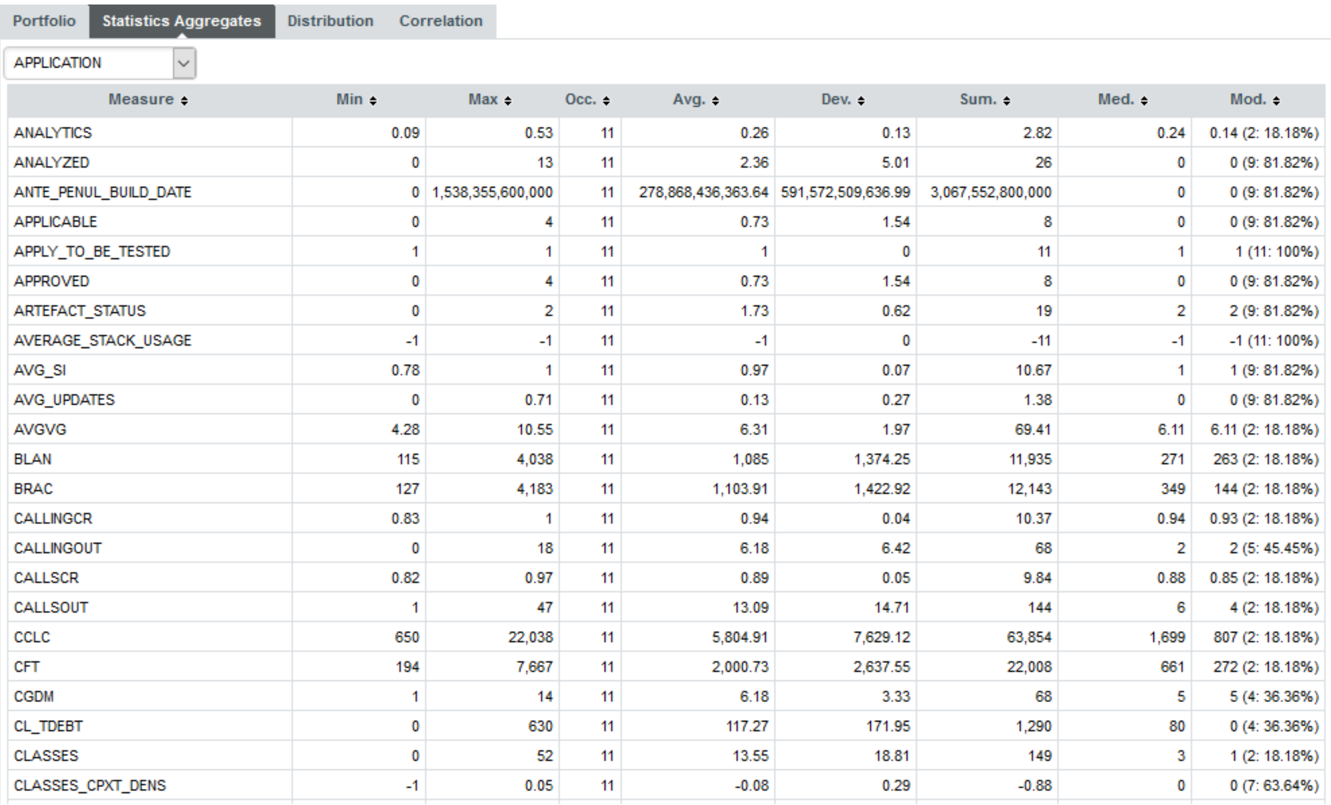


☑	Name ⬍	Version ⬍	Rating	Analysis Model ⬍	Colour ⬍	Owner ⬍	Build Time ▾
☑	Solaar	V1	D	Software Analytics	🟩	demo	Jan 18, 2019 10:35:24 AM
☑	Neptune2	V1	C	Software Analytics	🟧	demo	Jan 17, 2019 3:14:58 PM
☑	Sun	V7	D	Software Analytics	🟦	demo	Jan 17, 2019 10:36:43 AM
☑	Mars	V3.2.6	E	Software Analytics	🟥	demo	Jan 17, 2019 10:34:20 AM
☑	Saturn	Prel	E	Software Analytics	🟧	demo	Jan 17, 2019 10:33:28 AM
☑	Mercury	V2010B	F	Software Analytics	🟨	demo	Jan 17, 2019 10:33:06 AM
☑	Uranus	B625	E	Software Analytics	🟪	demo	Jan 17, 2019 10:32:34 AM
☑	Pluto	R9	D	Software Analytics	🟩	demo	Jan 17, 2019 10:32:18 AM
☑	Venus	Beta	D	Software Analytics	🟩	demo	Jan 17, 2019 10:32:01 AM
☑	Neptune	W25	D	Software Analytics	🟪	demo	Jan 17, 2019 10:31:45 AM
☑	Earth	V6	E	Software Analytics	🟦	demo	Jan 17, 2019 10:31:13 AM

The Capitalisation Projects tab

In the projects tab, choose the projects that will be used to aggregate statistics. In the example below, we will look at statistics for the Earth and Mars projects, which both use the same analysis model and have similar overall ratings. Select Earth and Mars from the list and click the **Statistics Aggregates**.

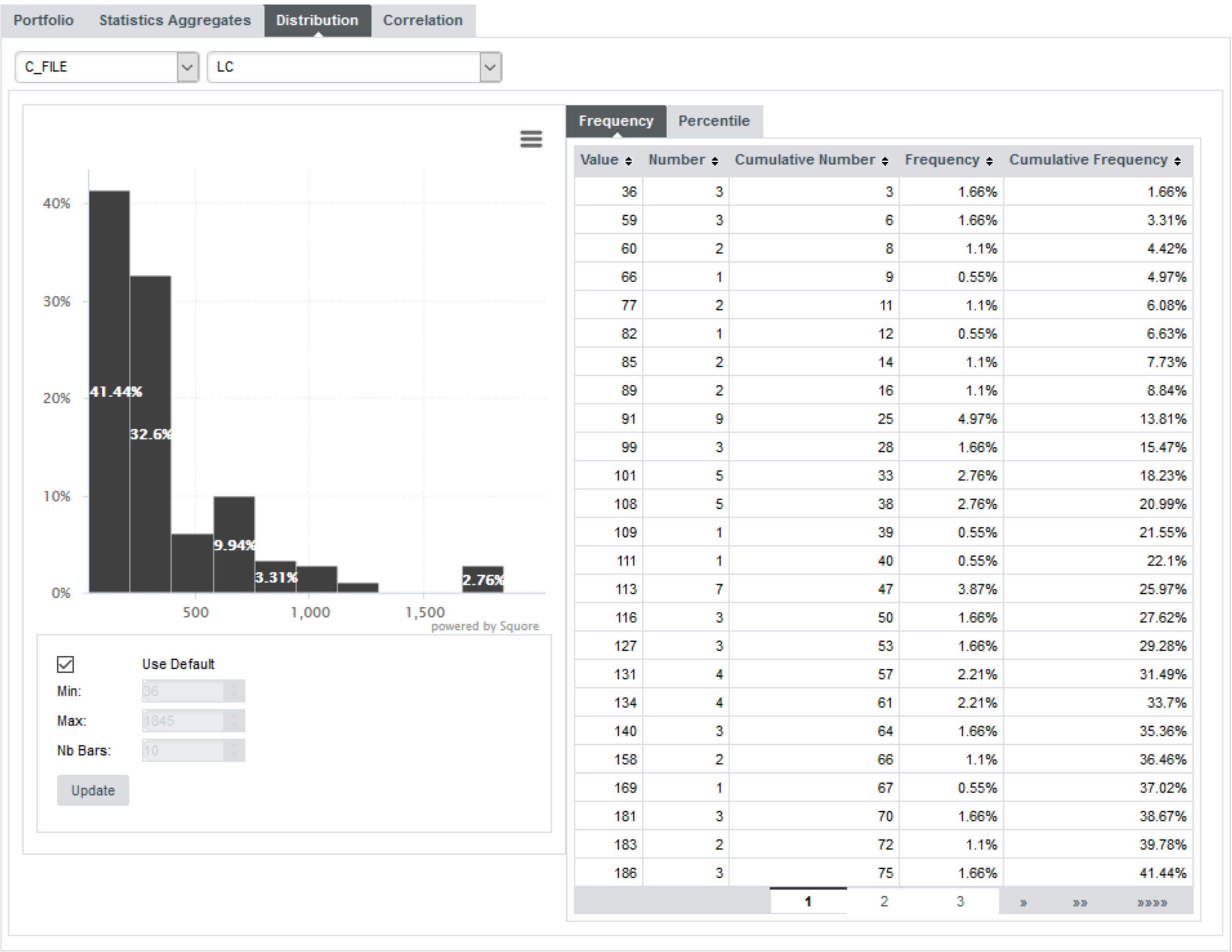
The Statistics Aggregates tab offers an overview of all your projects' data by providing minimum, maximum, average, number of occurrences, deviation, mod, sum and median results. Results are based on all the measures of an artefact type. This means that you have to specify an artefact type before any data is displayed.



Measure ⬍	Min ⬍	Max ⬍	Occ. ⬍	Avg. ⬍	Dev. ⬍	Sum. ⬍	Med. ⬍	Mod. ⬍
ANALYTICS	0.09	0.53	11	0.26	0.13	2.82	0.24	0.14 (2: 18.18%)
ANALYZED	0	13	11	2.36	5.01	26	0	0 (9: 81.82%)
ANTE_PENUL_BUILD_DATE	0	1,538,355,600,000	11	278,868,436,363.64	591,572,509,636.99	3,067,552,800,000	0	0 (9: 81.82%)
APPLICABLE	0	4	11	0.73	1.54	8	0	0 (9: 81.82%)
APPLY_TO_BE_TESTED	1	1	11	1	0	11	1	1 (11: 100%)
APPROVED	0	4	11	0.73	1.54	8	0	0 (9: 81.82%)
ARTEFACT_STATUS	0	2	11	1.73	0.62	19	2	2 (9: 81.82%)
AVERAGE_STACK_USAGE	-1	-1	11	-1	0	-11	-1	-1 (11: 100%)
AVG_SI	0.78	1	11	0.97	0.07	10.67	1	1 (9: 81.82%)
AVG_UPDATES	0	0.71	11	0.13	0.27	1.38	0	0 (9: 81.82%)
AVGVG	4.28	10.55	11	6.31	1.97	69.41	6.11	6.11 (2: 18.18%)
BLAN	115	4,038	11	1,085	1,374.25	11,935	271	263 (2: 18.18%)
BRAC	127	4,183	11	1,103.91	1,422.92	12,143	349	144 (2: 18.18%)
CALLINGCR	0.83	1	11	0.94	0.04	10.37	0.94	0.93 (2: 18.18%)
CALLINGOUT	0	18	11	6.18	6.42	68	2	2 (5: 45.45%)
CALLSCR	0.82	0.97	11	0.89	0.05	9.84	0.88	0.85 (2: 18.18%)
CALLSOUT	1	47	11	13.09	14.71	144	6	4 (2: 18.18%)
CCLC	650	22,038	11	5,804.91	7,629.12	63,854	1,699	807 (2: 18.18%)
CFT	194	7,667	11	2,000.73	2,637.55	22,008	661	272 (2: 18.18%)
CGDM	1	14	11	6.18	3.33	68	5	5 (4: 36.36%)
CL_TDEBT	0	630	11	117.27	171.95	1,290	80	0 (4: 36.36%)
CLASSES	0	52	11	13.55	18.81	149	3	1 (2: 18.18%)
CLASSES_CPXT_DENS	-1	0.05	11	-0.08	0.29	-0.88	0	0 (7: 63.64%)

The Capitalisation Base Statistics Aggregates at Application level

The Distribution tab offers the possibility to display any kind of distribution. The distribution is based on a measure of an artefact type. As a result, you have to select both an **Artefact type** and a **Measure** before you see any results. Note that you can change the parameters of the distribution graph by adjusting the number of bars, and the minimum and maximum values for the axes. The picture below shows the distribution of lines of code (a measure called *LC*) across all artefacts of type *FILE* for Mars and Earth.



☑ Use Default

Min: 36

Max: 1845

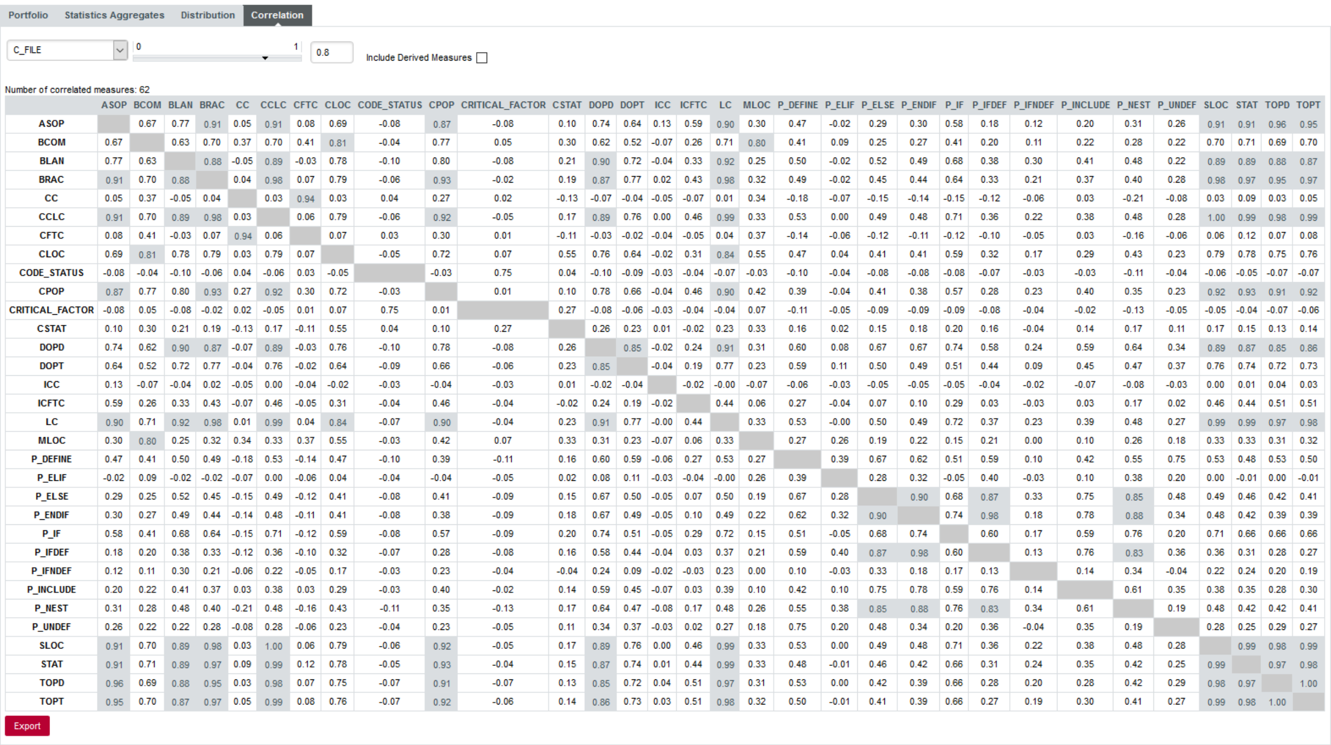
Nb Bars: 10

Update

Value	Number	Cumulative Number	Frequency	Cumulative Frequency
36	3	3	1.66%	1.66%
59	3	6	1.66%	3.31%
60	2	8	1.1%	4.42%
66	1	9	0.55%	4.97%
77	2	11	1.1%	6.08%
82	1	12	0.55%	6.63%
85	2	14	1.1%	7.73%
89	2	16	1.1%	8.84%
91	9	25	4.97%	13.81%
99	3	28	1.66%	15.47%
101	5	33	2.76%	18.23%
108	5	38	2.76%	20.99%
109	1	39	0.55%	21.55%
111	1	40	0.55%	22.1%
113	7	47	3.87%	25.97%
116	3	50	1.66%	27.62%
127	3	53	1.66%	29.28%
131	4	57	2.21%	31.49%
134	4	61	2.21%	33.7%
140	3	64	1.66%	35.36%
158	2	66	1.1%	36.46%
169	1	67	0.55%	37.02%
181	3	70	1.66%	38.67%
183	2	72	1.1%	39.78%
186	3	75	1.66%	41.44%

The Capitalisation Base Distribution Graph for lines of code per file

The Correlation tab displays the matrix of correlation of any data stored in the Squore database. Correlations are computed between artefacts of the same type, so you have to select the artefact type before any data is displayed. Squore highlights cells in the table in which correlations are above the threshold defined by moving the slider or entering a correlation coefficient directly in the text box provided. You can choose to include derived data by checking the box above the matrix table.



C_FILE 0 1 0.8 Include Derived Measures □

Number of correlated measures: 62

	ASOP	BCOM	BLAN	BRAC	CC	CCLC	CFTC	CLOC	CODE_STATUS	CPOP	CRITICAL_FACTOR	CSTAT	DOPD	DOPT	ICC	ICFTC	LC	MLOC	P_DEFINE	P_ELIF	P_ELSE	P_ENDIF	P_IF	P_IFDEF	P_IFNDEF	P_INCLUDE	P_NEST	P_UNDEF	SLOC	STAT	TOPD	TOPT
ASOP		0.67	0.77	0.91	0.05	0.91	0.08	0.69	-0.08	0.87	-0.08	0.10	0.74	0.64	0.13	0.59	0.90	0.30	0.47	-0.02	0.29	0.30	0.58	0.18	0.12	0.20	0.31	0.26	0.91	0.91	0.96	0.95
BCOM	0.67		0.63	0.70	0.37	0.70	0.41	0.81	-0.04	0.77	0.05	0.30	0.62	0.52	-0.07	0.26	0.71	0.80	0.41	0.09	0.25	0.27	0.41	0.20	0.11	0.22	0.28	0.22	0.70	0.71	0.69	0.70
BLAN	0.77	0.63		0.88	-0.05	0.89	-0.03	0.78	-0.10	0.80	-0.08	0.21	0.90	0.72	-0.04	0.33	0.92	0.25	0.50	-0.02	0.52	0.49	0.68	0.38	0.30	0.41	0.48	0.22	0.89	0.89	0.88	0.87
BRAC	0.91	0.70	0.88		0.04	0.98	0.07	0.79	-0.06	0.93	-0.02	0.19	0.87	0.77	0.02	0.43	0.98	0.32	0.49	-0.02	0.45	0.44	0.64	0.33	0.21	0.37	0.40	0.28	0.98	0.97	0.95	0.97
CC	0.05	0.37	-0.05	0.04		0.03	0.94	0.03	0.04	0.27	0.02	-0.13	-0.07	-0.04	-0.05	-0.07	0.01	0.34	-0.18	-0.07	-0.15	-0.14	-0.15	-0.12	-0.06	0.03	-0.21	-0.08	0.03	0.09	0.03	0.05
CCLC	0.91	0.70	0.89	0.98	0.03		0.06	0.79	-0.06	0.92	-0.05	0.17	0.89	0.76	0.00	0.46	0.99	0.33	0.53	0.00	0.49	0.48	0.71	0.36	0.22	0.38	0.48	0.28	1.00	0.99	0.98	0.99
CFTC	0.08	0.41	-0.03	0.07	0.94	0.06		0.07	0.03	0.30	0.01	-0.11	-0.03	-0.02	-0.04	-0.05	0.04	0.37	-0.14	-0.06	-0.12	-0.11	-0.12	-0.10	-0.05	0.03	-0.16	-0.06	0.06	0.12	0.07	0.08
CLOC	0.69	0.81	0.78	0.79	0.03	0.79	0.07		-0.05	0.72	0.07	0.55	0.76	0.64	-0.02	0.31	0.84	0.55	0.47	0.04	0.41	0.41	0.59	0.32	0.17	0.29	0.43	0.23	0.79	0.78	0.75	0.76
CODE_STATUS	-0.08	-0.04	-0.10	-0.06	0.04	-0.06	0.03	-0.05		-0.03	0.75	0.04	-0.10	-0.09	-0.03	-0.04	-0.07	-0.03	-0.10	-0.04	-0.08	-0.08	-0.08	-0.07	-0.03	-0.03	-0.11	-0.04	-0.06	-0.05	-0.07	-0.07
CPOP	0.87	0.77	0.80	0.93	0.27	0.92	0.30	0.72	-0.03		0.01	0.10	0.78	0.66	-0.04	0.46	0.90	0.42	0.39	-0.04	0.41	0.38	0.57	0.28	0.23	0.40	0.35	0.23	0.92	0.93	0.91	0.92
CRITICAL_FACTOR	-0.08	0.05	-0.08	-0.02	0.02	-0.05	0.01	0.07	0.75	0.01		0.27	-0.08	-0.06	-0.03	-0.04	-0.04	0.07	-0.11	-0.05	-0.09	-0.09	-0.09	-0.08	-0.04	-0.02	-0.13	-0.05	-0.05	-0.04	-0.07	-0.06
CSTAT	0.10	0.30	0.21	0.19	-0.13	0.17	-0.11	0.55	0.04	0.10	0.27		0.26	0.23	0.01	-0.02	0.23	0.33	0.16	0.02	0.15	0.18	0.20	0.16	-0.04	0.14	0.17	0.11	0.17	0.15	0.13	0.14
DOPD	0.74	0.62	0.90	0.87	-0.07	0.89	-0.03	0.76	-0.10	0.78	-0.08	0.26		0.85	-0.02	0.24	0.91	0.31	0.60	0.08	0.67	0.67	0.74	0.58	0.24	0.59	0.64	0.34	0.89	0.87	0.85	0.86
DOPT	0.64	0.52	0.72	0.77	-0.04	0.76	-0.02	0.64	-0.09	0.66	-0.06	0.23	0.85		-0.04	0.19	0.77	0.23	0.59	0.11	0.50	0.49	0.51	0.44	0.09	0.45	0.47	0.37	0.76	0.74	0.72	0.73
ICC	0.13	-0.07	-0.04	0.02	-0.05	0.00	-0.04	-0.02	-0.03	-0.04	-0.03	0.01	-0.02	-0.04		-0.02	-0.00	-0.07	-0.06	-0.03	-0.05	-0.05	-0.05	-0.04	-0.02	-0.07	-0.08	-0.03	0.00	0.01	0.04	0.03
ICFTC	0.59	0.26	0.33	0.43	-0.07	0.46	-0.05	0.31	-0.04	0.46	-0.04	-0.02	0.24	0.19	-0.02		0.44	0.06	0.27	-0.04	0.07	0.10	0.29	0.03	-0.03	0.03	0.17	0.02	0.46	0.44	0.51	0.51
LC	0.90	0.71	0.92	0.98	0.01	0.99	0.04	0.84	-0.07	0.90	-0.04	0.23	0.91	0.77	-0.00	0.44		0.33	0.53	-0.00	0.50	0.49	0.72	0.37	0.23	0.39	0.48	0.27	0.99	0.99	0.97	0.98
MLOC	0.30	0.80	0.25	0.32	0.34	0.33	0.37	0.55	-0.03	0.42	0.07	0.33	0.31	0.23	-0.07	0.06	0.33		0.27	0.26	0.19	0.22	0.15	0.21	0.00	0.10	0.26	0.18	0.33	0.33	0.31	0.32
P_DEFINE	0.47	0.41	0.50	0.49	-0.18	0.53	-0.14	0.47	-0.10	0.39	-0.11	0.16	0.60	0.59	-0.06	0.27	0.53	0.27		0.39	0.67	0.62	0.51	0.59	0.10	0.42	0.55	0.75	0.53	0.48	0.53	0.50
P_ELIF	-0.02	0.09	-0.02	-0.02	-0.07	0.00	-0.06	0.04	-0.04	-0.04	-0.05	0.02	0.08	0.11	-0.03	-0.04	-0.00	0.26	0.39		0.28	0.32	-0.05	0.40	-0.03	0.10	0.38	0.20	0.00	-0.01	0.00	-0.01
P_ELSE	0.29	0.25	0.52	0.45	-0.15	0.49	-0.12	0.41	-0.08	0.41	-0.09	0.15	0.67	0.50	-0.05	0.07	0.50	0.19	0.67	0.28		0.90	0.68	0.87	0.33	0.75	0.85	0.48	0.49	0.46	0.42	0.41
P_ENDIF	0.30	0.27	0.49	0.44	-0.14	0.48	-0.11	0.41	-0.08	0.38	-0.09	0.18	0.67	0.49	-0.05	0.10	0.49	0.22	0.62	0.32	0.90		0.74	0.98	0.18	0.78	0.88	0.34	0.48	0.42	0.39	0.39
P_IF	0.58	0.41	0.68	0.64	-0.15	0.71	-0.12	0.59	-0.08	0.57	-0.09	0.20	0.74	0.51	-0.05	0.29	0.72	0.15	0.51	-0.05	0.68	0.74		0.60	0.17	0.59	0.76	0.20	0.71	0.66	0.66	0.66
P_IFDEF	0.18	0.20	0.38	0.33	-0.12	0.36	-0.10	0.32	-0.07	0.28	-0.08	0.16	0.58	0.44	-0.04	0.03	0.37	0.21	0.59	0.40	0.87	0.98	0.60		0.13	0.76	0.83	0.36	0.36	0.31	0.28	0.27
P_IFNDEF	0.12	0.11	0.30	0.21	-0.06	0.22	-0.05	0.17	-0.03	0.23	-0.04	-0.04	0.24	0.09	-0.02	-0.03	0.23	0.00	0.10	-0.03	0.33	0.18	0.17	0.13		0.14	0.34	-0.04	0.22	0.24	0.20	0.19
P_INCLUDE	0.20	0.22	0.41	0.37	0.03	0.38	0.03	0.29	-0.03	0.40	-0.02	0.14	0.59	0.45	-0.07	0.03	0.39	0.10	0.42	0.10	0.75	0.78	0.59	0.76	0.14		0.61	0.35	0.38	0.35	0.28	0.30
P_NEST	0.31	0.28	0.48	0.40	-0.21	0.48	-0.16	0.43	-0.11	0.35	-0.13	0.17	0.64	0.47	-0.08	0.17	0.48	0.26	0.55	0.38	0.85	0.88	0.76	0.83	0.34	0.61		0.19	0.48	0.42	0.42	0.41
P_UNDEF	0.26	0.22	0.22	0.28	-0.08	0.28	-0.06	0.23	-0.04	0.23	-0.05	0.11	0.34	0.37	-0.03	0.02	0.27	0.18	0.75	0.20	0.48	0.34	0.20	0.36	-0.04	0.35	0.19		0.28	0.25	0.29	0.27
SLOC	0.91	0.70	0.89	0.98	0.03	1.00	0.06	0.79	-0.06	0.92	-0.05	0.17	0.89	0.76	0.00	0.46	0.99	0.33	0.53	0.00	0.49	0.48	0.71	0.36	0.22	0.38	0.48	0.28		0.99	0.98	0.99
STAT	0.91	0.71	0.89	0.97	0.09	0.99	0.12	0.78	-0.05	0.93	-0.04	0.15	0.87	0.74	0.01	0.44	0.99	0.33	0.48	-0.01	0.46	0.42	0.66	0.31	0.24	0.35	0.42	0.25	0.99		0.97	0.98
TOPD	0.96	0.69	0.88	0.95	0.03	0.98	0.07	0.75	-0.07	0.91	-0.07	0.13	0.85	0.72	0.04	0.51	0.97	0.31	0.53	0.00	0.42	0.39	0.66	0.28	0.20	0.28	0.42	0.29	0.98	0.97		1.00
TOPT	0.95	0.70	0.87	0.97	0.05	0.99	0.08	0.76	-0.07	0.92	-0.06	0.14	0.86	0.73	0.03	0.51	0.98	0.32	0.50	-0.01	0.41	0.39	0.66	0.27	0.19	0.30	0.41	0.27	0.99	0.98	1.00	

The Capitalisation Base correlation table for files measures with a highlighting threshold of 0.8

Base measures are the ones directly reported by various tools included in the analysis. Derived measures are metrics computed based on these base measures or other derived measures.

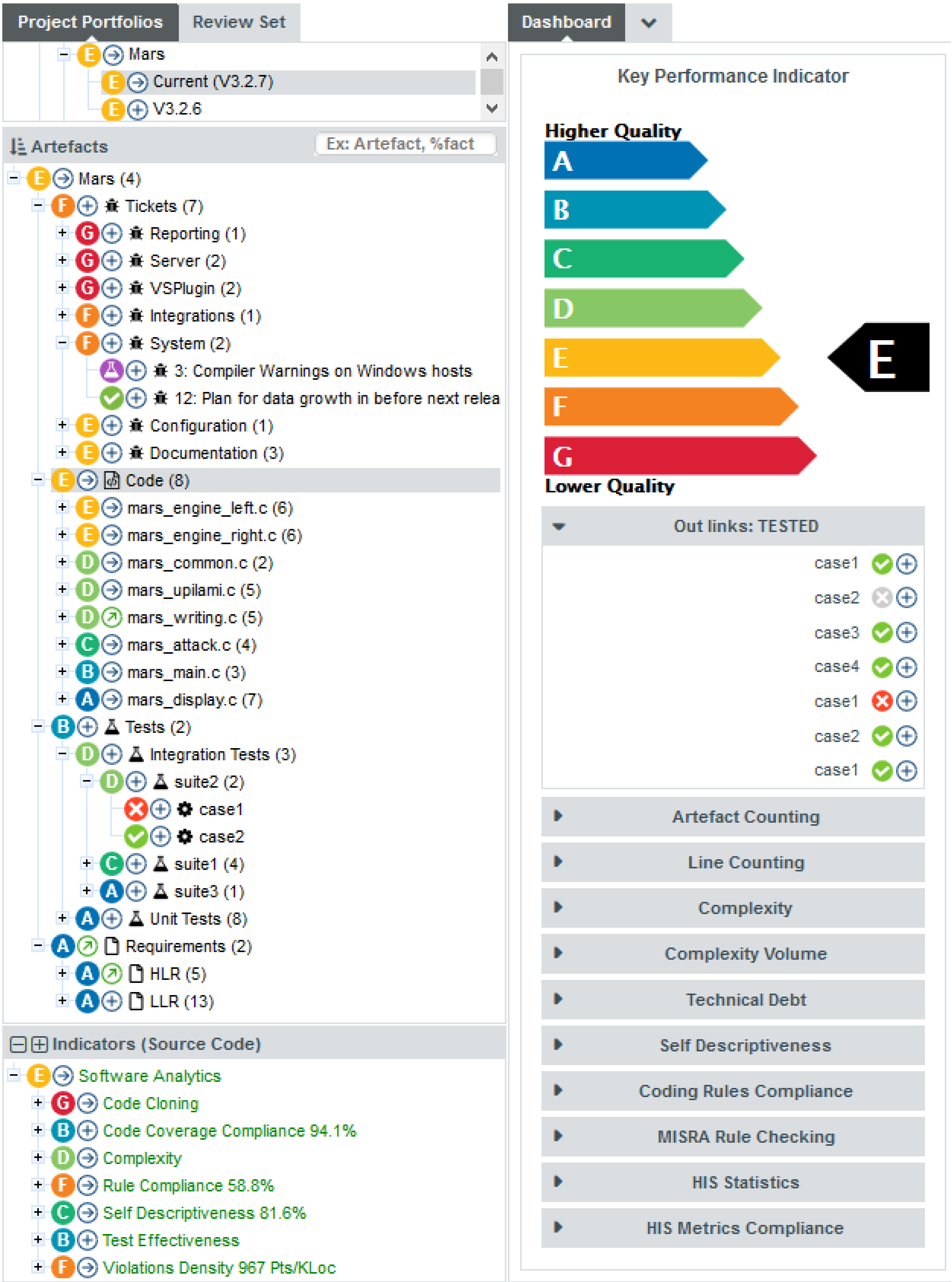
You can choose to export the results of the correlation matrix to a CSV file. The resulting CSV file contains all metrics pairs for which a correlation exists.

If the Export button is greyed out, your licence does not include the option to export data to CSV files.

Chapter 7. Going Beyond Source Code

Squore allows incorporating more than source code in your projects. Software Analytics, the default analysis model and has built-in support for importing requirements and tickets related to your development project, as well as results from your test campaigns and resource usage.

The Mars project is an example of a C development project that also allows you to track the progress of tests and tasks using the **Test Effectiveness**, **Code Coverage Compliance**, **Innovation Rate**, **Tickets Completion Rate** and **Verified Ratio** indicators.



Project Portfolios | Review Set

- E Mars
  - E Current (V3.2.7)
  - E V3.2.6

Artefacts — Ex: Artefact, %fact

- E Mars (4)
  - F Tickets (7)
    - G Reporting (1)
    - G Server (2)
    - G VSPlugin (2)
    - F Integrations (1)
    - F System (2)
      - 3: Compiler Warnings on Windows hosts
      - 12: Plan for data growth in before next relea
    - E Configuration (1)
    - E Documentation (3)
  - E Code (8)
    - E mars_engine_left.c (6)
    - E mars_engine_right.c (6)
    - D mars_common.c (2)
    - D mars_upilami.c (5)
    - D mars_writing.c (5)
    - C mars_attack.c (4)
    - B mars_main.c (3)
    - A mars_display.c (7)
  - B Tests (2)
    - D Integration Tests (3)
      - D suite2 (2)
        - case1
        - case2
      - C suite1 (4)
      - A suite3 (1)
    - A Unit Tests (8)
  - A Requirements (2)
    - A HLR (5)
    - A LLR (13)

Indicators (Source Code)

- E Software Analytics
  - G Code Cloning
  - B Code Coverage Compliance 94.1%
  - D Complexity
  - F Rule Compliance 58.8%
  - C Self Descriptiveness 81.6%
  - B Test Effectiveness
  - F Violations Density 967 Pts/KLoc

Dashboard

Key Performance Indicator

Out links: TESTED

case1	✓
case2	✗
case3	✓
case4	✓
case1	✗
case2	✓
case1	✓

- Artefact Counting
- Line Counting
- Complexity
- Complexity Volume
- Technical Debt
- Self Descriptiveness
- Coding Rules Compliance
- MISRA Rule Checking
- HIS Statistics
- HIS Metrics Compliance

Ticket, test and source code artefacts in the Mars and the score card for a source code file

This chapter describes how to understand these indicators using the charts and highlight categories in the Software Analytics model and shows you how to use the available Data Providers to import test and ticket data into your own Squore projects.

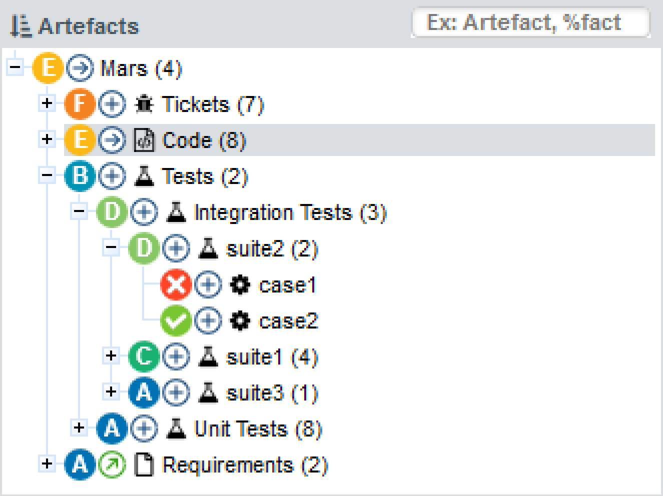
Test Management

The Test Effectiveness indicator is based on a ratio of passed and failed tests in your project.

It is enabled by default in your project if you use a Data Provider that imports test artefacts, like JUnit, VectorCAST, RTRT or MSTest.

The Code Coverage Compliance indicator is based on function coverage metrics imported by Data Providers like RTRT, JaCoCo, GCov or NCover among others.

In order to view test results, expand the **Tests** node in the Artefact Tree for the Mars and switch to the **Code Coverage** dashboard, which provides information specific to the test results for your project.



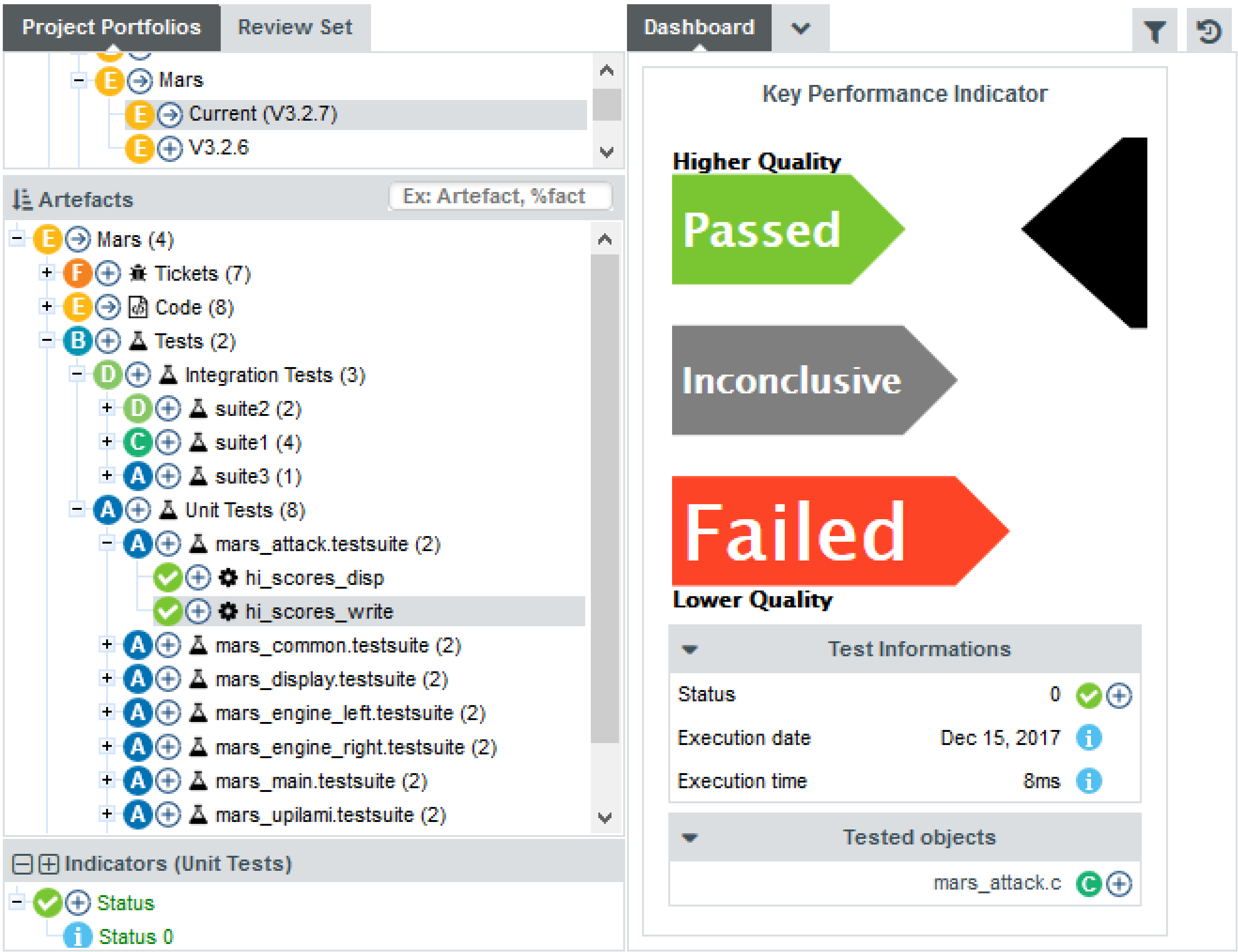
Artefacts   Ex: Artefact, %fact

- E Mars (4)
  - F Tickets (7)
  - E Code (8)
  - B Tests (2)
    - D Integration Tests (3)
      - D suite2 (2)
        - ✗ case1
        - ✓ case2
      - C suite1 (4)
      - A suite3 (1)
    - A Unit Tests (8)
  - A Requirements (2)

Test artefacts in Mars

Expand the **Tests** node in the tree to uncover the hierarchy of tests. Each test is imported with its execution result, date and run time. The overall rating of the Tests node is the test effectiveness indicator, which also contributes to the overall project rating.

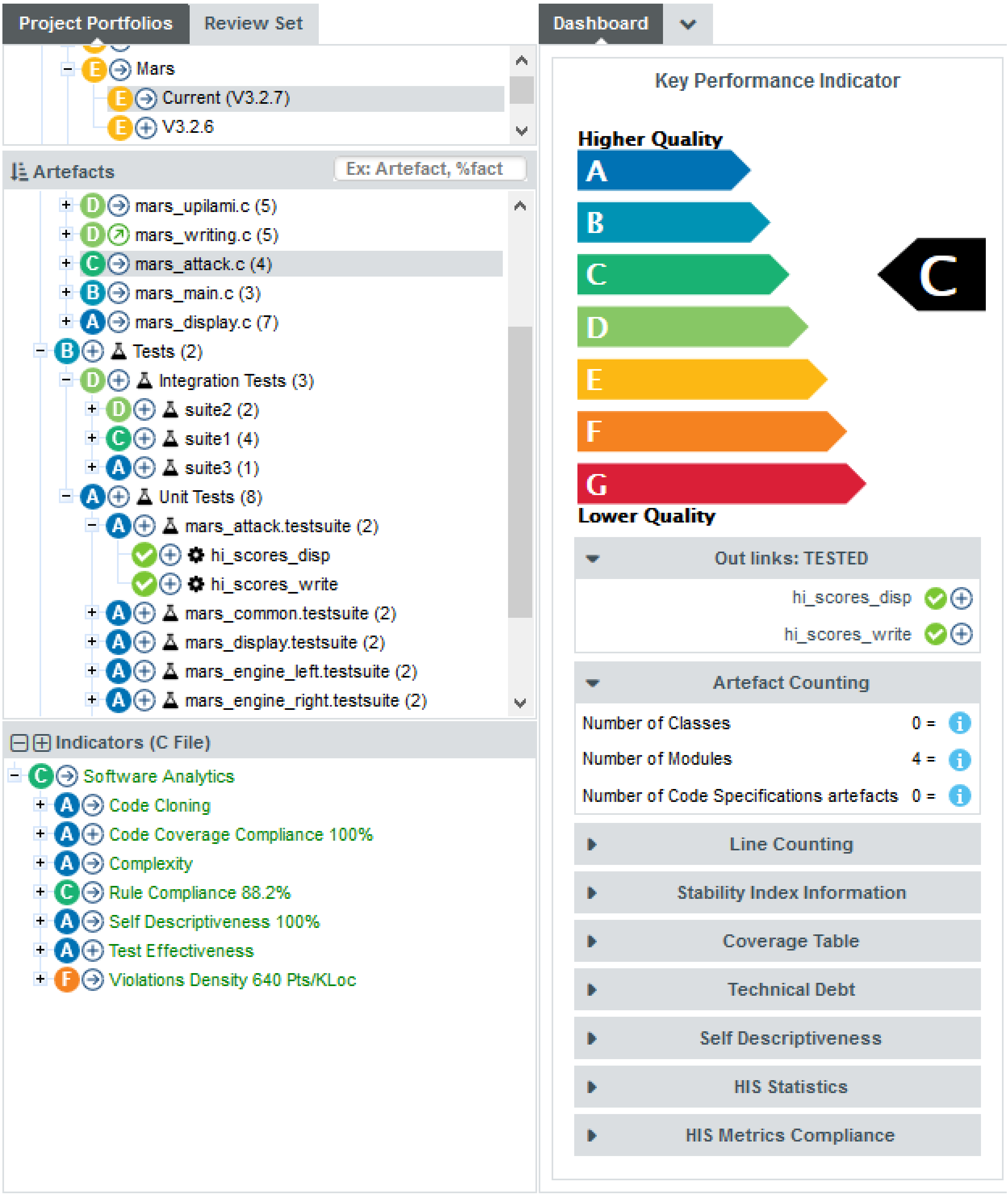
Clicking a test artefact displays a dashboard containing the test information, as well as a table containing links to the objects tested by the test, as shown below:



Test Informations	
Status	0
Execution date	Dec 15, 2017
Execution time	8ms

The dashboard for a test artefact

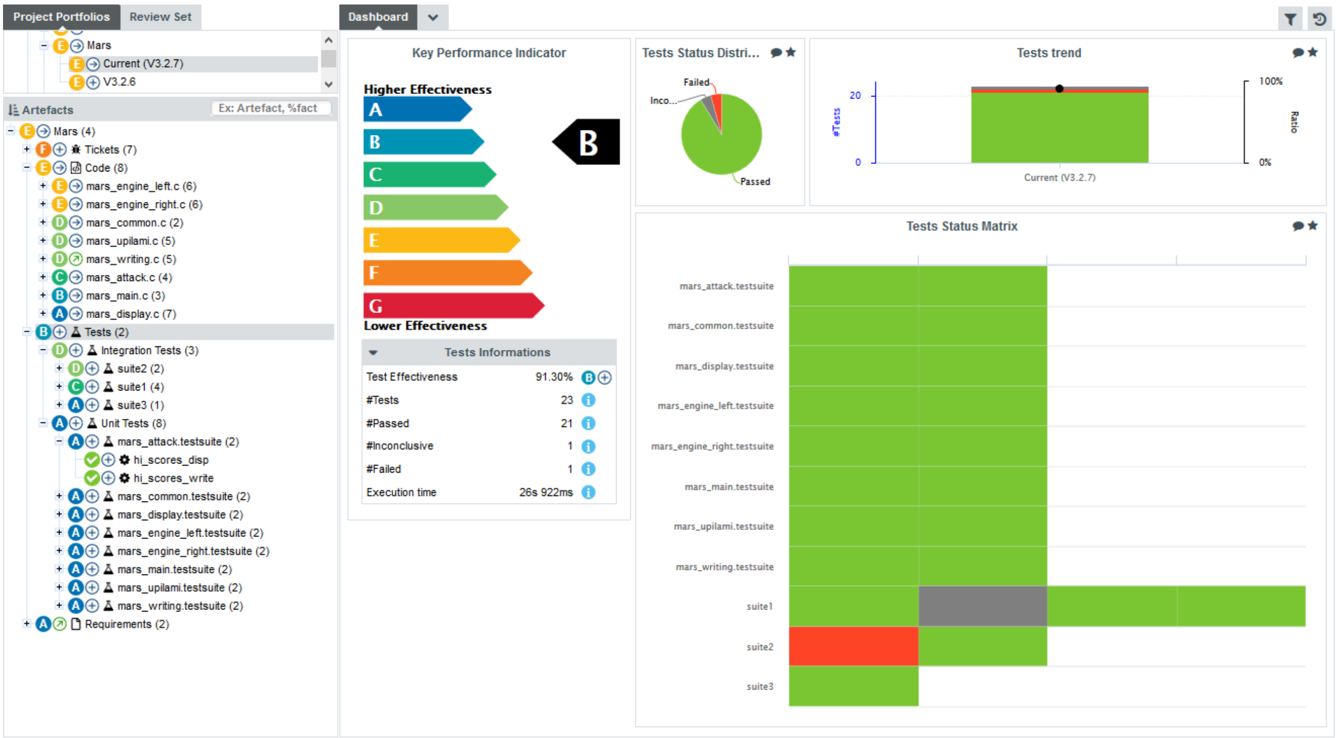
Clicking the link to the tested source code file brings up a dashboard showing the complete list of tests covering the source file, which are also clickable.



Number of Classes	0 =
Number of Modules	4 =
Number of Code Specifications artefacts	0 =

Links to tests in a dashboard for a source code file

You can click the root test node (or any test suite artefact) to get a history of test execution as well as a text execution matrix showing the test results of all test cases in the project.



- E Mars (4)
  - F Tickets (7)
  - E Code (8)
    - E mars_engine_left.c (6)
    - E mars_engine_right.c (6)
    - D mars_common.c (2)
    - D mars_upilami.c (5)
    - D mars_writing.c (5)
    - C mars_attack.c (4)
    - B mars_main.c (3)
    - A mars_display.c (7)
  - B Tests (2)
    - D Integration Tests (3)
      - D suite2 (2)
      - C suite1 (4)
      - A suite3 (1)
    - A Unit Tests (8)
      - A mars_attack.testsuite (2)
        - ✓ hi_scores_disp
        - ✓ hi_scores_write
      - A mars_common.testsuite (2)
      - A mars_display.testsuite (2)
      - A mars_engine_left.testsuite (2)
      - A mars_engine_right.testsuite (2)
      - A mars_main.testsuite (2)
      - A mars_upilami.testsuite (2)
      - A mars_writing.testsuite (2)
  - A Requirements (2)

Tests Informations

Test Effectiveness	91.30% B
#Tests	23
#Passed	21
#Inconclusive	1
#Failed	1
Execution time	26s 922ms

Test status summary for the project

The Test Status Distribution chart can help you identify the failed tests immediately: click the chart thumbnail in the dashboard and click on the red pie slice to open the Data tab of the Chart Viewer. The tab shows the list of test artefacts with the Failed status. Each artefact is clickable so you can view its dashboard directly. You can additionally send the list of artefacts to the Review Set or create a highlight definition with these artefacts so you can display additional metrics for each of them.

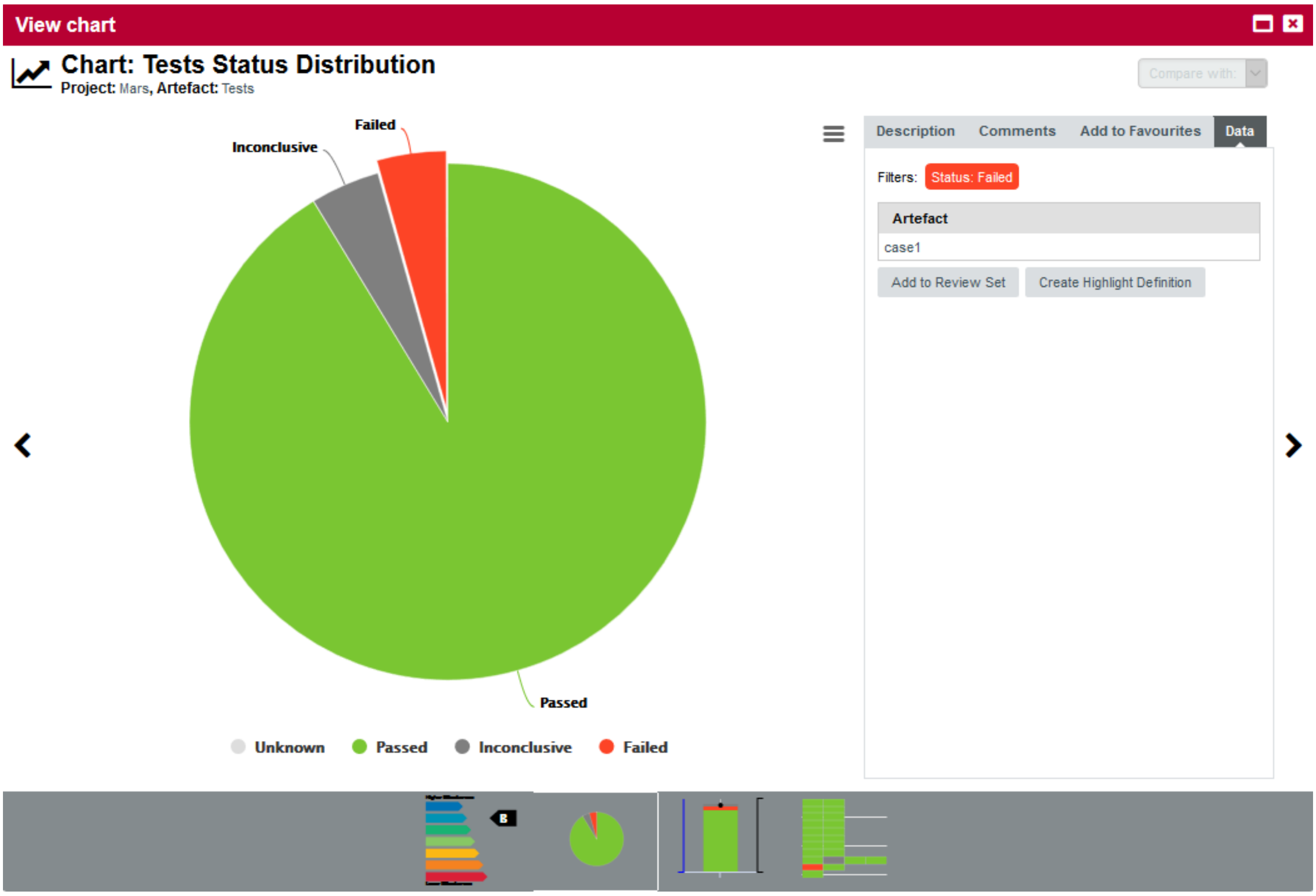


Chart: Tests Status Distribution

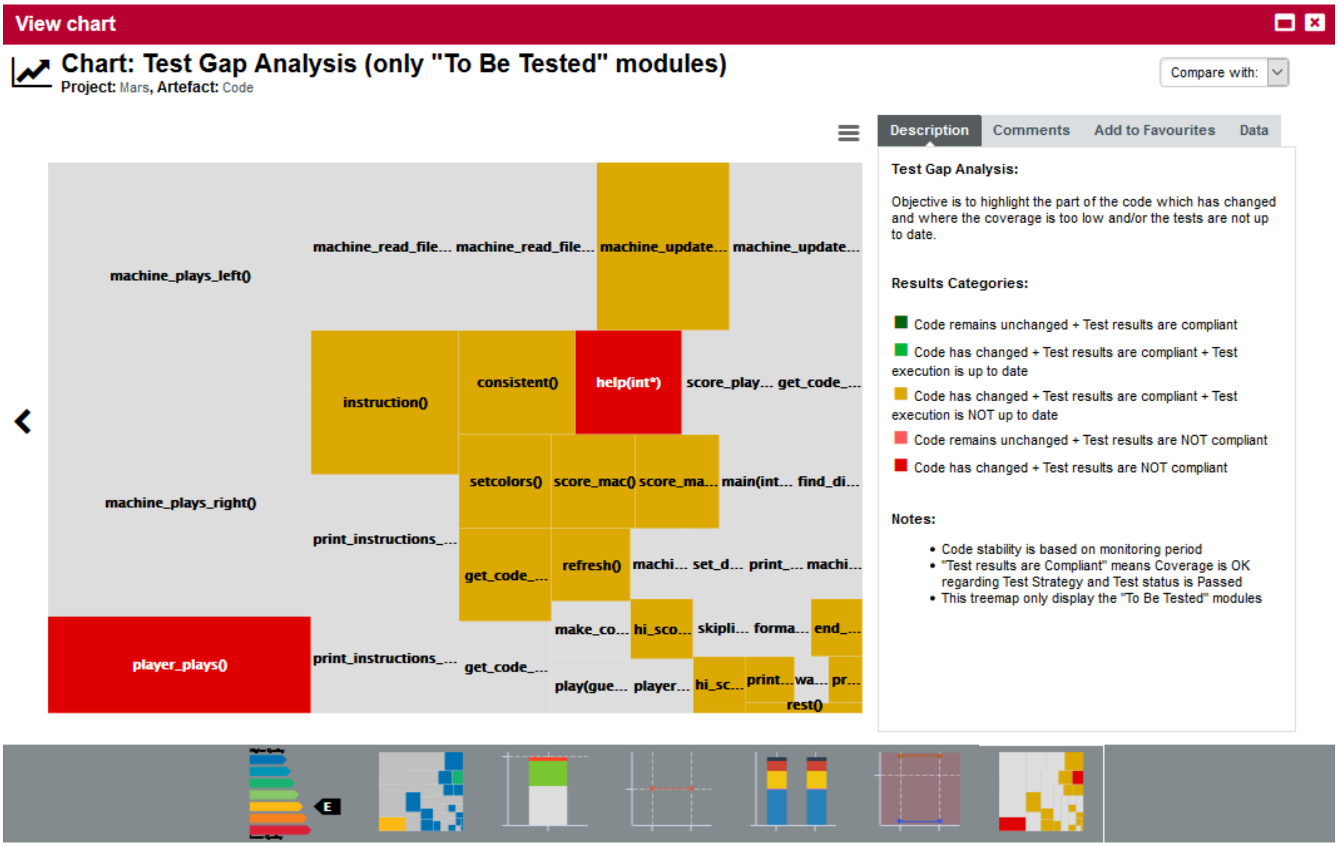
Project: Mars, Artefact: Tests

Filters: Status: Failed

Artefact
case1

Test status summary for the project

If you click the **Code** node, the dashboard includes a **Test Gap Analysis** chart that helps you adjust your test strategy by highlighting the risky artefacts in your project based on recent code changes and test execution status.



Test Gap Analysis:

Objective is to highlight the part of the code which has changed and where the coverage is too low and/or the tests are not up to date.

Results Categories:

🟩 Code remains unchanged + Test results are compliant

🟩 Code has changed + Test results are compliant + Test execution is up to date

🟨 Code has changed + Test results are compliant + Test execution is NOT up to date

🟥 Code remains unchanged + Test results are NOT compliant

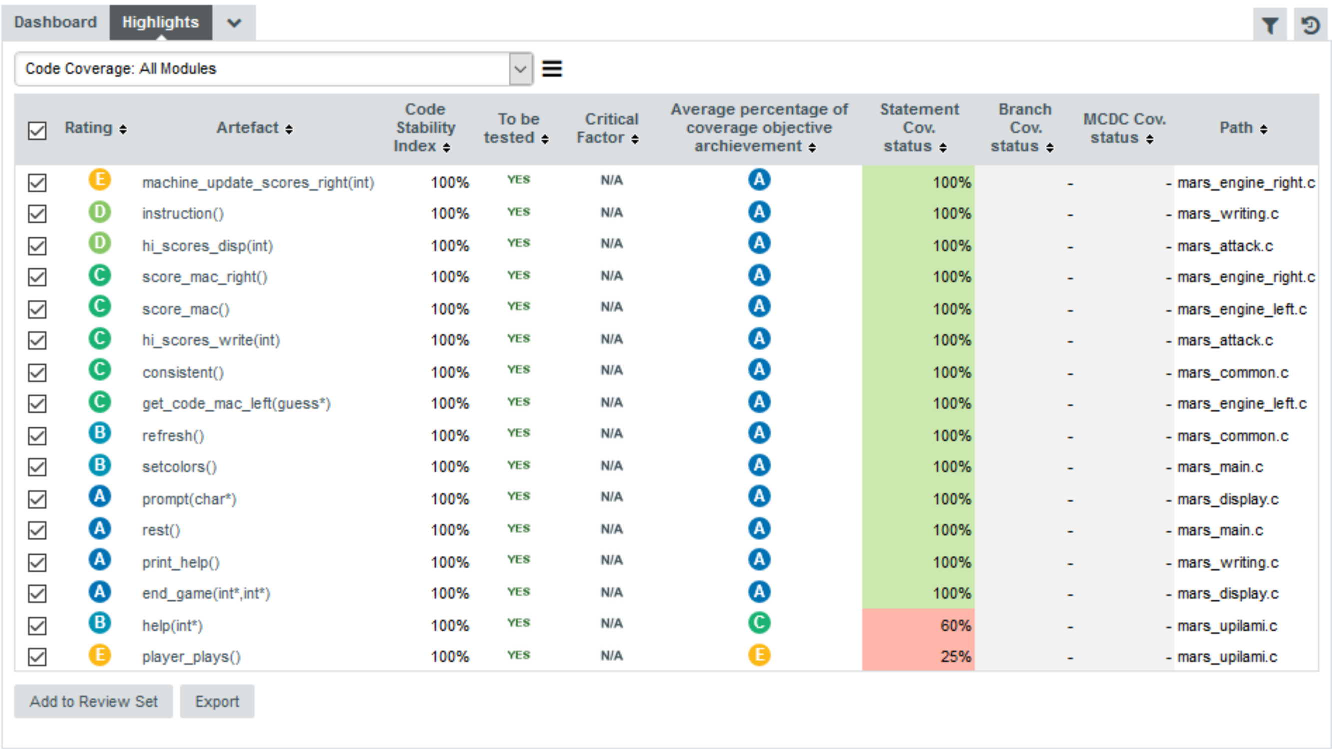
🟥 Code has changed + Test results are NOT compliant

Notes:

- Code stability is based on monitoring period
- "Test results are Compliant" means Coverage is OK regarding Test Strategy and Test status is Passed
- This treemap only display the "To Be Tested" modules

The Test Gap Analysis chart

The Highlights tab also contains predefined categories to help you analyse insufficiently tested artefacts, modified artefacts and display a summary of artefacts that comply with the code coverage threshold for the project. The screenshot below shows a list of function artefacts in the project together with their code coverage metrics:



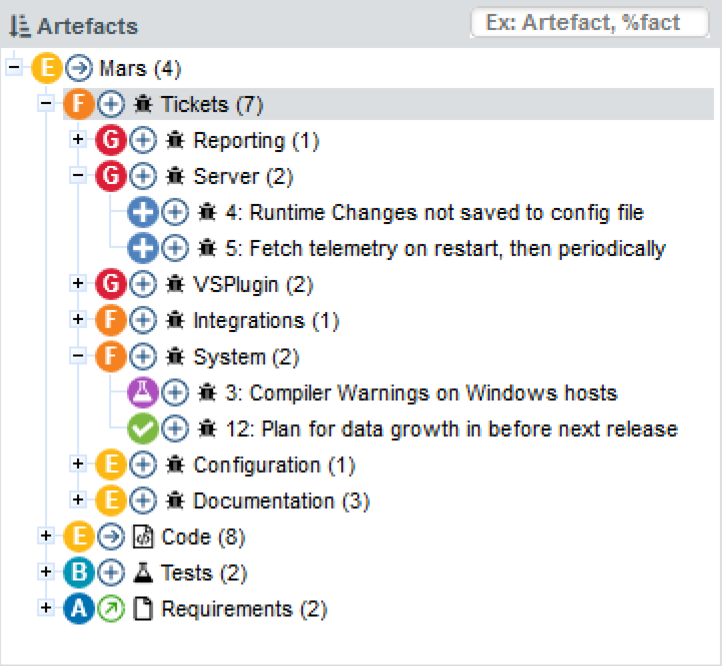
☑	Rating	Artefact	Code Stability Index	To be tested	Critical Factor	Average percentage of coverage objective archievement	Statement Cov. status	Branch Cov. status	MCDC Cov. status	Path
☑	E	machine_update_scores_right(int)	100%	YES	N/A	A	100%	-	-	mars_engine_right.c
☑	D	instruction()	100%	YES	N/A	A	100%	-	-	mars_writing.c
☑	D	hi_scores_disp(int)	100%	YES	N/A	A	100%	-	-	mars_attack.c
☑	C	score_mac_right()	100%	YES	N/A	A	100%	-	-	mars_engine_right.c
☑	C	score_mac()	100%	YES	N/A	A	100%	-	-	mars_engine_left.c
☑	C	hi_scores_write(int)	100%	YES	N/A	A	100%	-	-	mars_attack.c
☑	C	consistent()	100%	YES	N/A	A	100%	-	-	mars_common.c
☑	C	get_code_mac_left(guess*)	100%	YES	N/A	A	100%	-	-	mars_engine_left.c
☑	B	refresh()	100%	YES	N/A	A	100%	-	-	mars_common.c
☑	B	setcolors()	100%	YES	N/A	A	100%	-	-	mars_main.c
☑	A	prompt(char*)	100%	YES	N/A	A	100%	-	-	mars_display.c
☑	A	rest()	100%	YES	N/A	A	100%	-	-	mars_main.c
☑	A	print_help()	100%	YES	N/A	A	100%	-	-	mars_writing.c
☑	A	end_game(int*,int*)	100%	YES	N/A	A	100%	-	-	mars_display.c
☑	B	help(int*)	100%	YES	N/A	C	60%	-	-	mars_upilami.c
☑	E	player_plays()	100%	YES	N/A	E	25%	-	-	mars_upilami.c

The Code Coverage highlight showing compliant and non-compliant modules

Ticket Management

Importing tickets related to your project activates the **Ticket Completion Rate** and **Innovation Rate** indicators in your project.

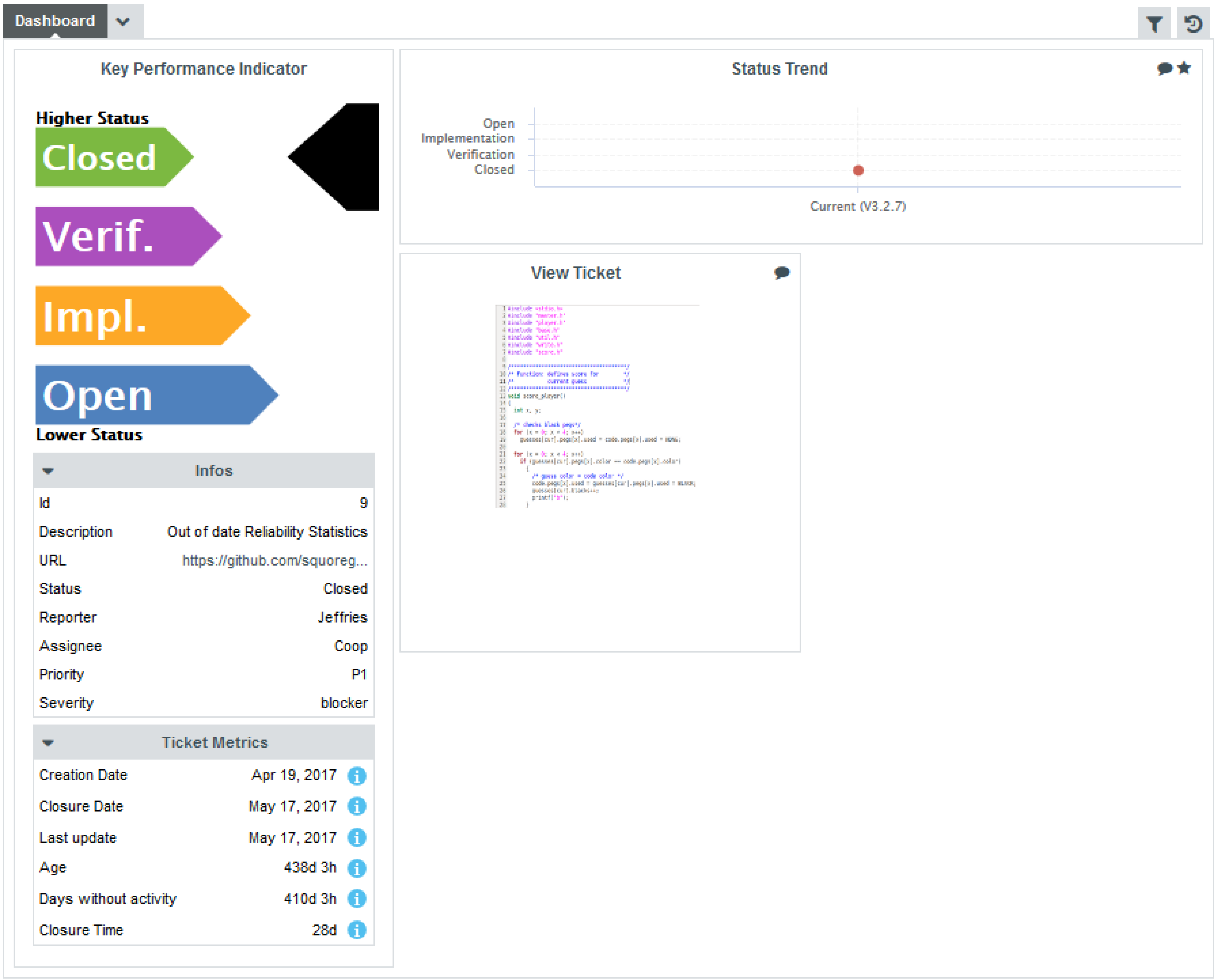
The Ticket Completion Rate helps you track the ratio of open and closed tickets in your project, while the Innovation Rate is an indicator of the ratio of enhancements versus defects being worked on. Both indicators are enabled when importing tickets using a Data Provider like Mantis or Jira.



- E Mars (4)
  - F Tickets (7)
    - G Reporting (1)
    - G Server (2)
      - 4: Runtime Changes not saved to config file
      - 5: Fetch telemetry on restart, then periodically
    - G VSPlugin (2)
    - F Integrations (1)
    - F System (2)
      - 3: Compiler Warnings on Windows hosts
      - 12: Plan for data growth in before next release
    - E Configuration (1)
    - E Documentation (3)
  - E Code (8)
  - B Tests (2)
  - A Requirements (2)

Ticket artefacts in Mars, grouped by component

The dashboard for each ticket provides details about the ticket's activity and status history, as well as a direct link to open the ticket in your ticket management tool.



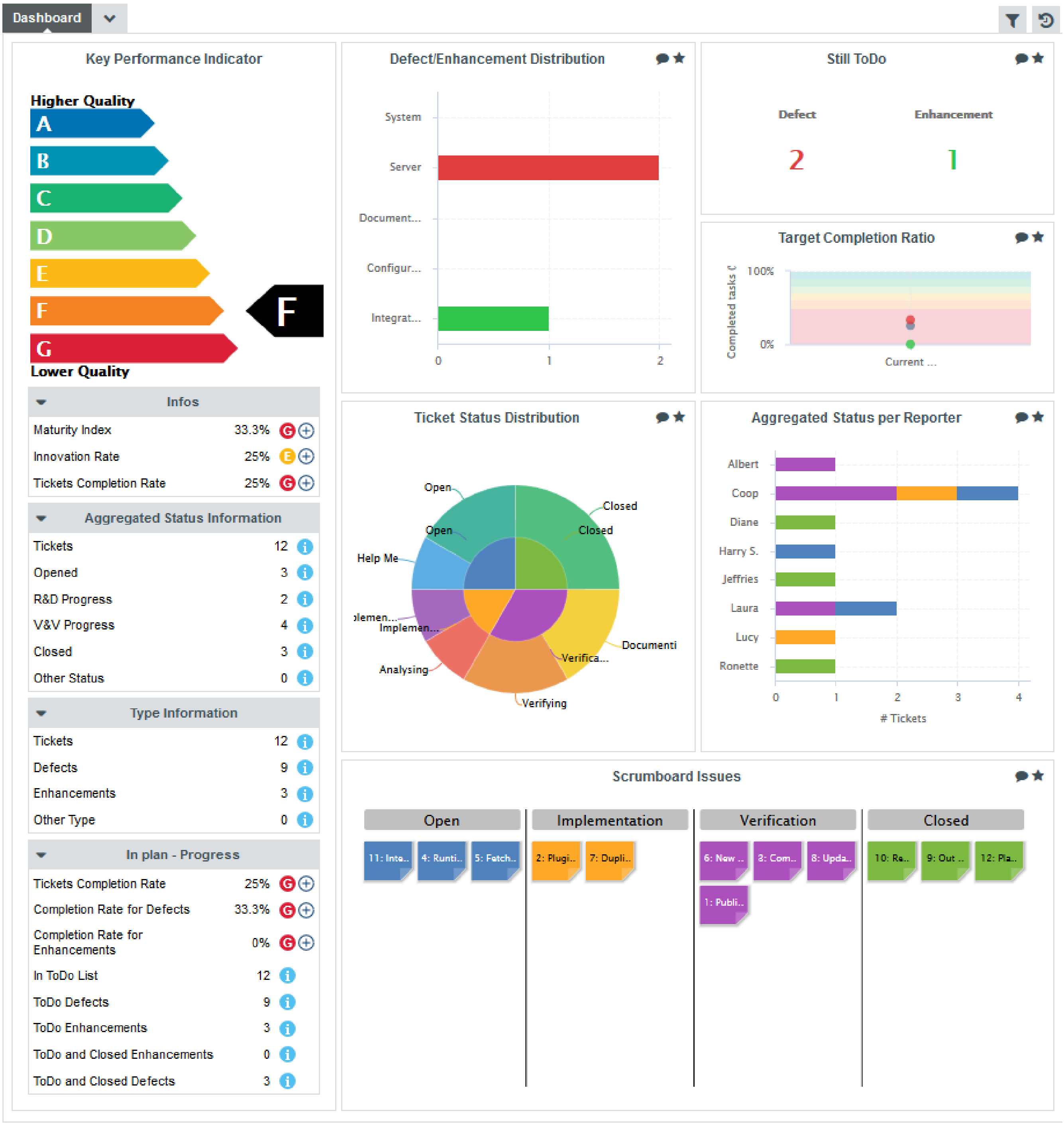
```
#include <stdio.h>
#include "master.h"
#include "player.h"
#include "base.h"
#include "util.h"
#include "win32.h"
#include "score.h"

/*****************************************/
/* Function: defines score for          */
/*           current guess              */
/*****************************************/
void score_player()
{
  int x, y;

  /* checks black pegs*/
  for (x = 0; x < 4; x++)
    guesses[cur].pegs[x].used = code.pegs[x].used = NONE;

  for (x = 0; x < 4; x++)
    if (guesses[cur].pegs[x].color == code.pegs[x].color)
      {
        /* guess color = code color */
        code.pegs[x].used = guesses[cur].pegs[x].used = BLACK;
        guesses[cur].blacks++;
        printf("B");
      }
```

Infos	
Id	9
Description	Out of date Reliability Statistics
URL	https://github.com/squoreg...
Status	Closed
Reporter	Jeffries
Assignee	Coop
Priority	P1
Severity	blocker

Ticket Metrics	
Creation Date	Apr 19, 2017
Closure Date	May 17, 2017
Last update	May 17, 2017
Age	438d 3h
Days without activity	410d 3h
Closure Time	28d

Ticket Dashboard

The dashboard for the **Tickets** node (or any ticket container artefact like each component in the Mars project) shows a summary of tickets by status and reporter, as well as a scrum board so you can keep track of task completion progress.



Infos		
Maturity Index	33.3%	G
Innovation Rate	25%	E
Tickets Completion Rate	25%	G

Aggregated Status Information	
Tickets	12
Opened	3
R&D Progress	2
V&V Progress	4
Closed	3
Other Status	0

Type Information	
Tickets	12
Defects	9
Enhancements	3
Other Type	0

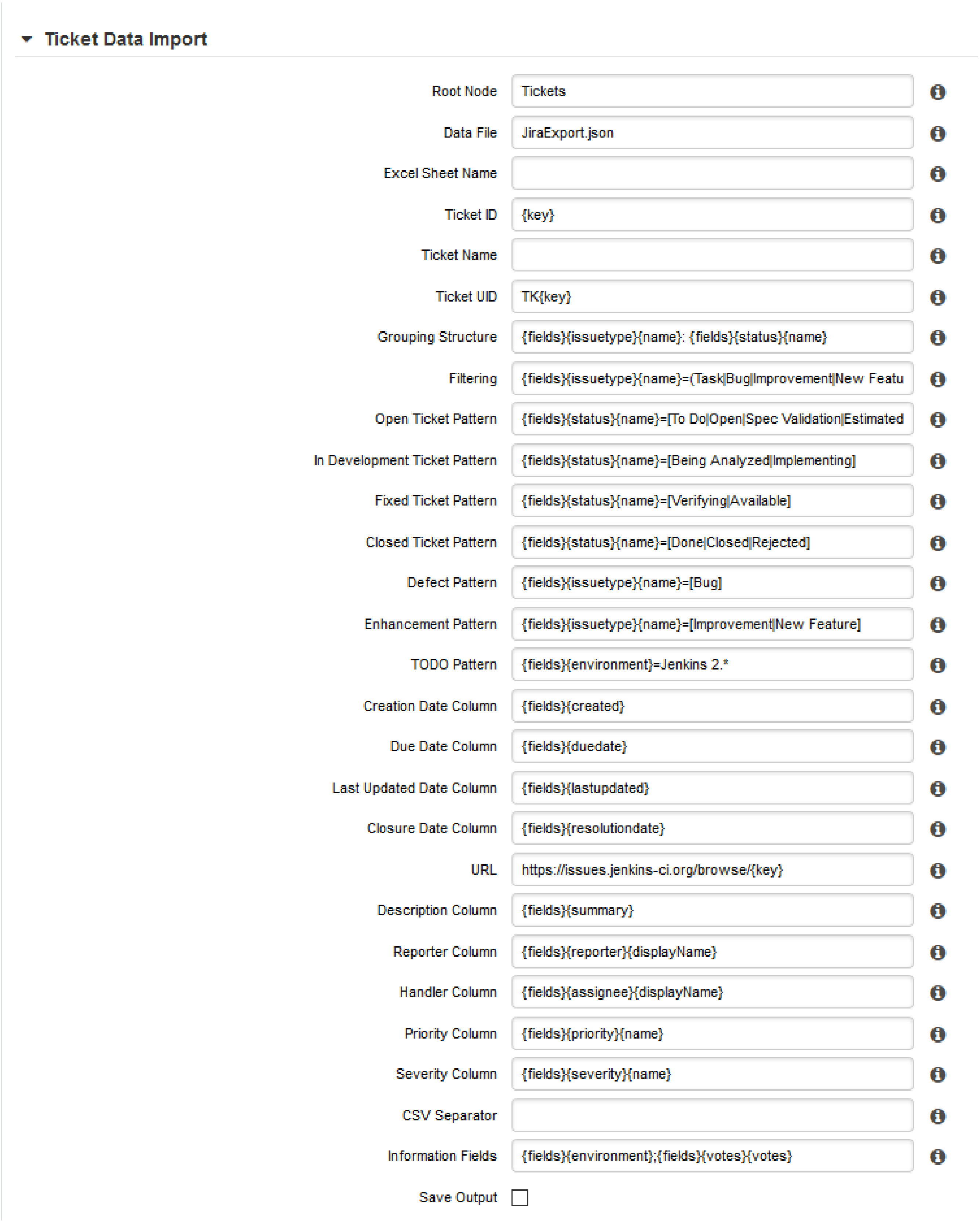
In plan - Progress		
Tickets Completion Rate	25%	G
Completion Rate for Defects	33.3%	G
Completion Rate for Enhancements	0%	G
In ToDo List	12	
ToDo Defects	9	
ToDo Enhancements	3	
ToDo and Closed Enhancements	0	
ToDo and Closed Defects	3	

Overall Dashboard for all tickets in the project

The ticket Data Provider are based on a common framework for importing tickets from various sources and in various file formats. To try it out in your project, you can start with a JSON or CSV file containing ticket data and define regular patterns to isolate data for the supported ticket metrics:

- Ticket ID and URL
- Grouping structure and filtering
- Definition of Open, In Progress, In Validation and Closed statuses
- Definition of defect and enhancement types
- Creation and closure dates
- Description, reporter, handler, priority and severity fields
- Any other data you want to import as textual information in Squore

As an example, the following is the definition used to extract data from issues exported from Jira in JSON format:



Ticket Data Import

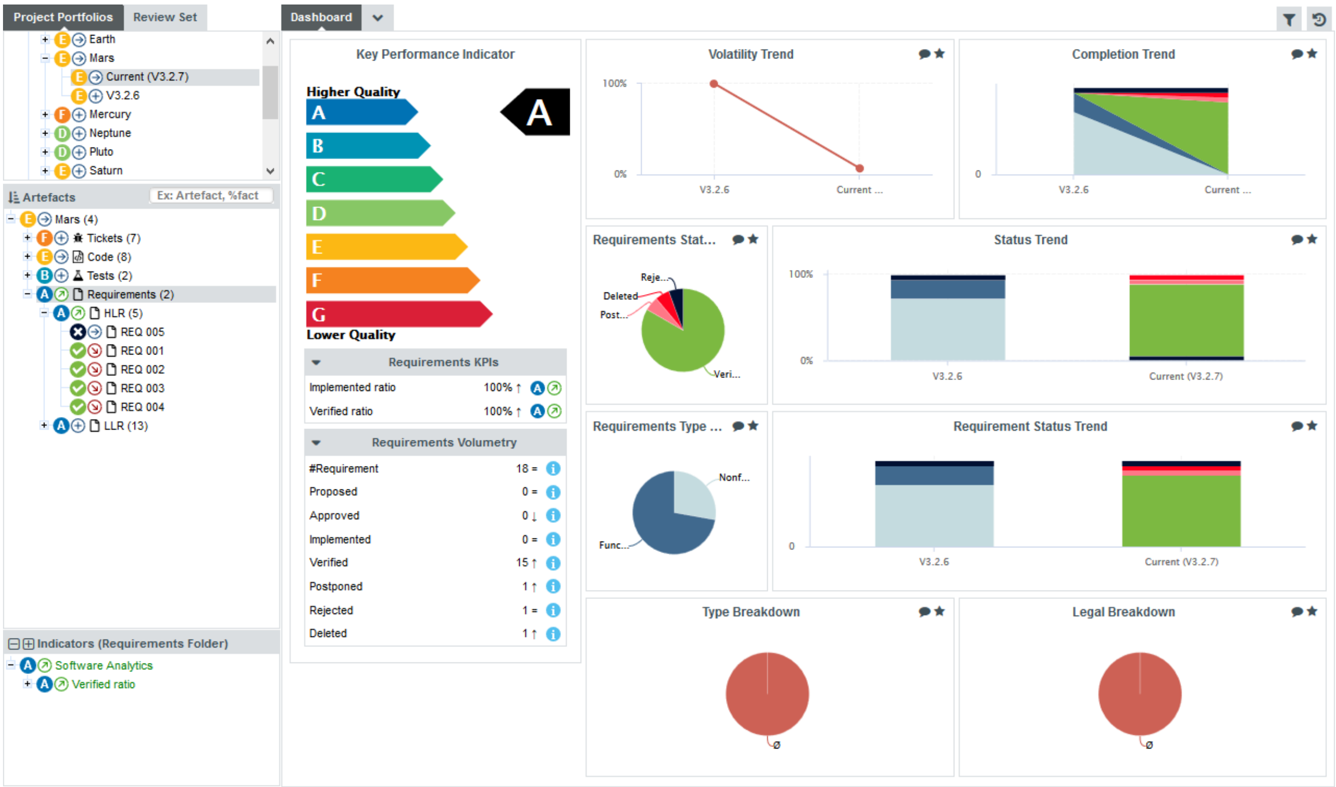
Root Node	Tickets
Data File	JiraExport.json
Excel Sheet Name	
Ticket ID	{key}
Ticket Name	
Ticket UID	TK{key}
Grouping Structure	{fields}{issuetype}{name}: {fields}{status}{name}
Filtering	{fields}{issuetype}{name}=(Task|Bug|Improvement|New Featu
Open Ticket Pattern	{fields}{status}{name}=[To Do|Open|Spec Validation|Estimated
In Development Ticket Pattern	{fields}{status}{name}=[Being Analyzed|Implementing]
Fixed Ticket Pattern	{fields}{status}{name}=[Verifying|Available]
Closed Ticket Pattern	{fields}{status}{name}=[Done|Closed|Rejected]
Defect Pattern	{fields}{issuetype}{name}=[Bug]
Enhancement Pattern	{fields}{issuetype}{name}=[Improvement|New Feature]
TODO Pattern	{fields}{environment}=Jenkins 2.*
Creation Date Column	{fields}{created}
Due Date Column	{fields}{duedate}
Last Updated Date Column	{fields}{lastupdated}
Closure Date Column	{fields}{resolutiondate}
URL	https://issues.jenkins-ci.org/browse/{key}
Description Column	{fields}{summary}
Reporter Column	{fields}{reporter}{displayName}
Handler Column	{fields}{assignee}{displayName}
Priority Column	{fields}{priority}{name}
Severity Column	{fields}{severity}{name}
CSV Separator	
Information Fields	{fields}{environment};{fields}{votes}{votes}
Save Output	☐

Parameters for a Jira JSON file

The Jira Data Provider automatically retrieves the file from your Jira instance and uses these parameters by default so you do not have to configure them for each project.

Requirement Management

Using a Data Provider to import requirements as artefacts in your project activate the **Verified Ratio** indicator. Various dashboards are available to track the volatility of your set of requirements and the evolution of single requirements.



Requirements KPIs

Implemented ratio	100% ↑	A
Verified ratio	100% ↑	A

Requirements Volumetry

#Requirement	18 =	ℹ
Proposed	0 =	ℹ
Approved	0 ↓	ℹ
Implemented	0 =	ℹ
Verified	15 ↑	ℹ
Postponed	1 ↑	ℹ
Rejected	1 =	ℹ
Deleted	1 ↑	ℹ

Overall Dashboard for all requirements in the project

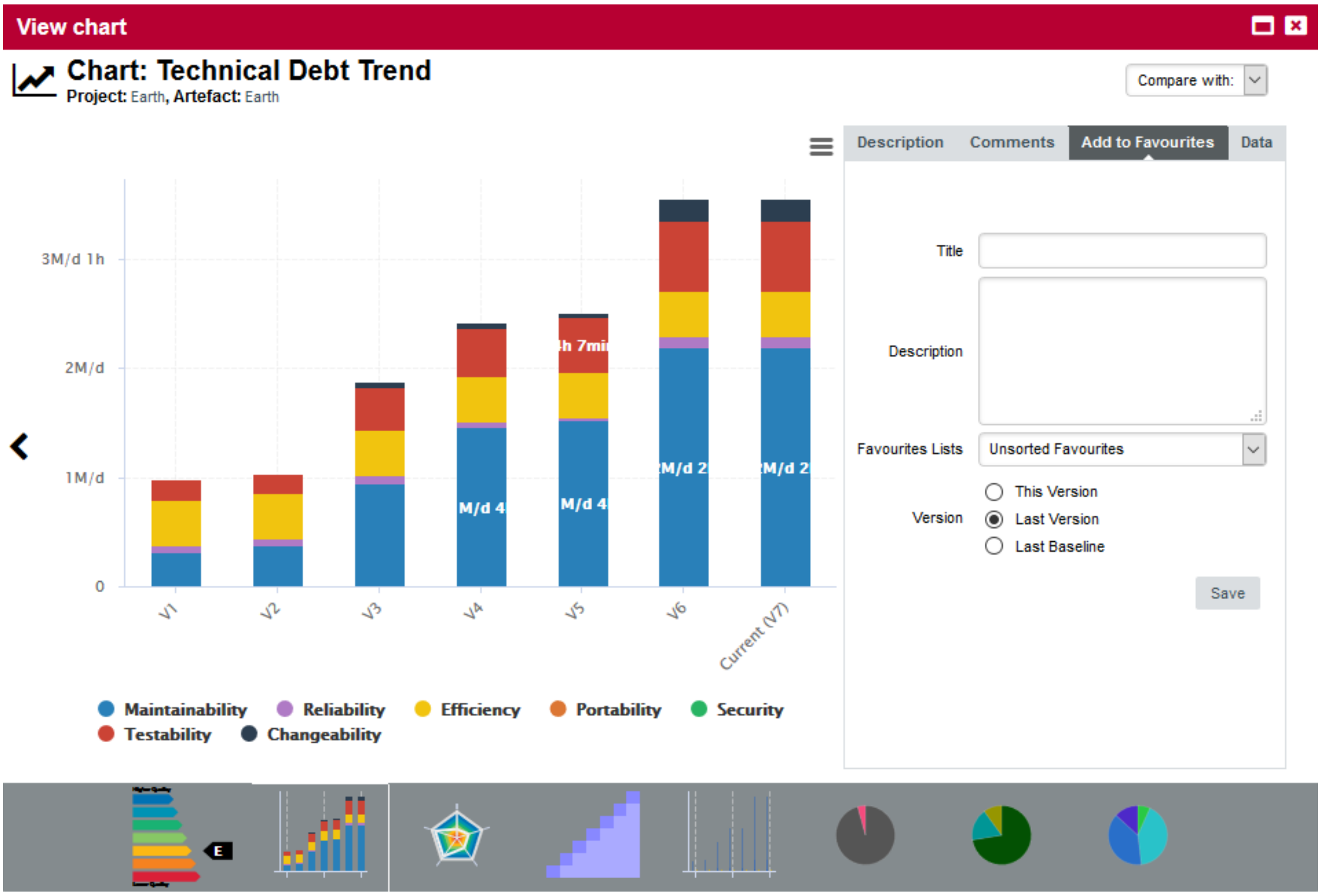
Chapter 8. Track Your Favourite Indicators

By clicking on Favourites in the main menu bar, you can view all the charts you marked as favourite in the dashboards across all of your projects. You can group charts into lists and reorder them as you see fit. The charts you mark as favourites are also the ones that are accessible to view on your mobile devices when away from your desk. This section covers everything you need to know about favourites and Squore's mobile interface: Squore Mobile.

Building a cross-project Dashboard in Favourites

Each of the chart thumbnails has a star (★) icon that you can click to mark a chart as favourite.

Clicking a star icon opens the Chart Viewer on the Favourites tab, as shown below:



View chart

Chart: Technical Debt Trend

Project: Earth, Artefact: Earth

Title:

Description:

Favourites Lists: Unsorted Favourites

Version: ○ This Version ● Last Version ○ Last Baseline

Adding a favourite in the chart viewer

The popup allows you to:

- Type a custom title and description for your chart so that you can for example write down why you are monitoring it.
- Select a list of favourites to add the chart to. By default, your charts are added to a list called Unsorted Favourites. You can create more lists and move charts between lists from the Favourites page.
- Select a version of the chart to display. The latest version is selected by default (**Last Version**), but you can alternatively select the exact version you clicked on (**Current Version**), or the latest baseline(**Last Baseline**).

When you are satisfied with your choices, click on **Save** to add the chart to your favourites. You can add charts from any project you have access to.

Refer to the next section to learn how to view and manage the charts you saved as favourites.

Managing Favourites

All the charts you added from the Explorer were added to a list called Unsorted Favourites. You can delete this list and create other lists using the ✚ and ━ icons.

When you have more than one list, you can drag and drop charts between lists.

In order to see the full size of a chart you marked as favourite, click its thumbnail on the left pane to open it in the right pane. The screenshot below shows an example of a list of favourites and a maximised chart. Note that the right pane contains links that allow you to go back to the project's or artefact's dashboard directly.

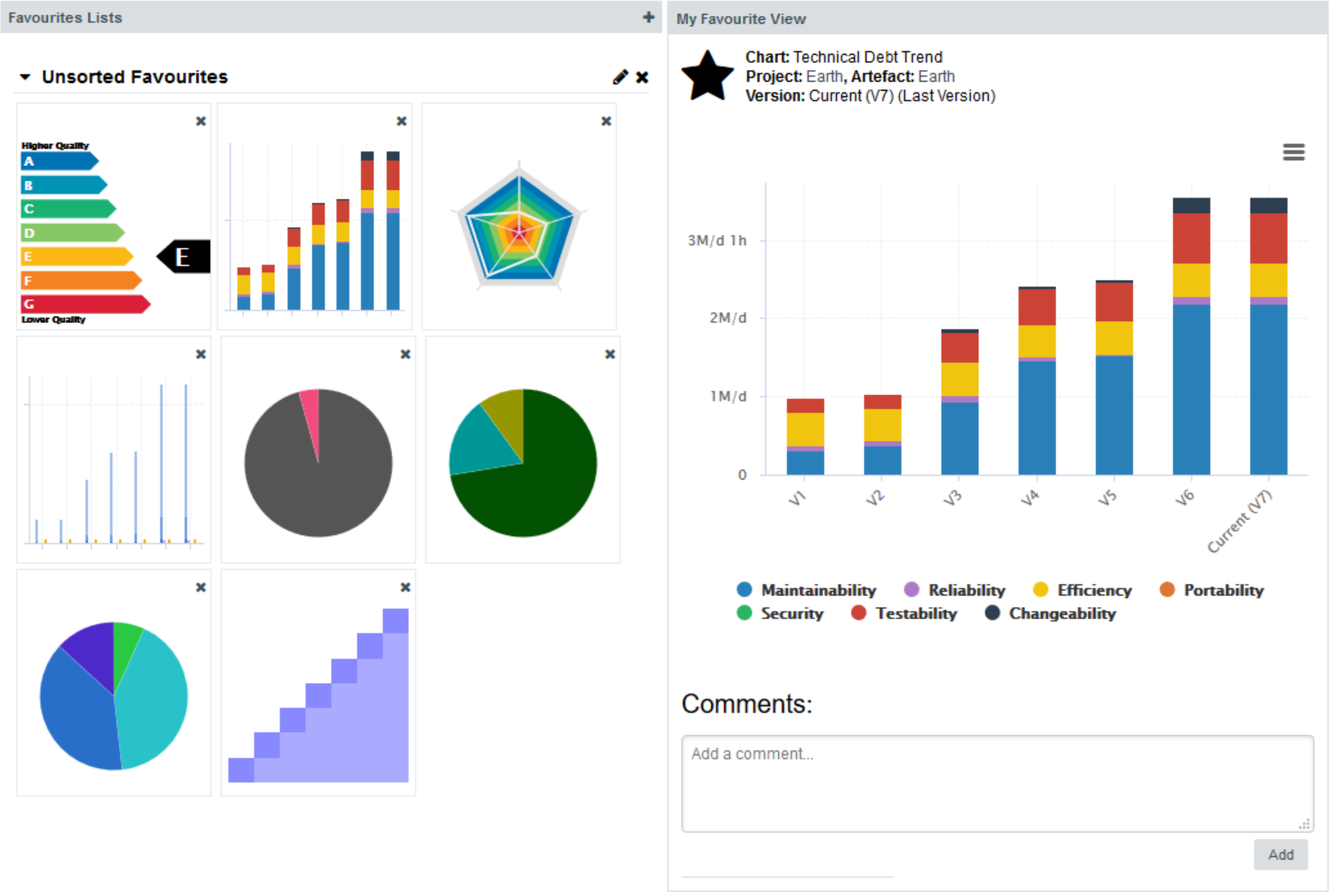


Chart: Technical Debt Trend
Project: Earth, **Artefact:** Earth
Version: Current (V7) (Last Version)

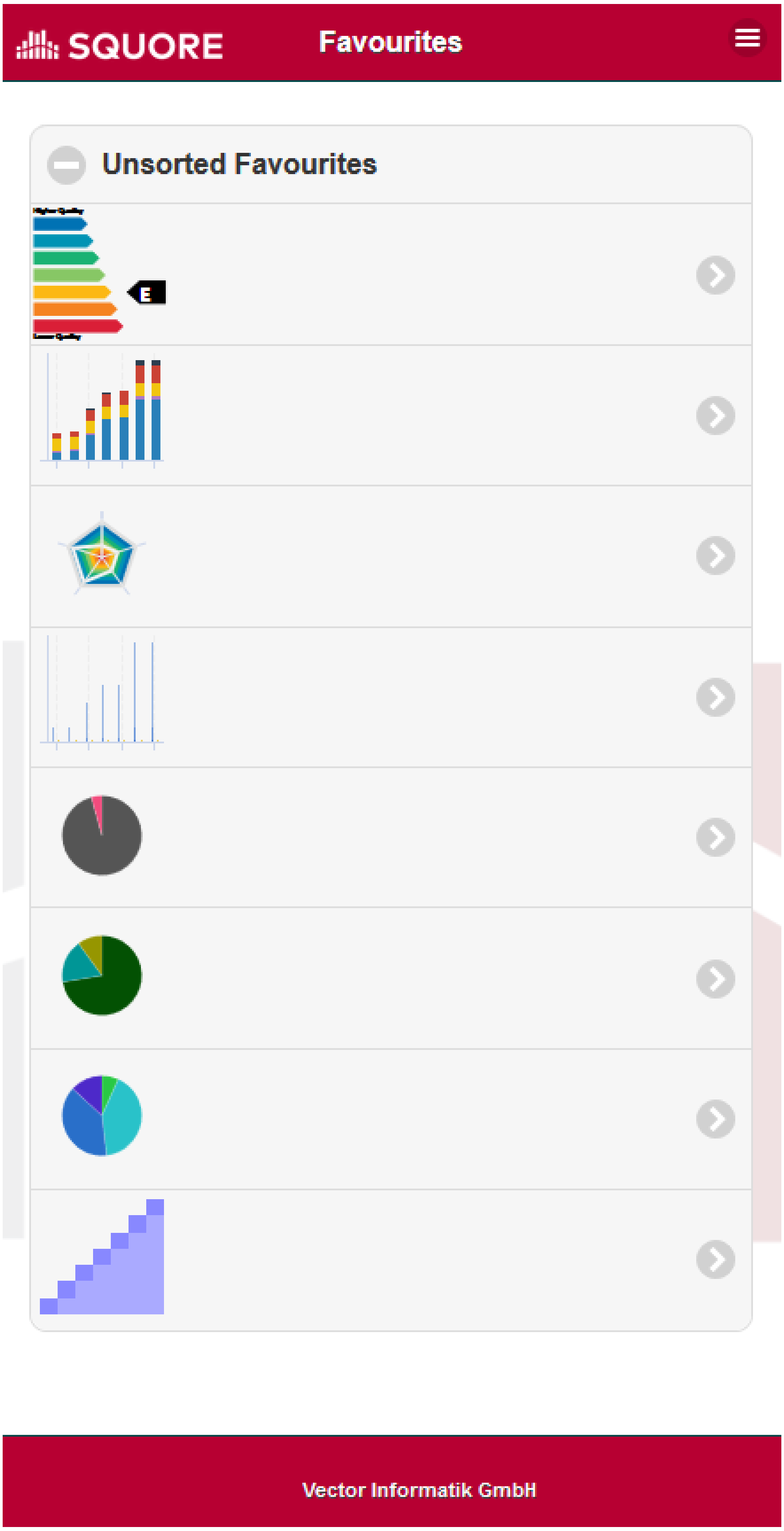
Comments:

The full Favourites page

Squore Mobile

The list of charts you marked as favourites in Squore is the list of charts you can access via Squore Mobile

Squore Mobile is a touch-friendly interface for Squore that is accessible from http://localhost:8180/SQuORE_Server/Mobile.



Squore Mobile

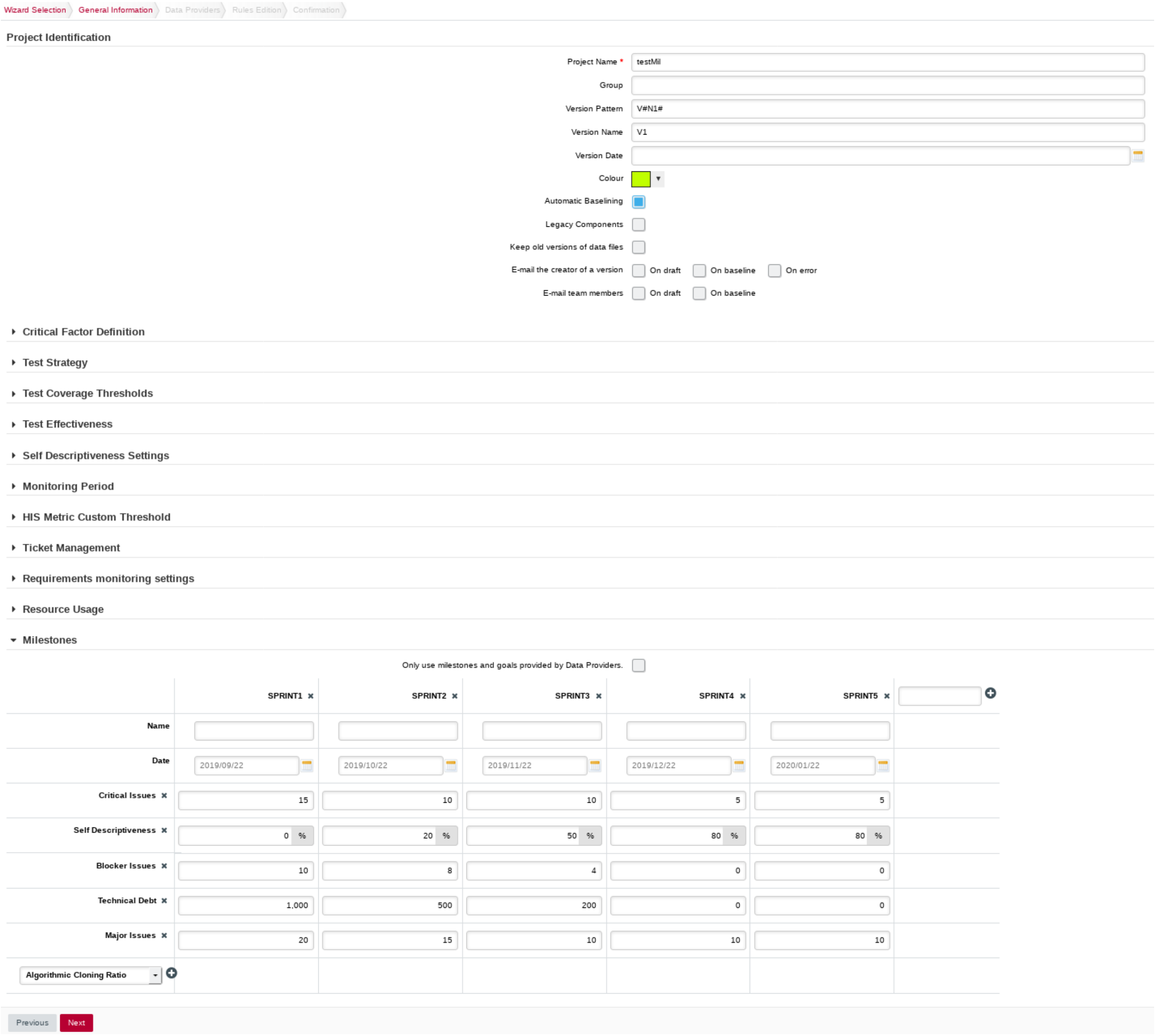
When you log into Squore Mobile, you can swipe through all the charts you added to your favourite lists from your mobile device.

Chapter 9. Focus on Your Milestones

Squore allows tracking your progress by setting milestones, which consist of a series of goals for specific metrics at certain dates in the life of your project. In this chapter, you will learn how to set up these goals and how to read dashboard charts that show deviations from these goals or changes in your project milestones.

Setting up Goals

Not all models support milestones, but if yours does, you will see a Milestones pane on the first page of the project wizard. The Milestones pane is where you can see the existing milestones for a project, with their associated goals and dates.



Project Identification

Project Name * testMil

Group

Version Pattern V#N1#

Version Date

Version Name V1

Colour 🟩

Automatic Baselining ☑

Legacy Components □

Keep old versions of data files □

E-mail the creator of a version □ On draft □ On baseline □ On error

E-mail team members □ On draft □ On baseline

- Critical Factor Definition
- Test Strategy
- Test Coverage Thresholds
- Test Effectiveness
- Self Descriptiveness Settings
- Monitoring Period
- HIS Metric Custom Threshold
- Ticket Management
- Requirements monitoring settings
- Resource Usage
- Milestones

Only use milestones and goals provided by Data Providers. □

	SPRINT1 ✖	SPRINT2 ✖	SPRINT3 ✖	SPRINT4 ✖	SPRINT5 ✖
Name					
Date	2019/09/22	2019/10/22	2019/11/22	2019/12/22	2020/01/22
Critical Issues ✖	15	10	10	5	5
Self Descriptiveness ✖	0 %	20 %	50 %	80 %	80 %
Blocker Issues ✖	10	8	4	0	0
Technical Debt ✖	1,000	500	200	0	0
Major Issues ✖	20	15	10	10	10
Algorithmic Cloning Ratio					

The Milestones pane in the project wizard

In the example above, our model defines 5 milestones (SPRINT1 to SPRINT5) for the lifecycle of our project.

Each milestone has a set date and defines goals for the following key performance indicators in our project:

- **Blocker Issues**
- **Technical Debt**
- **Self Descriptiveness**
- **Major Issues**
- **Critical Issues**
- **Coding Standard Compliance**

The Milestones pane allows you to change the dates and goals for your project. If a milestone is optional and is not relevant for your project, you can remove it by clicking the **x** next to its name. This is possible for all the milestones in our example above. By clicking the **+** icon to the right of the last milestone, you can create a new milestone for the project and define your own goals. You can also add a new goal for your project by selecting a metric from the list at the bottom of the table and clicking the **\+** icon.

When you are satisfied with the milestones set for your project, click the Next button to continue with the creation of the project.

Goals and dates can be modified every time you create a new version of the project if you decide that your schedule slips. Goals and dates are versioned, so your dashboard can always show you when in the timeline of your project you decided to change your milestones.

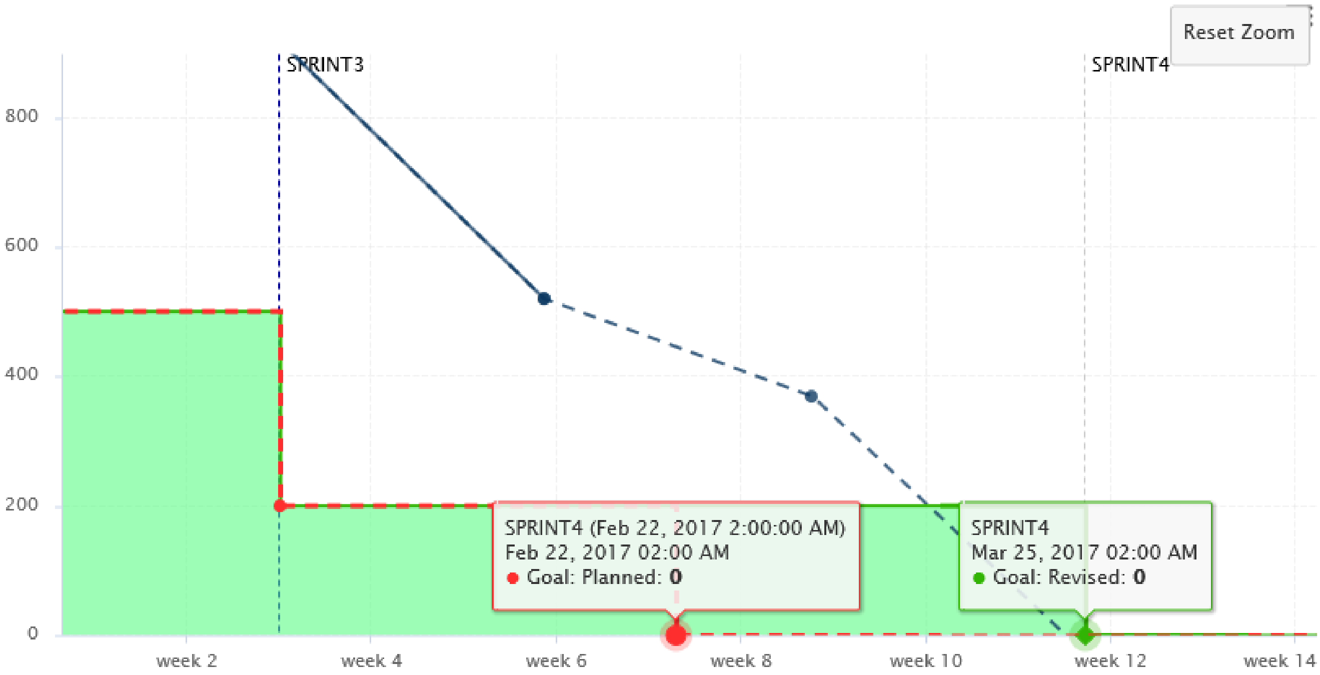
> In order to avoid possible conflicts between the User Interface and the Data Provider, please note that you can focus the milestones creation with the Data Provider by clicking the checkbox "Only use milestones and goals provided by Data Provider. Checking this box hides the milestones creation section of the wizard user interface.

Milestones on your Dashboard

When you consult the dashboard of a project that uses milestones, the functionality allows you to:

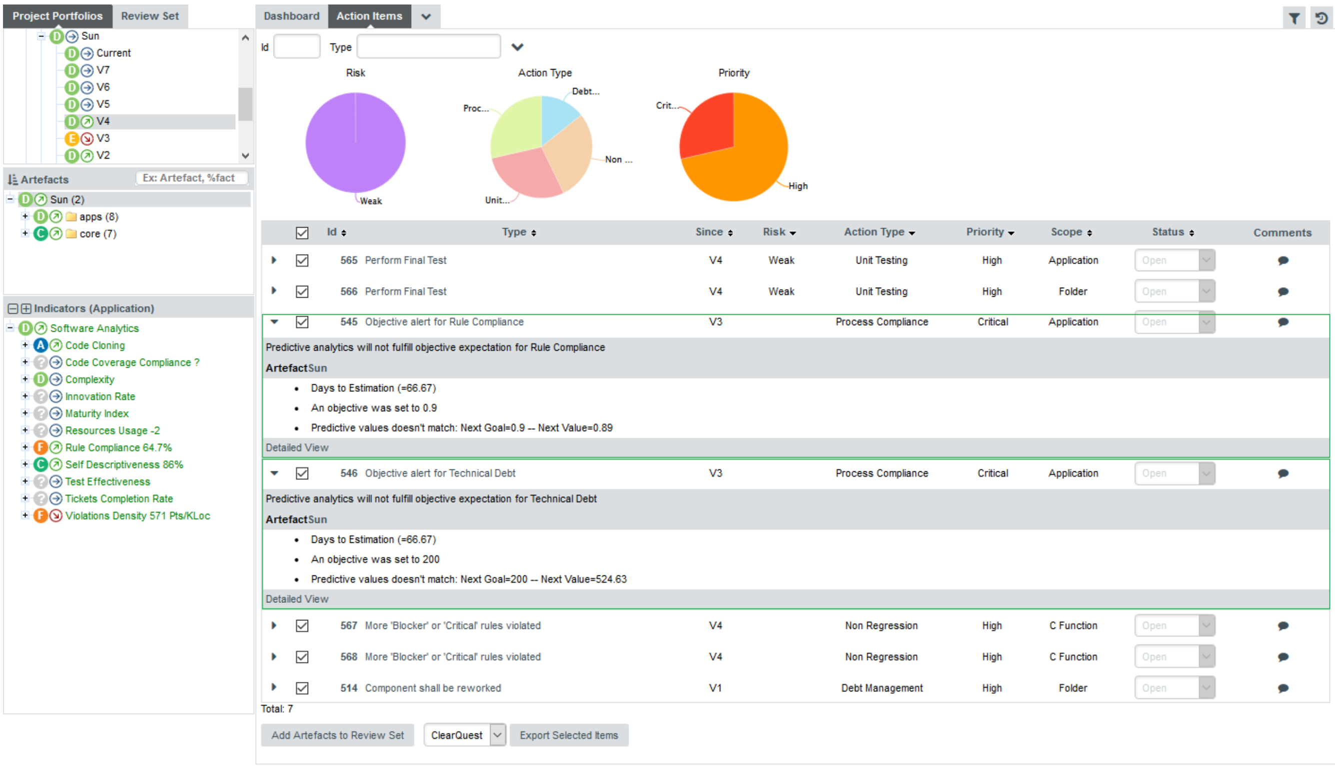
- Display the goals defined for each milestone in your project
- Display the changes made to the goals defined for each milestone
- Display the date changes for your milestones
- Show markers for milestone dates and goals

The following is an example of a chart that mixes objectives, projected performance and milestone date changes:



A chart tracking your technical debt progress, projected performance, goals and milestone date slips

Some action items on your model can also take advantage of this feature to warn you about poor performance:



☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
☑	565	Perform Final Test	V4	Weak	Unit Testing	High	Application	Open	
☑	566	Perform Final Test	V4	Weak	Unit Testing	High	Folder	Open	
☑	545	Objective alert for Rule Compliance	V3		Process Compliance	Critical	Application	Open	

Predictive analytics will not fulfill objective expectation for Rule Compliance

ArtefactSun

- Days to Estimation (=66.67)
- An objective was set to 0.9
- Predictive values doesn't match: Next Goal=0.9 -- Next Value=0.89

Detailed View

☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
☑	546	Objective alert for Technical Debt	V3		Process Compliance	Critical	Application	Open	

Predictive analytics will not fulfill objective expectation for Technical Debt

ArtefactSun

- Days to Estimation (=66.67)
- An objective was set to 200
- Predictive values doesn't match: Next Goal=200 -- Next Value=524.63

Detailed View

☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
☑	567	More 'Blocker' or 'Critical' rules violated	V4		Non Regression	High	C Function	Open	
☑	568	More 'Blocker' or 'Critical' rules violated	V4		Non Regression	High	C Function	Open	
☑	514	Component shall be reworked	V1		Debt Management	High	Folder	Open	

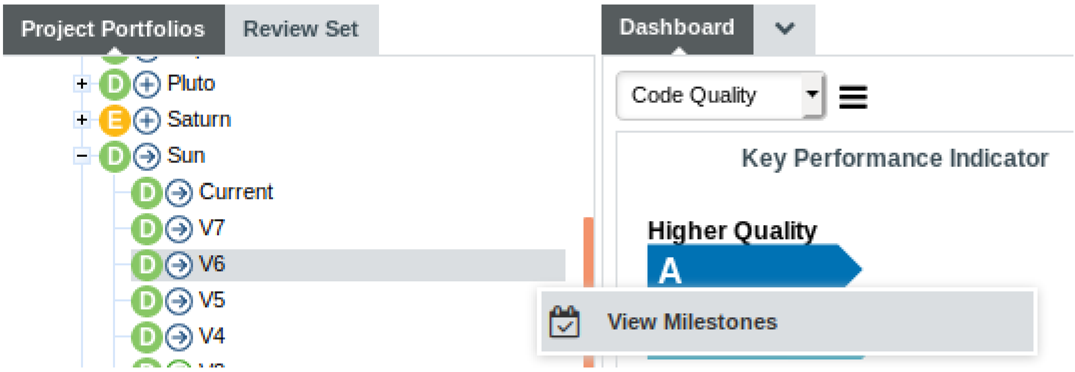
Total: 7

Action items based on milestone dates and goals

For more information on how you can improve your model with milestones and the above chart and action items, refer to Milestones Tutorial and the Configuration Guide.

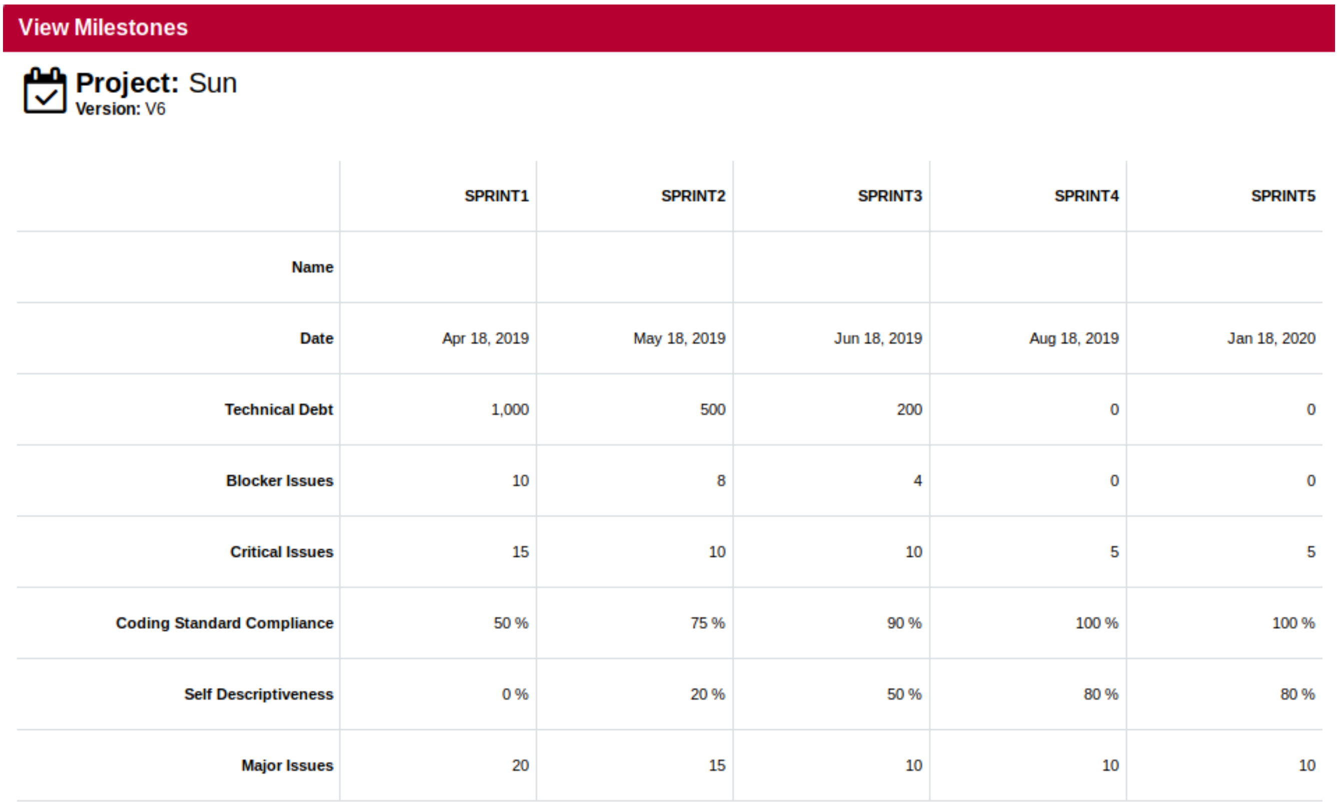
Milestones in Project Portfolios

You can also visualize the milestones setup for each version of your project in the Project Portfolios :



Visualizing project's versions milestones via sandwich menu in Project Portfolios

A pop-up shows you the milestones definition at version creation :



View Milestones

Project: Sun
Version: V6

	SPRINT1	SPRINT2	SPRINT3	SPRINT4	SPRINT5
Name					
Date	Apr 18, 2019	May 18, 2019	Jun 18, 2019	Aug 18, 2019	Jan 18, 2020
Technical Debt	1,000	500	200	0	0
Blocker Issues	10	8	4	0	0
Critical Issues	15	10	10	5	5
Coding Standard Compliance	50 %	75 %	90 %	100 %	100 %
Self Descriptiveness	0 %	20 %	50 %	80 %	80 %
Major Issues	20	15	10	10	10

Project's version milestones pop-up

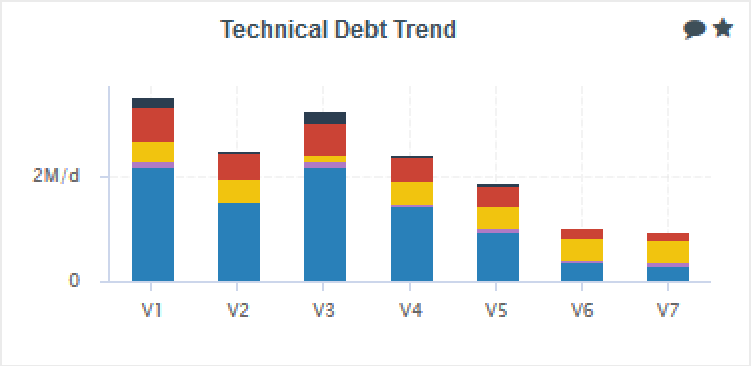
Chapter 10. Communicating With Squore

Comments and Notifications

Squore allows posting comments about charts, artefacts, action items and findings. Users in a project team can view and reply to comments when they notice that a discussion thread has received new posts since their last visit. You can also choose when a discussion no longer accepts comments or is removed from the project. In this section, you will learn the basics of commenting all around the dashboard.

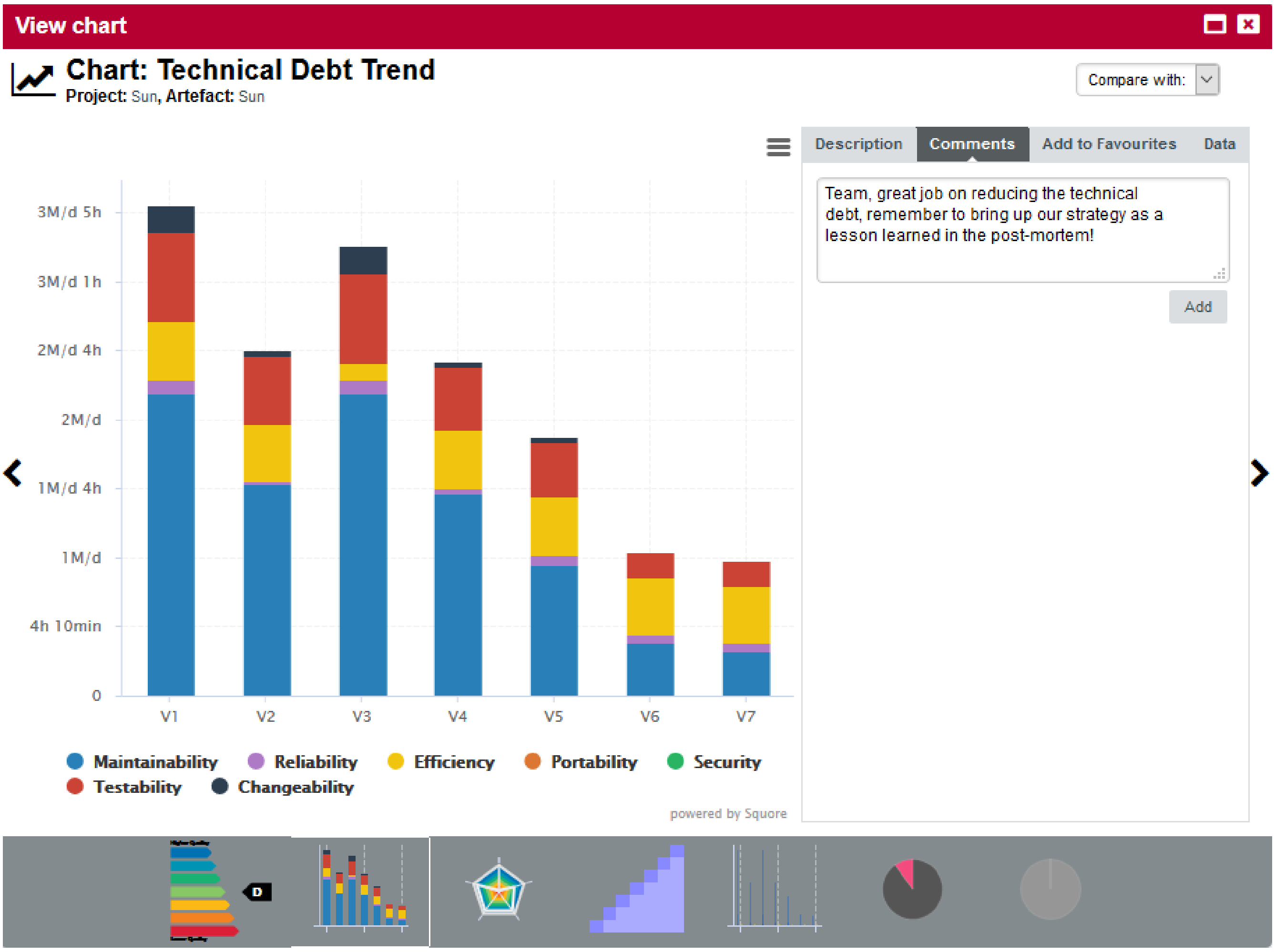
Commenting Charts

Every chart on your dashboard shows a speech bubble icon next to its title, as shown in the picture below:



A chart thumbnail showing a speech bubble icon

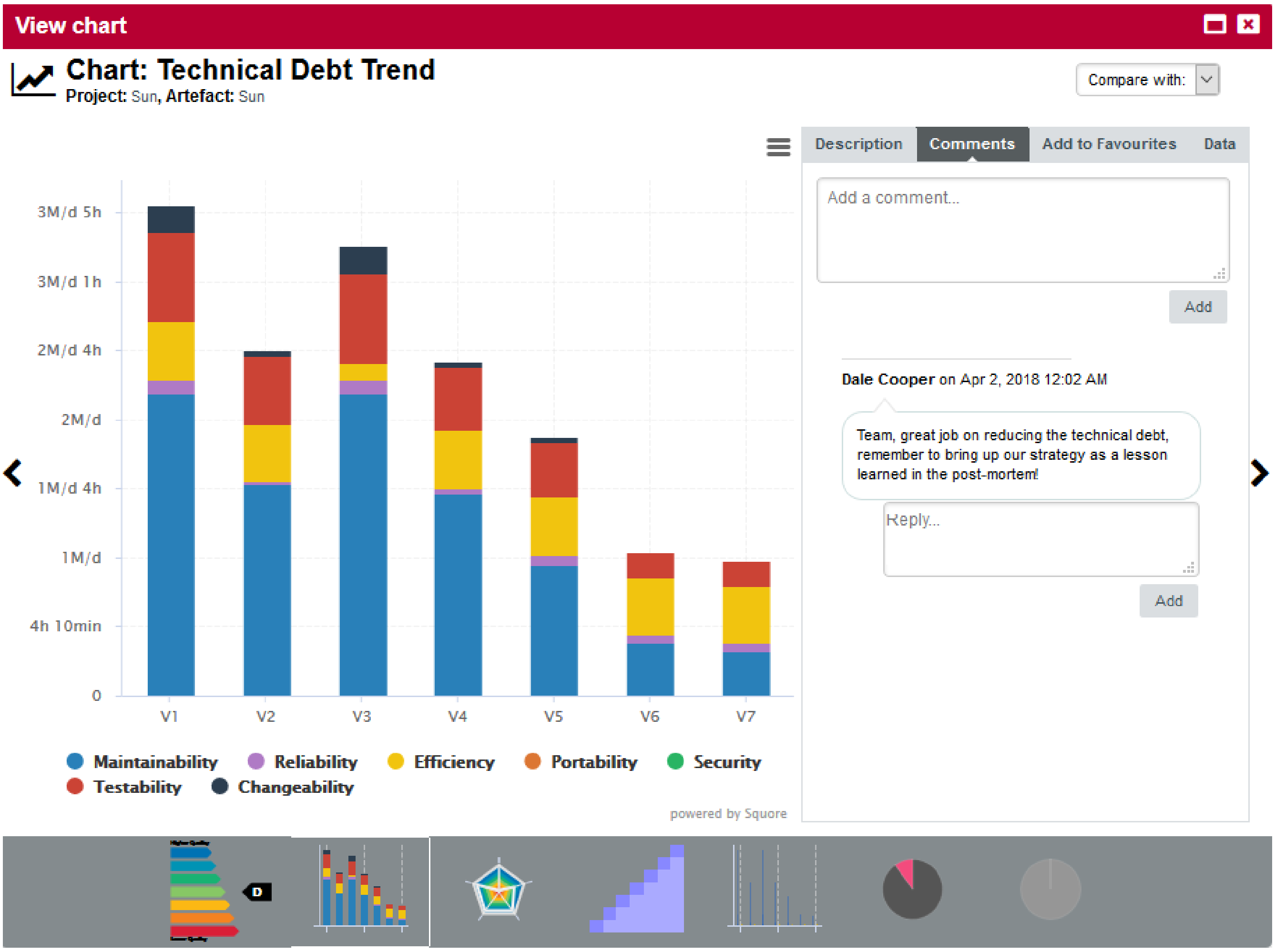
Clicking the icon brings up the Chart Viewer on the comment tab, in which you can confirm which chart you are commenting on and type your comment.



Team, great job on reducing the technical debt, remember to bring up our strategy as a lesson learned in the post-mortem!

Typing a comment in the chart viewer

When you click **Add**, your comment is saved, and you can reply or add another comment.

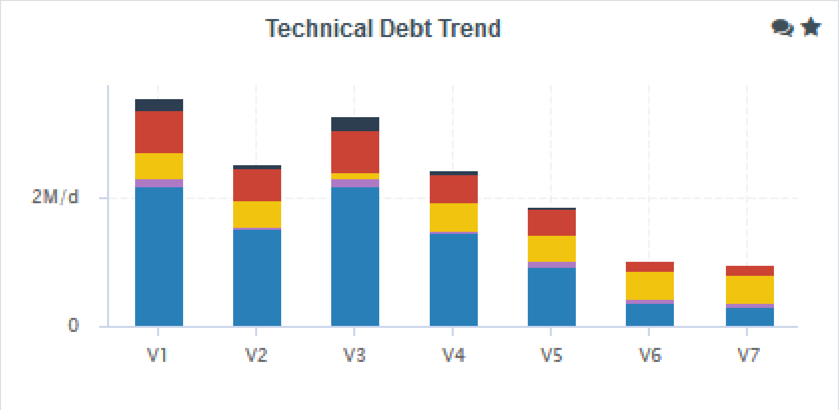


Dale Cooper on Apr 2, 2018 12:02 AM

Team, great job on reducing the technical debt, remember to bring up our strategy as a lesson learned in the post-mortem!

The Comment pop-up after adding your first comment

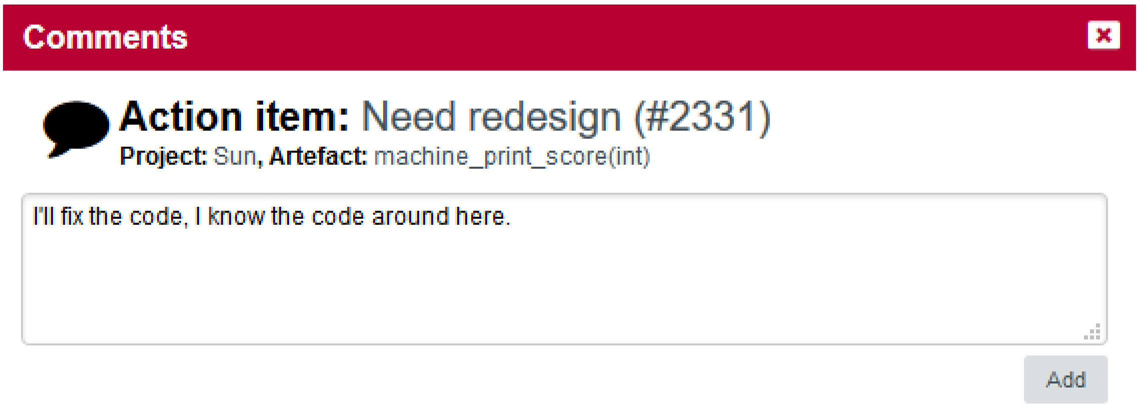
When you close the pop-up, notice that the speech bubble icon next to the chart you commented changes to indicate that a discussion about this item has been started.



A chart thumbnail with a discussion indicator

Commenting Action Items

In the Action Items tab, you bring up the comment pop up by clicking the speech bubble in the **Comments** column of the table. Links allow you to jump to the Action Item's detailed description, the application level dashboard or the artefact dashboard directly.



Action item: Need redesign (#2331)

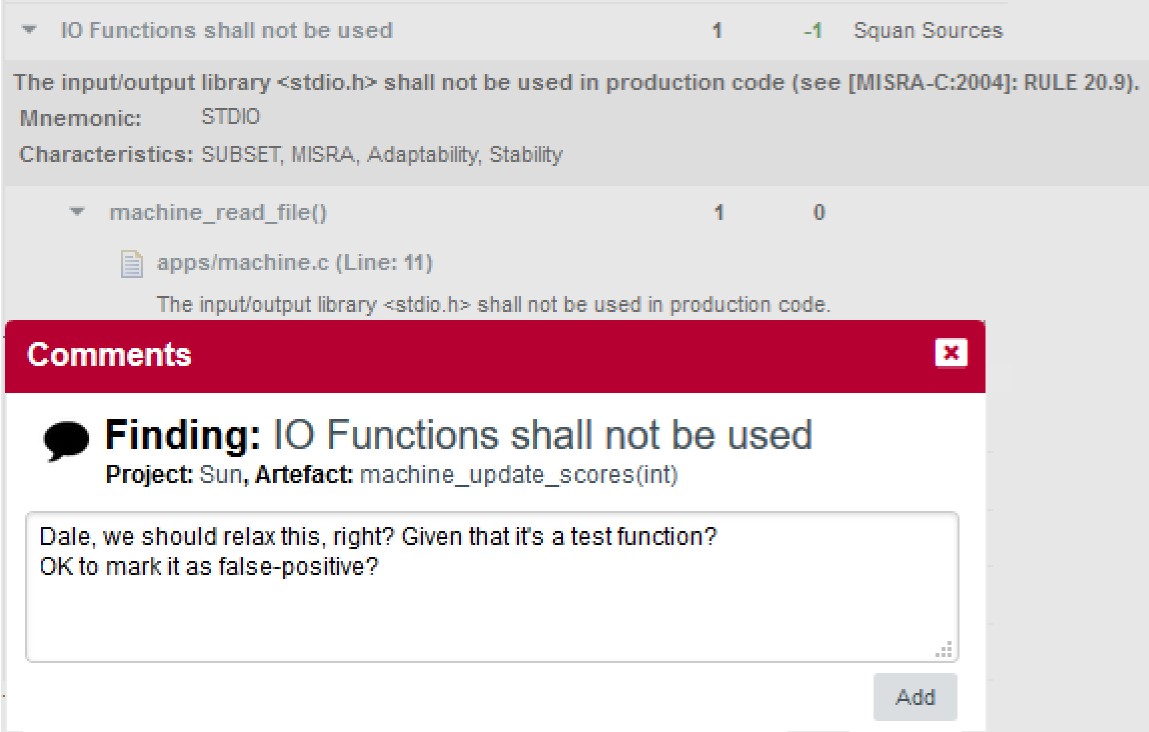
Project: Sun, **Artefact:** machine_print_score(int)

I'll fix the code, I know the code around here.

A comment thread initiated about an Action Item

Commenting Findings

On the Findings tab, each violation shows a comment icon that you can click to start a discussion.



IO Functions shall not be used 1 -1 Squan Sources

The input/output library <stdio.h> shall not be used in production code (see [MISRA-C:2004]: RULE 20.9).

Mnemonic: STDIO

Characteristics: SUBSET, MISRA, Adaptability, Stability

machine_read_file() 1 0

apps/machine.c (Line: 11)

The input/output library <stdio.h> shall not be used in production code.

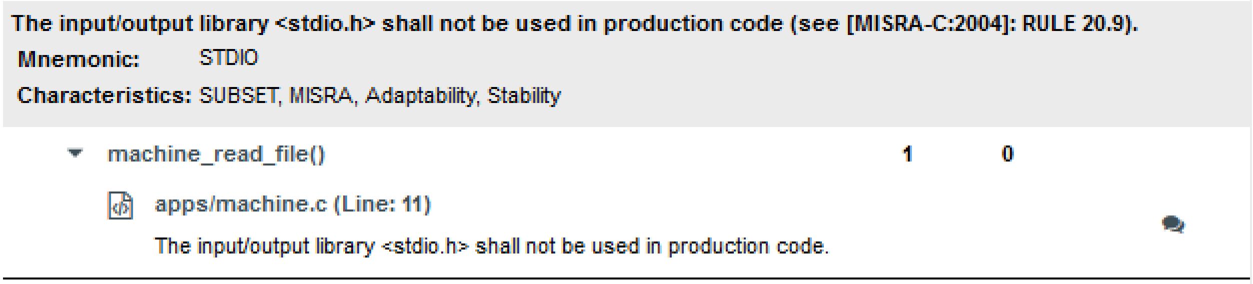
Finding: IO Functions shall not be used

Project: Sun, **Artefact:** machine_update_scores(int)

Dale, we should relax this, right? Given that it's a test function?
OK to mark it as false-positive?

A comment thread initiated about an finding

As on the Dashboard and Action Items tabs, the comment icon indicates whether a discussion has been started about a violation.



The input/output library <stdio.h> shall not be used in production code (see [MISRA-C:2004]: RULE 20.9).

Mnemonic: STDIO

Characteristics: SUBSET, MISRA, Adaptability, Stability

machine_read_file() 1 0

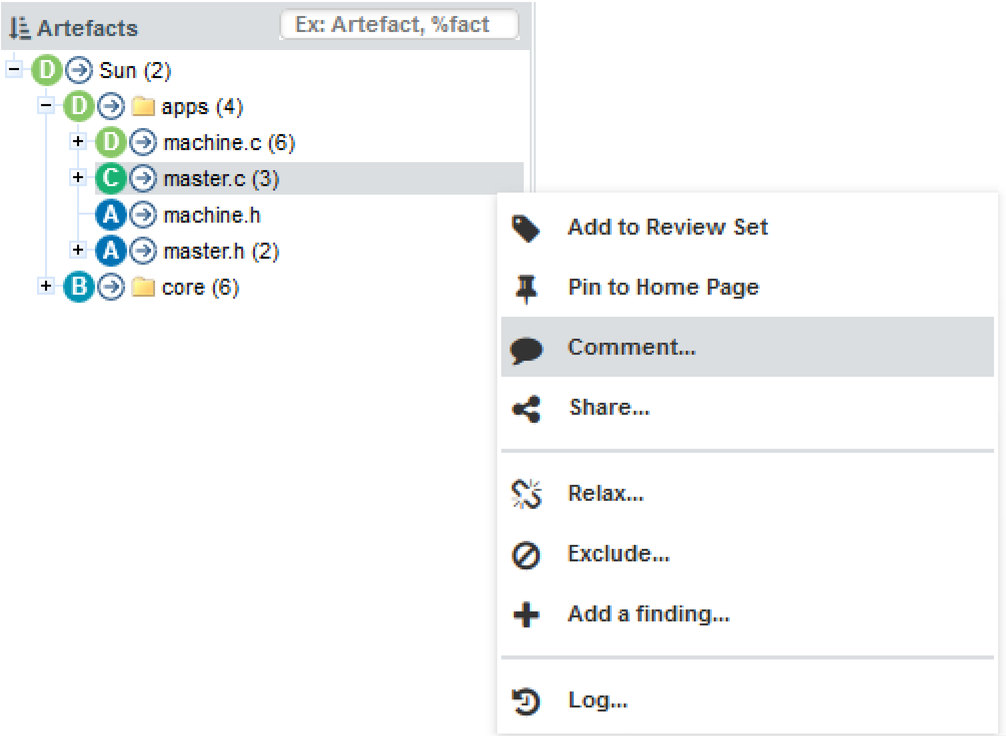
apps/machine.c (Line: 11)

The input/output library <stdio.h> shall not be used in production code.

An ongoing discussion on the Findings tab

Commenting From the Artefact Tree

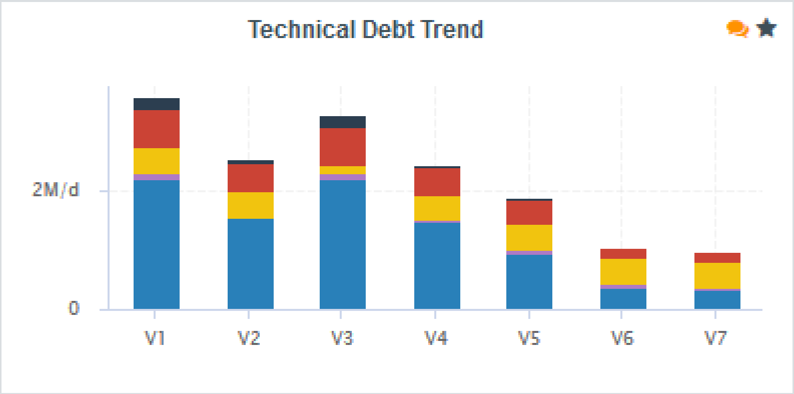
You can start discussions on artefacts by clicking **Comment...** item in the artefact action menu, as shown below:



The artefact context menu item to bring up the comment pop-up for artefacts

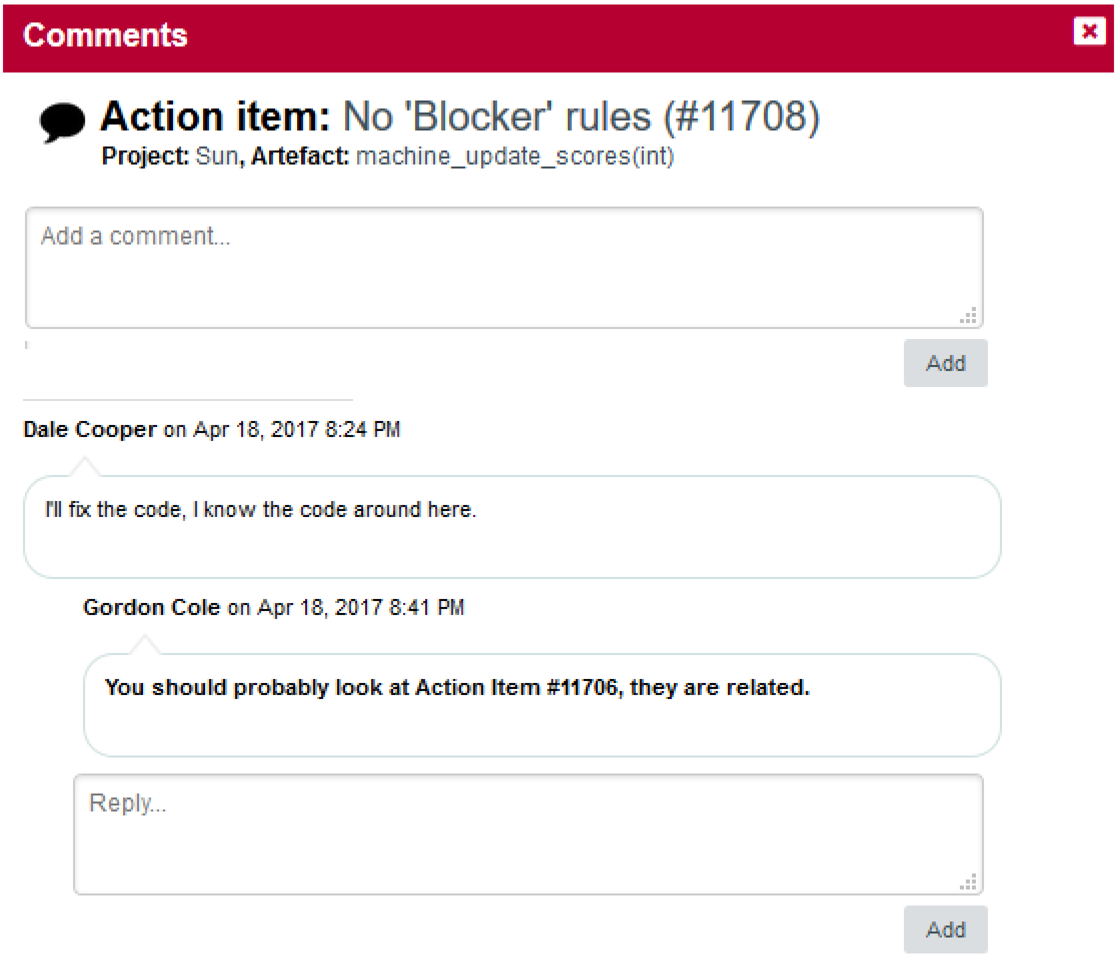
Following Discussions

When you log in to Squore, you can find out which discussions have new comments by looking at the items that show the **New Comment!** icon:



You have unread comments in the discussion about this chart

In the discussion pop-up, new comments since your last visit are highlighted:



Action item: No 'Blocker' rules (#11708)

Project: Sun, **Artefact:** machine_update_scores(int)

Dale Cooper on Apr 18, 2017 8:24 PM

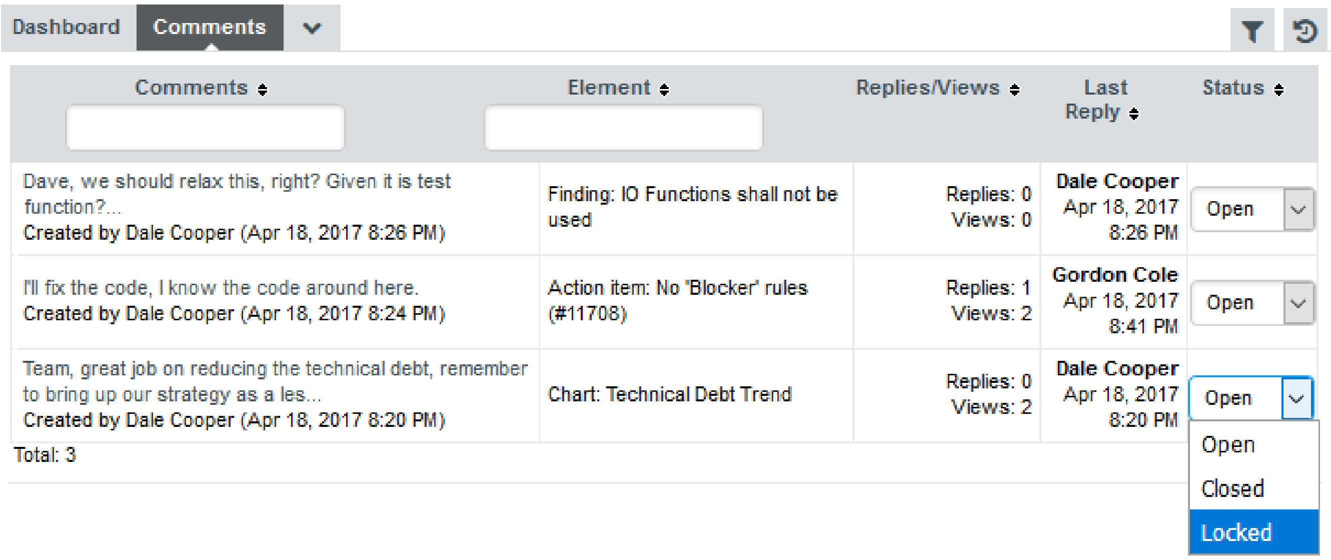
I'll fix the code, I know the code around here.

Gordon Cole on Apr 18, 2017 8:41 PM

You should probably look at Action Item #11706, they are related.

A reply to my comment appears in bold

You can also get an exhaustive view of all the discussions for the project by viewing them in the **Comments** tab in the Explorer:



Comments	Element	Replies/Views	Last Reply	Status
Dave, we should relax this, right? Given it is test function?... Created by Dale Cooper (Apr 18, 2017 8:26 PM)	Finding: IO Functions shall not be used	Replies: 0 Views: 0	**Dale Cooper** Apr 18, 2017 8:26 PM	Open
I'll fix the code, I know the code around here. Created by Dale Cooper (Apr 18, 2017 8:24 PM)	Action item: No 'Blocker' rules (#11708)	Replies: 1 Views: 2	**Gordon Cole** Apr 18, 2017 8:41 PM	Open
Team, great job on reducing the technical debt, remember to bring up our strategy as a les... Created by Dale Cooper (Apr 18, 2017 8:20 PM)	Chart: Technical Debt Trend	Replies: 0 Views: 2	**Dale Cooper** Apr 18, 2017 8:20 PM	Open

Total: 3

Overview of discussions about a project in the Comments tab

From this view, discussions can be set to one the following statuses:

- **Open:** New comments are accepted in this discussion
- **Closed:** No new comments are accepted in this discussion, and it will be deleted in the next analysis
- **Locked:** No new comments are accepted in this discussion, but the discussion thread will be

saved for the next analyses

> Any user in the project team can view and take part in all open conversations in the project.

Linking to Projects

There are two ways to build direct links to projects in Squore:

- using IDs with the **RestoreContext** page
- using names with the **LoadDashboard** page

Each method supports different parameters to build direct links to a tab of the Explorer for the specified project, as explained below.

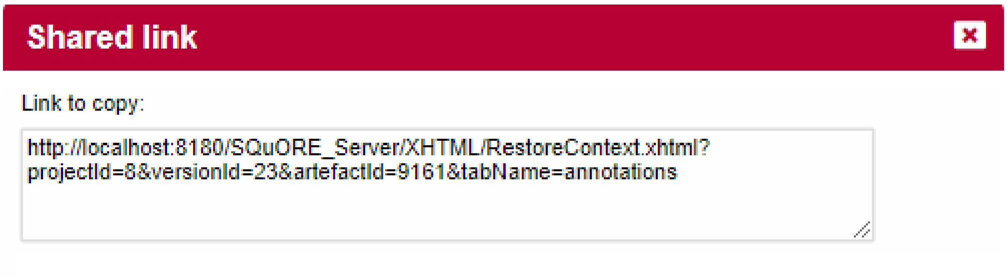
RestoreContext

http://localhost:8180/SQuORE_Server/XHTML/RestoreContext.xhtml

Links to the Squore Explorer using IDs. The URL accepts the following parameters:

- **modelId** to link to a model node in the portfolio
- **projectId** to link to the latest version of the specified project
- **versionId** to link to a specific version of a project
- **artefactId** to link to a specific artefact in a project (can be combined with projectId and versionId)
- **tabName** to display a specific tab of the Explorer. This parameter can be combined with any of the ones above and must be one of:
    - dashboard (default)
    - action-items
    - highlights
    - findings
    - documents
    - attributes (Forms)
    - indicators
    - measures
    - annotations

Users can copy a RestoreContext link from the Home page, the Projects page, or generate one using the **Share...** dialog in an artefact's context menu, which is the only way to find an artefactId. Model IDs are not exposed anywhere in the web interface.



Link to copy:

```
http://localhost:8180/SQuORE_Server/XHTML/RestoreContext.xhtml?
projectId=8&versionId=23&artefactId=9161&tabName=annotations
```

The sharing dialog from the web UI with a full RestoreContext URL

Project and version IDs are printed in the project's output XML file, making it easy to parse and build a URL dynamically when using continuous integration to launch analyses.

LoadDashboard

http://localhost:8180/SQuORE_Server/XHTML/MyDashboard/dashboard/LoadDashboard.xhtml

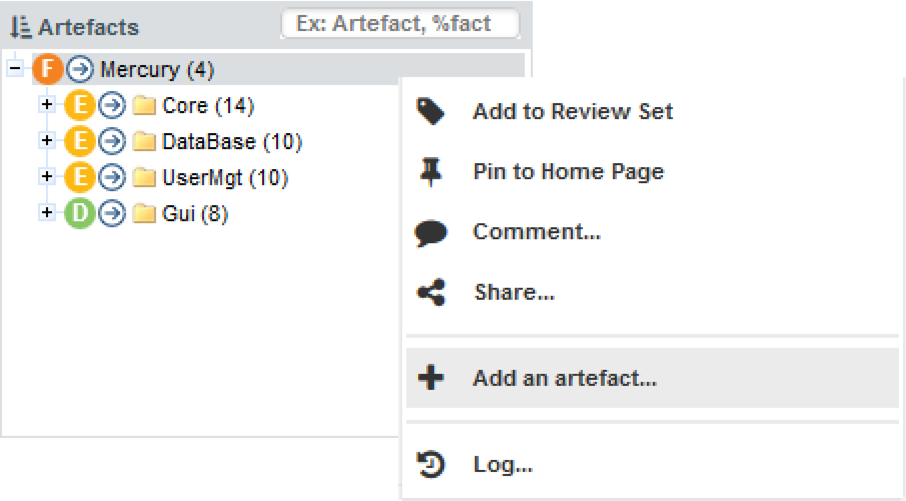
Links to the Squore Explorer using names instead of IDs. The URL accepts the following parameters:

- **application (mandatory)** to specify the project to link to
- **version (optional)** to specify which version of the project to display. When not specified, the latest version fo the project is displayed
- **artefactId (optional)** to link to a specific artefact in the project
- **tabName** to display a specific tab of the Explorer. This parameter can be combined with any of the ones above and must be one of:
    - dashboard (default)
    - action-items
    - highlights
    - findings
    - documents
    - attributes (Forms)
    - indicators
    - measures
    - annotations

The following is a URL that links to the version called V5 in the project called Earth. Since no artefactId and tabName are specified, the Dashboard tab will be displayed for the root node of the project: http://localhost:8180/SQuORE_Server/XHTML/MyDashboard/dashboard/LoadDashboard.xhtml?application=Earth&version=V5.

Adding and Removing Artefacts Manually

While you review results and comments, you can add artefacts manually to your project as needed. To add an artefact, make sure you are on the Current version of the project and click the node to which you want to add a child artefact. If this node supports adding artefacts, the Add an Artefact option will be available in the menu:

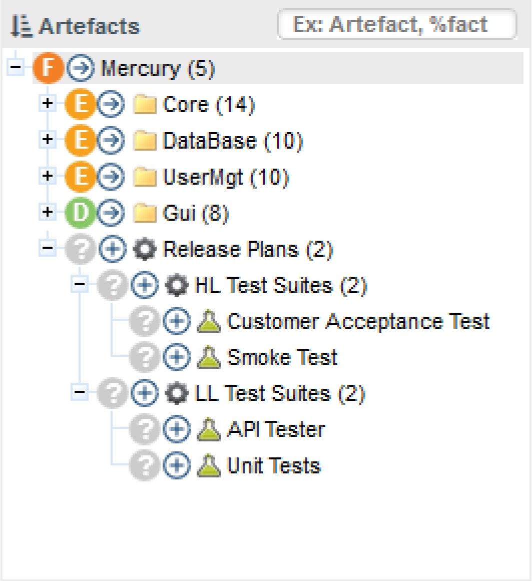


The artefact context menu with the Add an Artefact option highlighted

Click the menu and choose an artefact type and an artefact name to add the artefact to the tree.

The type of artefact you can add depends on the model you are using. The model also defines where in the tree the new artefacts can be added.

Here is what the Artefact Tree looks like after manually building a test plan tree:



The artefact tree with manual artefacts not yet rated

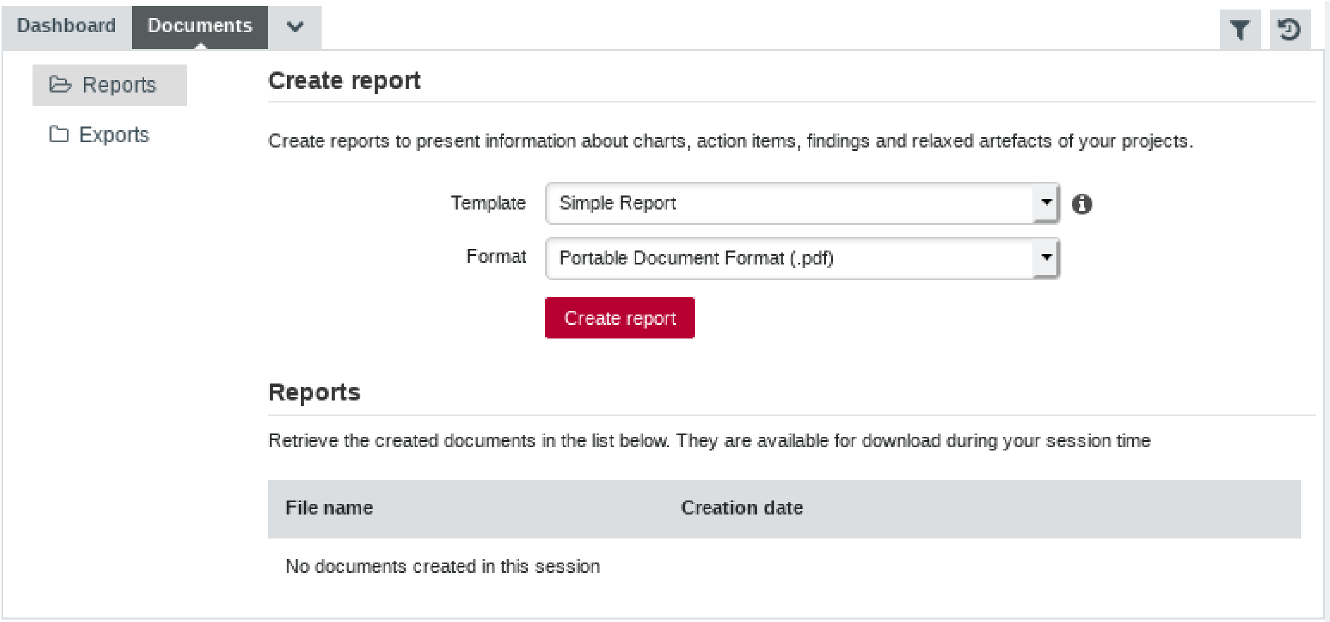
When you run a new analysis of the project, the new artefacts will get rated according to what is defined in your model.

Artefacts that were added manually can also be deleted from the tree. Note that artefact edition is tied to a permission in a user's project role within a project. To learn more about project roles, refer to User Project Roles

Reporting Project Status

You can generate reports and export data to csv with Squore so you can communicate your progress to others.

In order to create a report, for the currently selected artefact in the tree, click the **Documents** tab in the Explorer. The Documents page opens as shown in the picture below:



Dashboard | Documents

Reports

Exports

Create report

Create reports to present information about charts, action items, findings and relaxed artefacts of your projects.

Template: Simple Report

Format: Portable Document Format (.pdf)

Create report

Reports

Retrieve the created documents in the list below. They are available for download during your session time

File name	Creation date
No documents created in this session	

The Documents tab

The Documents tab offers a choice of report and export types in various output formats. This choice between document types is made on the left part of the tab, by clicking on one of the two following icons:

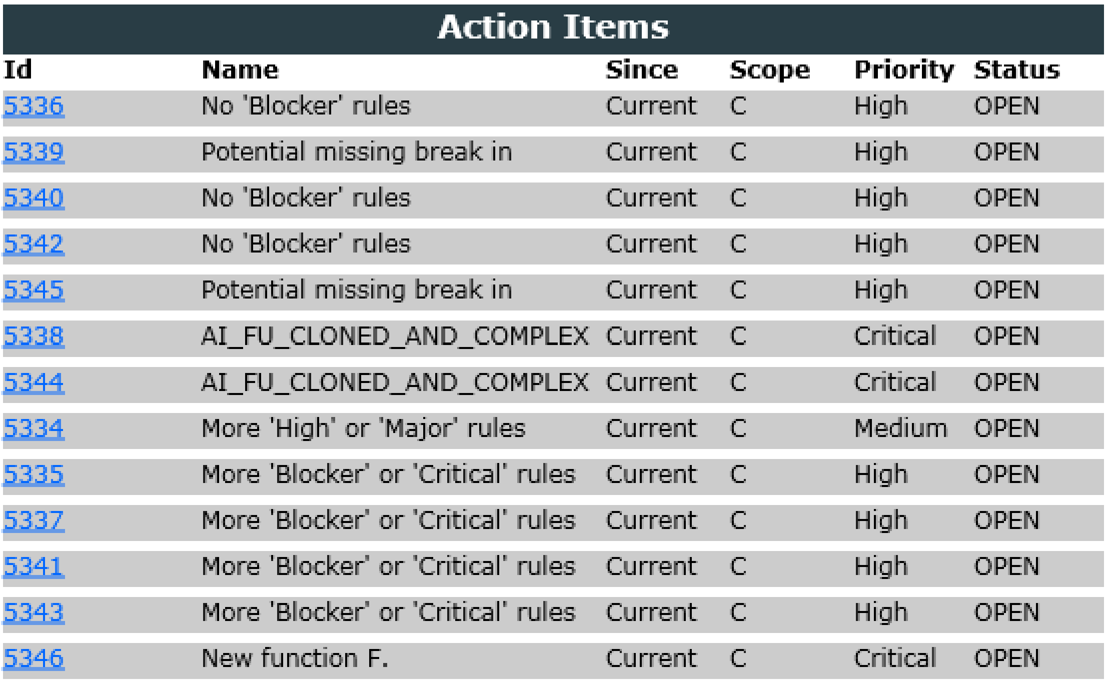
Reports

Exports

Document types list

Reports are primarily used to present information visually including charts and data about action items, findings and relaxed artefacts. The output formats currently supported are .pdf, .pptx, .docx.

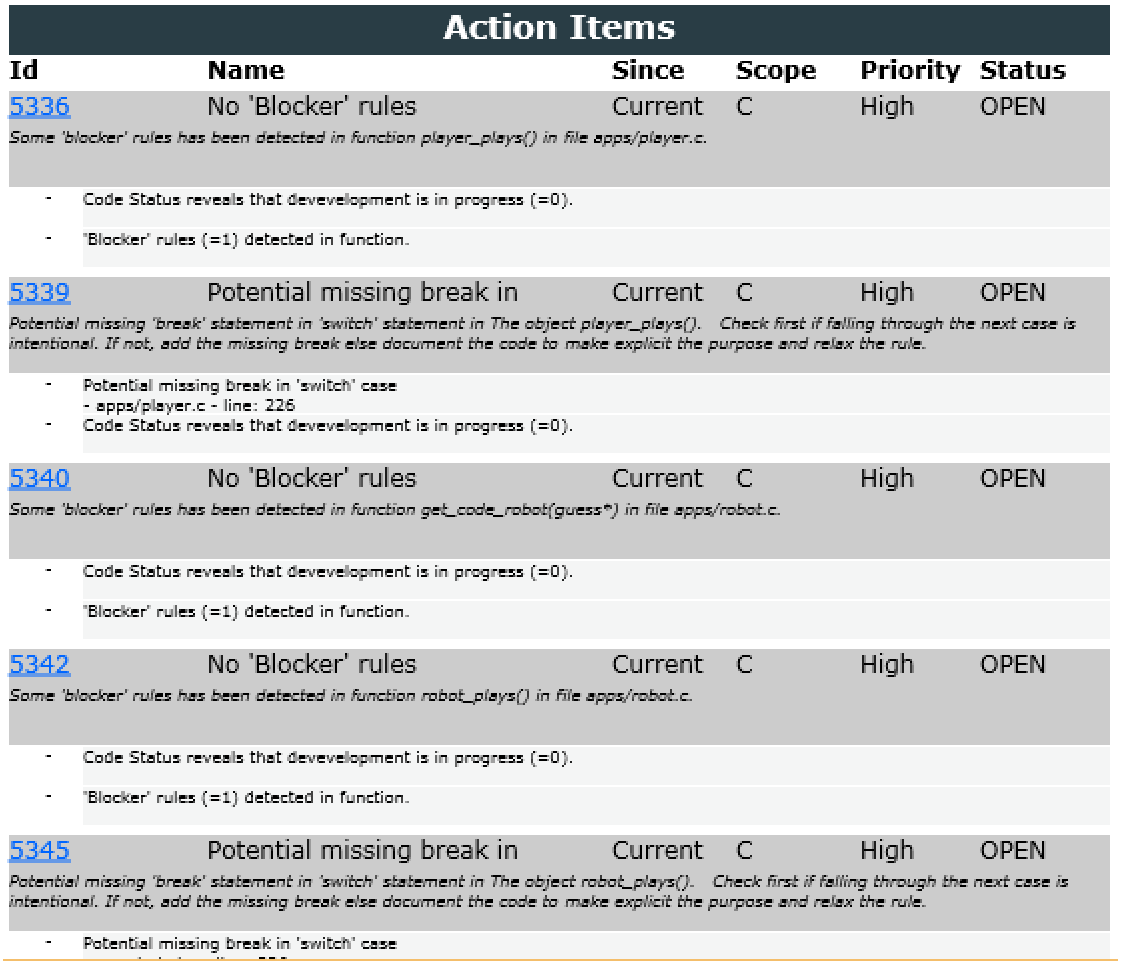
Depending to your licence options, reports can be created in full or synthetic view. The synthetic view omits details about the exact location of violations in the code in the report, but provide a link to read the full details in the Squore web interface. The screenshots below show the difference between a synthetic report and a full report.



Action Items

Id	Name	Since	Scope	Priority	Status
5336	No 'Blocker' rules	Current	C	High	OPEN
5339	Potential missing break in	Current	C	High	OPEN
5340	No 'Blocker' rules	Current	C	High	OPEN
5342	No 'Blocker' rules	Current	C	High	OPEN
5345	Potential missing break in	Current	C	High	OPEN
5338	AI_FU_CLONED_AND_COMPLEX	Current	C	Critical	OPEN
5344	AI_FU_CLONED_AND_COMPLEX	Current	C	Critical	OPEN
5334	More 'High' or 'Major' rules	Current	C	Medium	OPEN
5335	More 'Blocker' or 'Critical' rules	Current	C	High	OPEN
5337	More 'Blocker' or 'Critical' rules	Current	C	High	OPEN
5341	More 'Blocker' or 'Critical' rules	Current	C	High	OPEN
5343	More 'Blocker' or 'Critical' rules	Current	C	High	OPEN
5346	New function F.	Current	C	Critical	OPEN

Details about action items in a synthetic report



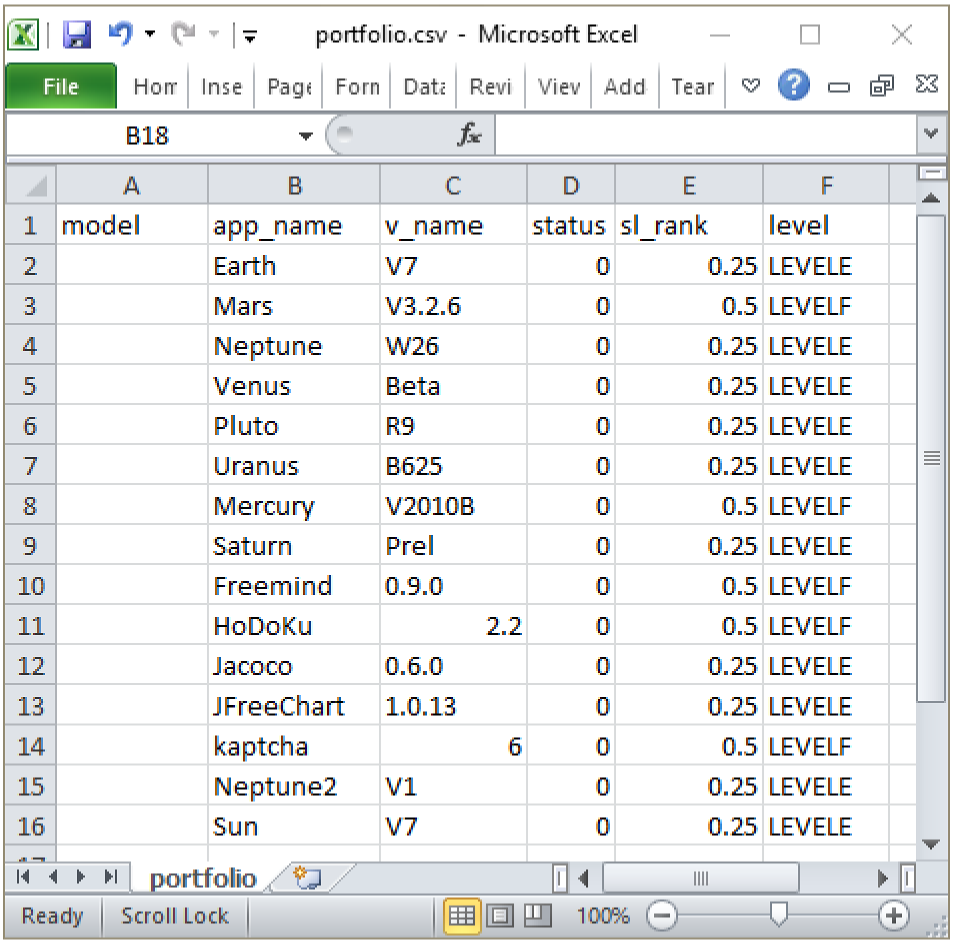
Action Items

<table>
<tr><th>Id</th><th>Name</th><th>Since</th><th>Scope</th><th>Priority</th><th>Status</th></tr>
<tr><td>5336</td><td>No 'Blocker' rules</td><td>Current</td><td>C</td><td>High</td><td>OPEN</td></tr>
<tr><td colspan="6"><i>Some 'blocker' rules has been detected in function player_plays() in file apps/player.c.</i></td></tr>
<tr><td>-</td><td colspan="5">Code Status reveals that devevelopment is in progress (=0).</td></tr>
<tr><td>-</td><td colspan="5">'Blocker' rules (=1) detected in function.</td></tr>
<tr><td>5339</td><td>Potential missing break in</td><td>Current</td><td>C</td><td>High</td><td>OPEN</td></tr>
<tr><td colspan="6"><i>Potential missing 'break' statement in 'switch' statement in The object player_plays().   Check first if falling through the next case is intentional. If not, add the missing break else document the code to make explicit the purpose and relax the rule.</i></td></tr>
<tr><td>-</td><td colspan="5">Potential missing break in 'switch' case
- apps/player.c - line: 226</td></tr>
<tr><td>-</td><td colspan="5">Code Status reveals that devevelopment is in progress (=0).</td></tr>
<tr><td>5340</td><td>No 'Blocker' rules</td><td>Current</td><td>C</td><td>High</td><td>OPEN</td></tr>
<tr><td colspan="6"><i>Some 'blocker' rules has been detected in function get_code_robot(guess*) in file apps/robot.c.</i></td></tr>
<tr><td>-</td><td colspan="5">Code Status reveals that devevelopment is in progress (=0).</td></tr>
<tr><td>-</td><td colspan="5">'Blocker' rules (=1) detected in function.</td></tr>
<tr><td>5342</td><td>No 'Blocker' rules</td><td>Current</td><td>C</td><td>High</td><td>OPEN</td></tr>
<tr><td colspan="6"><i>Some 'blocker' rules has been detected in function robot_plays() in file apps/robot.c.</i></td></tr>
<tr><td>-</td><td colspan="5">Code Status reveals that devevelopment is in progress (=0).</td></tr>
<tr><td>-</td><td colspan="5">'Blocker' rules (=1) detected in function.</td></tr>
<tr><td>5345</td><td>Potential missing break in</td><td>Current</td><td>C</td><td>High</td><td>OPEN</td></tr>
<tr><td colspan="6"><i>Potential missing 'break' statement in 'switch' statement in The object robot_plays().   Check first if falling through the next case is intentional. If not, add the missing break else document the code to make explicit the purpose and relax the rule.</i></td></tr>
<tr><td>-</td><td colspan="5">Potential missing break in 'switch' case</td></tr>
</table>

Details about action items in a full report

Exports can be used to extract information in various formats in order to import it into another tool. Clicking the **Create** button generates and downloads the file in your browser. Note that the availability of the export feature depends on your licence.

The following is an example of CSV export file obtained by launching the Project Portfolio export at model-level to get details:



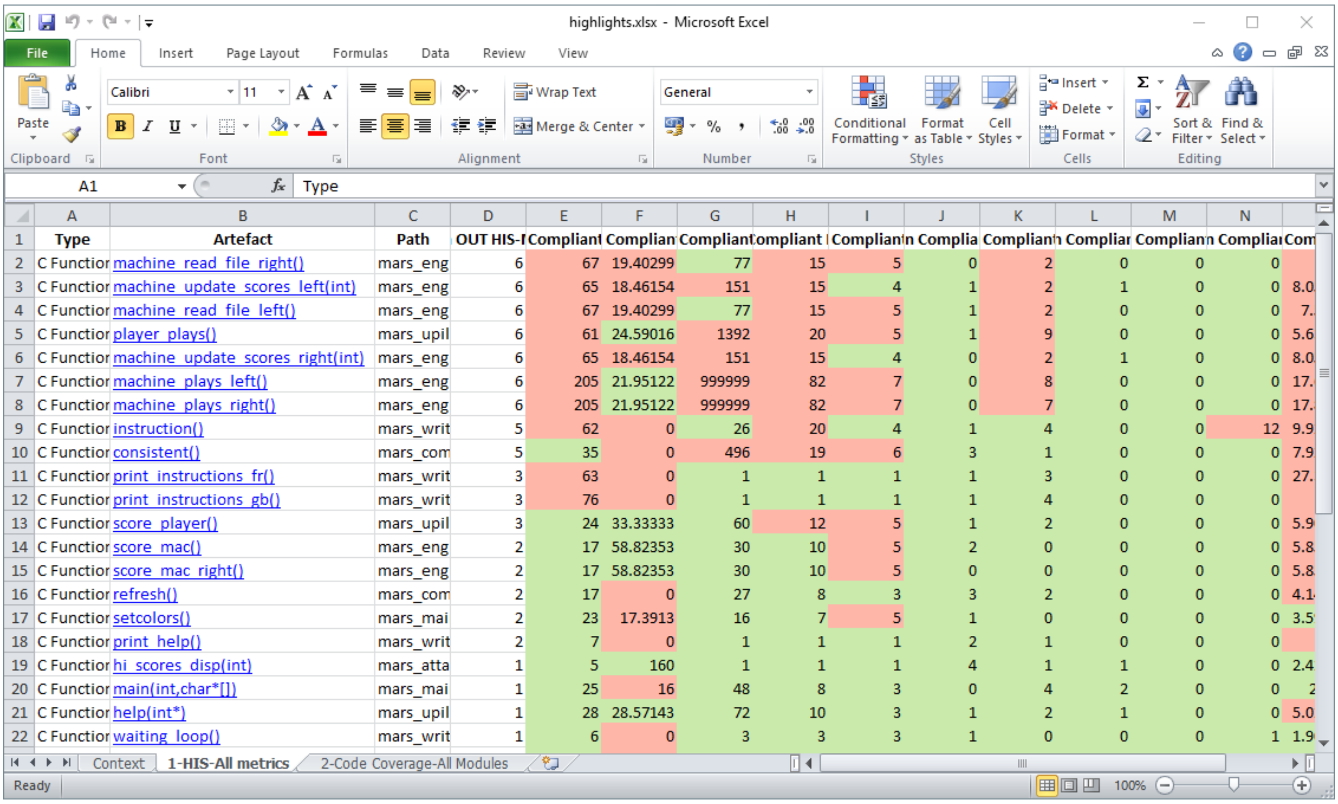
	A	B	C	D	E	F
1	model	app_name	v_name	status	sl_rank	level
2		Earth	V7	0	0.25	LEVELE
3		Mars	V3.2.6	0	0.5	LEVELF
4		Neptune	W26	0	0.25	LEVELE
5		Venus	Beta	0	0.25	LEVELE
6		Pluto	R9	0	0.25	LEVELE
7		Uranus	B625	0	0.25	LEVELE
8		Mercury	V2010B	0	0.5	LEVELF
9		Saturn	Prel	0	0.25	LEVELE
10		Freemind	0.9.0	0	0.5	LEVELF
11		HoDoKu	2.2	0	0.5	LEVELF
12		Jacoco	0.6.0	0	0.25	LEVELE
13		JFreeChart	1.0.13	0	0.25	LEVELE
14		kaptcha	6	0	0.5	LEVELF
15		Neptune2	V1	0	0.25	LEVELE
16		Sun	V7	0	0.25	LEVELE

The Project Portfolio export lists all ratings for projects in a specific analysis model

Other exports ask you to choose a list of highlights. In this case, the exported data contains the artefacts and metrics defined by the selected highlights.

The following is an example of a file generated by the

Highlights to Excel export with the *HIS: All Metrics* and *Code Coverage: All Modules* highlights selected for the Mars project:



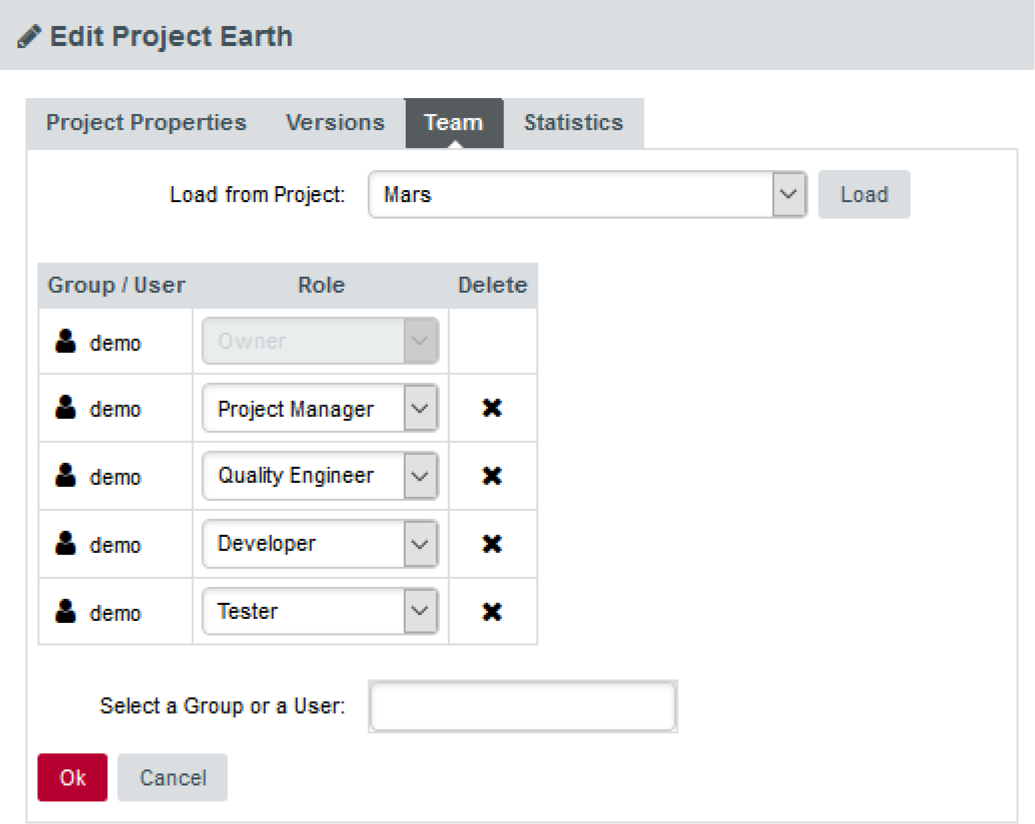
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	Type	Artefact	Path	OUT HIS-I	Compliant	Complian	Compliant	ompliant I	Complian	n Complia	Complian	n Compliar	Complian	n Complia	Com
2	C Functior	machine_read_file_right()	mars_eng	6	67	19.40299	77	15	5	0	2	0	0	0	
3	C Functior	machine_update_scores_left(int)	mars_eng	6	65	18.46154	151	15	4	1	2	1	0	0	8.0
4	C Functior	machine_read_file_left()	mars_eng	6	67	19.40299	77	15	5	1	2	0	0	0	7.
5	C Functior	player_plays()	mars_upil	6	61	24.59016	1392	20	5	1	9	0	0	0	5.6
6	C Functior	machine_update_scores_right(int)	mars_eng	6	65	18.46154	151	15	4	0	2	1	0	0	8.0
7	C Functior	machine_plays_left()	mars_eng	6	205	21.95122	999999	82	7	0	8	0	0	0	17.
8	C Functior	machine_plays_right()	mars_eng	6	205	21.95122	999999	82	7	0	7	0	0	0	17.
9	C Functior	instruction()	mars_writ	5	62	0	26	20	4	1	4	0	0	12	9.9
10	C Functior	consistent()	mars_com	5	35	0	496	19	6	3	1	0	0	0	7.9
11	C Functior	print_instructions_fr()	mars_writ	3	63	0	1	1	1	1	3	0	0	0	27.
12	C Functior	print_instructions_gb()	mars_writ	3	76	0	1	1	1	1	4	0	0	0	
13	C Functior	score_player()	mars_upil	3	24	33.33333	60	12	5	1	2	0	0	0	5.9
14	C Functior	score_mac()	mars_eng	2	17	58.82353	30	10	5	2	0	0	0	0	5.8
15	C Functior	score_mac_right()	mars_eng	2	17	58.82353	30	10	5	0	0	0	0	0	5.8
16	C Functior	refresh()	mars_com	2	17	0	27	8	3	3	2	0	0	0	4.1
17	C Functior	setcolors()	mars_mai	2	23	17.3913	16	7	5	1	0	0	0	0	3.5
18	C Functior	print_help()	mars_writ	2	7	0	1	1	1	2	1	0	0	0	
19	C Functior	hi_scores_disp(int)	mars_atta	1	5	160	1	1	1	4	1	1	0	0	2.4
20	C Functior	main(int,char*[])	mars_mai	1	25	16	48	8	3	0	4	2	0	0	2
21	C Functior	help(int*)	mars_upil	1	28	28.57143	72	10	3	1	2	1	0	0	5.0
22	C Functior	waiting_loop()	mars_writ	1	6	0	3	3	3	1	0	0	0	1	1.9

Excel Export from Highlights of the Mars project

Reports and Exports are highly customisable, consult your Squore administrator or refer to the Squore Configuration Guide to learn more about how to tweak the report contents or format.

Define roles for your team members

When you create a project, you become its owner, and remain the only user who can view it in Squore by default. In order to make it visible to more users, the project owner has to create a project team of users and groups and assign them project roles. This is done in the **Manage** page of a project in the **Team** tab, as shown below:



✎ Edit Project Earth

Project Properties | Versions | Team | Statistics

Load from Project: Mars — Load

Group / User	Role	Delete
👤 demo	Owner	
👤 demo	Project Manager	✖
👤 demo	Quality Engineer	✖
👤 demo	Developer	✖
👤 demo	Tester	✖

Select a Group or a User:

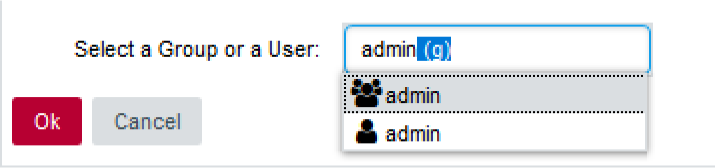
Ok — Cancel

The Team tab

The project scope can be set directly from the command line when creating a new project, if you use the **teamUser** and **teamGroup** options. For more details, refer to the Command Line Interface manual.

In order to give visibility to the user **admin** over the projects created by the user **demo**, follow these steps:

1. Log in as the demo user and go to the Projects page.
2. Click the Manage icon (✎) for the project Earth
3. Click on the Team tab to view the project team.
4. Type admin in **Select a Group or a User**. The list will show all users (👤) and groups (👥) available matching the search term.



Select a Group or a User: admin (g)

👥 admin

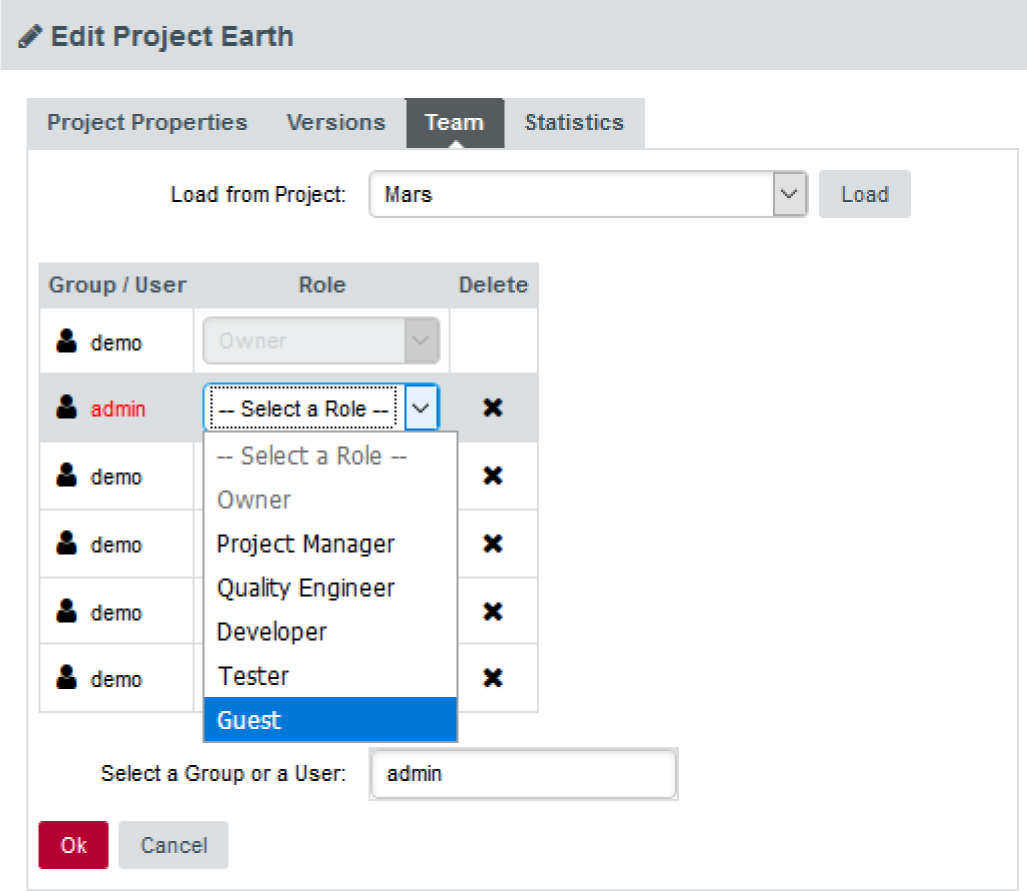
👤 admin

Ok — Cancel

The users and groups matching admin

Click the admin user to add it to the project team.

5. Now that admin is listed in the project team, you need to pick a project role for the user within this project. Select **Guest** from the list.



Load from Project: Mars

Group / User	Role	Delete
demo	Owner	
admin	-- Select a Role --	✖
demo		✖
demo		✖
demo		✖
demo		✖

Select a Group or a User: admin

The project roles available for the user in this project

This predefined project role allows a user to consult the results of baseline versions of a project without making any changes. For more information about project roles, consult User Project Roles.

6. Click **Apply** to apply your changes.

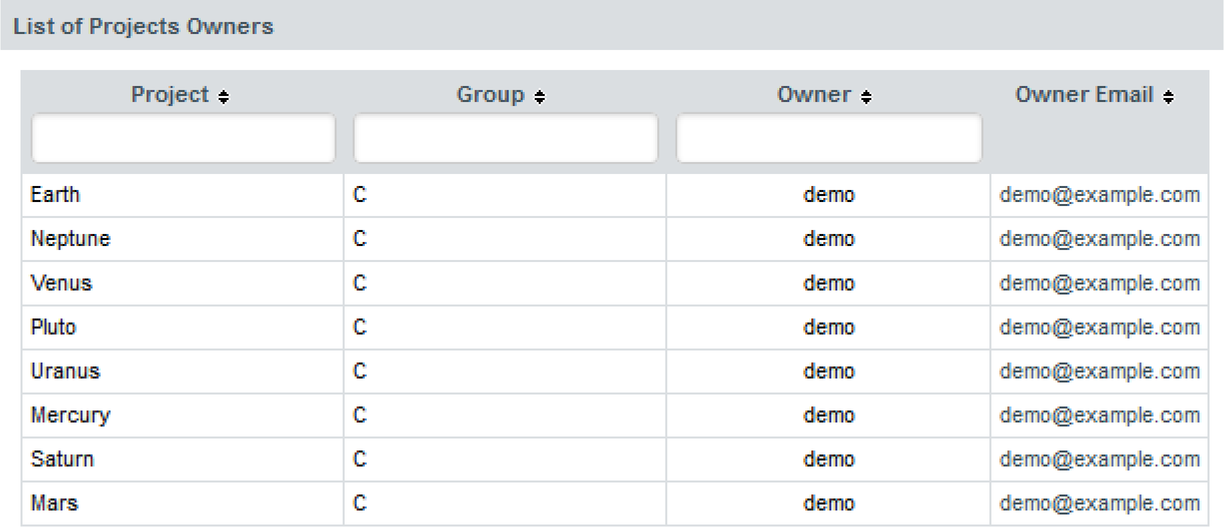
The admin user can now log in and will see the Earth project in their Explorer.

If you want to configure the rest of the sample projects the way you configured Earth, you can copy the project team to another project:

1. Click on **Manage > Team** for the project Mars
2. Select Earth from the **Load from Project** dropdown and click **Load**.
3. The users and their project roles have now been copied as they were set up in the Earth project. You can make adjustments or click **Apply** to confirm your changes.

Finding Other Projects

A Squore administrator may allow you browse a list of projects created by other users so you can contact them and request to be added to their project's team. When this feature is enabled, you can click the **Ask access to Project Owners** button on the Projects page to view a list of projects that other users have created, as shown below:



List of Projects Owners

Project	Group	Owner	Owner Email
Earth	C	demo	demo@example.com
Neptune	C	demo	demo@example.com
Venus	C	demo	demo@example.com
Pluto	C	demo	demo@example.com
Uranus	C	demo	demo@example.com
Mercury	C	demo	demo@example.com
Saturn	C	demo	demo@example.com
Mars	C	demo	demo@example.com

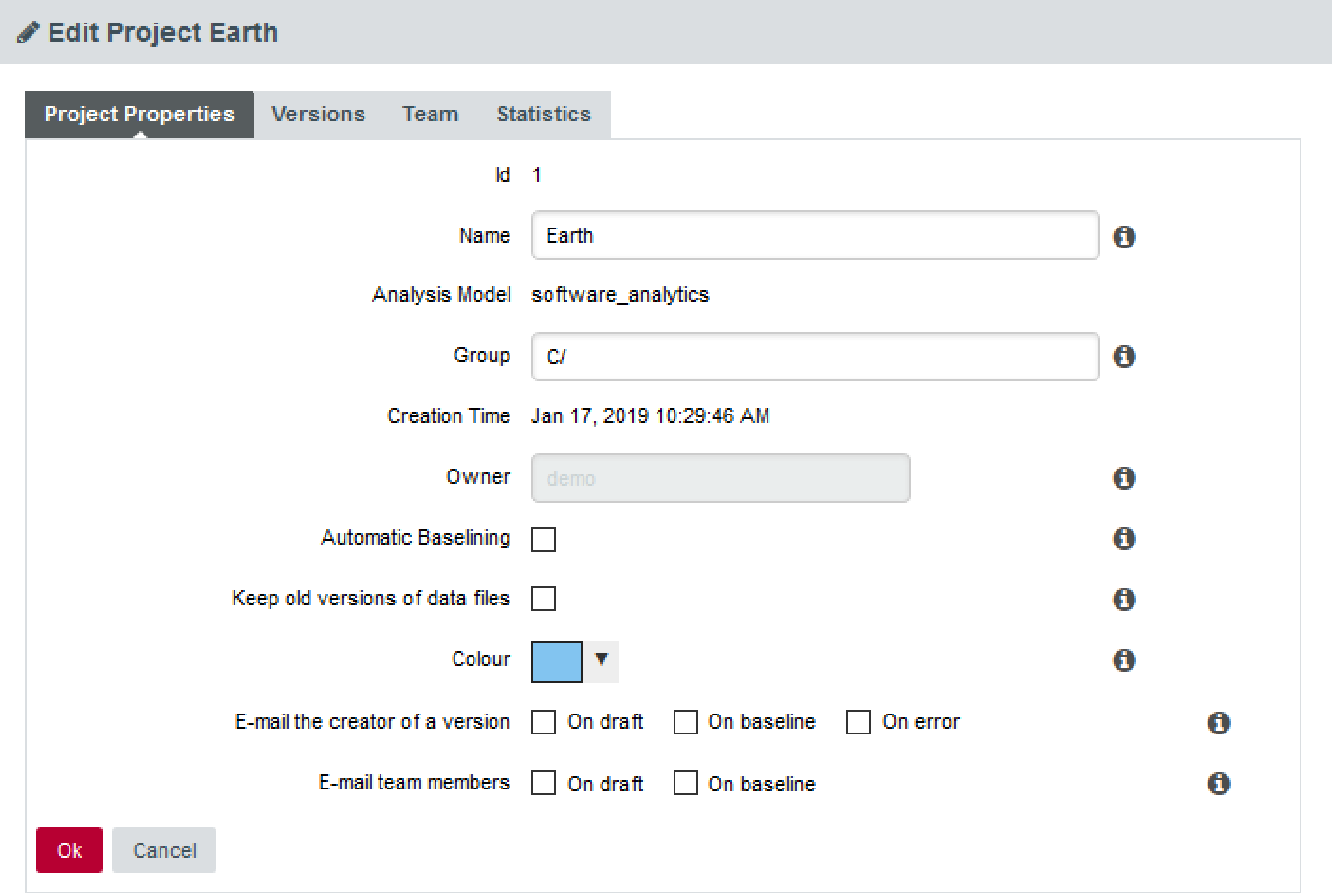
The list of available projects on the server and their owners

If you want to become a team member of one of the projects listed, click the project owner's e-mail address to send them a message and request to be added to their project's team.

If the **Ask access to Project Owners** button is not displayed on the Projects page, contact your Squore administrator to set up access following the instructions provided in the Installation and Administration Guide.

E-mail Notifications

You can configure each project in Squore so that an e-mail notification is sent out after a new version is created. This functionality is available for users who can create and manage projects, either in the **General Information** section of the project wizard, or the in the Project Properties tab of the Manage Project page:



Id	1
Name	Earth
Analysis Model	software_analytics
Group	C/
Creation Time	Jan 17, 2019 10:29:46 AM
Owner	demo
Automatic Baselining	□
Keep old versions of data files	□
Colour	🟦 ▼
E-mail the creator of a version	□ On draft □ On baseline □ On error
E-mail team members	□ On draft □ On baseline

E-mail notification options in Manage Project

The conditions on which you can to trigger an e-mail are:

- **On draft:** sends an e-mail every time a draft version is successfully analysed.
- **On baseline:** sends an e-mail every time a baseline version is successfully analysed, or every time a draft version is baselined.
- **On error:** sends an e-mail every time an analysis ends with the *Warnings* or *Error* status.

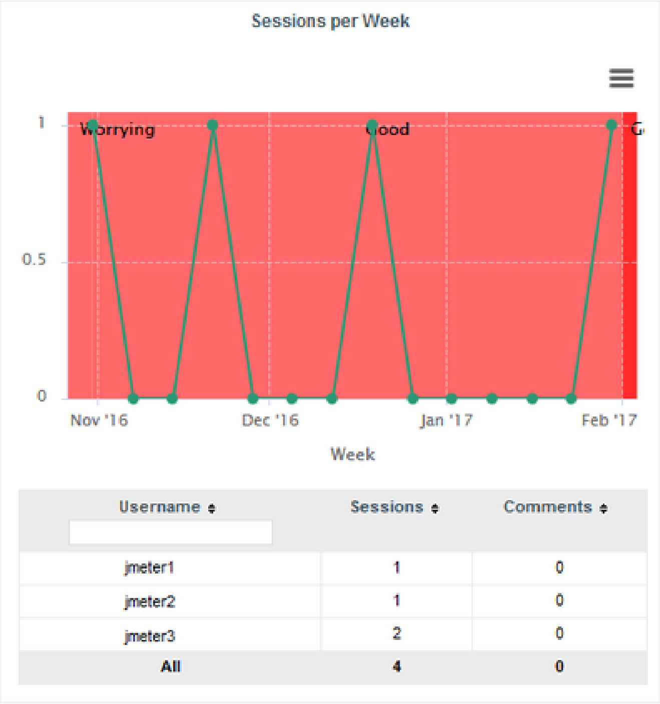
The e-mail contains a description of the version, the number of new artefacts, the number of action items, a list of the new action items and the number of new findings.

Usage Statistics

You can get information on how your collaborators are viewing the projects you manage or the models you develop by using the statistics features of Squore. This section describes the information available to project managers and model developers via **Models > Statistics** and the **Manage Project** page.

Statistics for Project Managers

As a project manager, you can use project statistics to investigate the popularity of your project by going to **Manage > Statistics**.



Username ⬍	Sessions ⬍	Comments ⬍
jmeter1	1	0
jmeter2	1	0
jmeter3	2	0
All	**4**	**0**

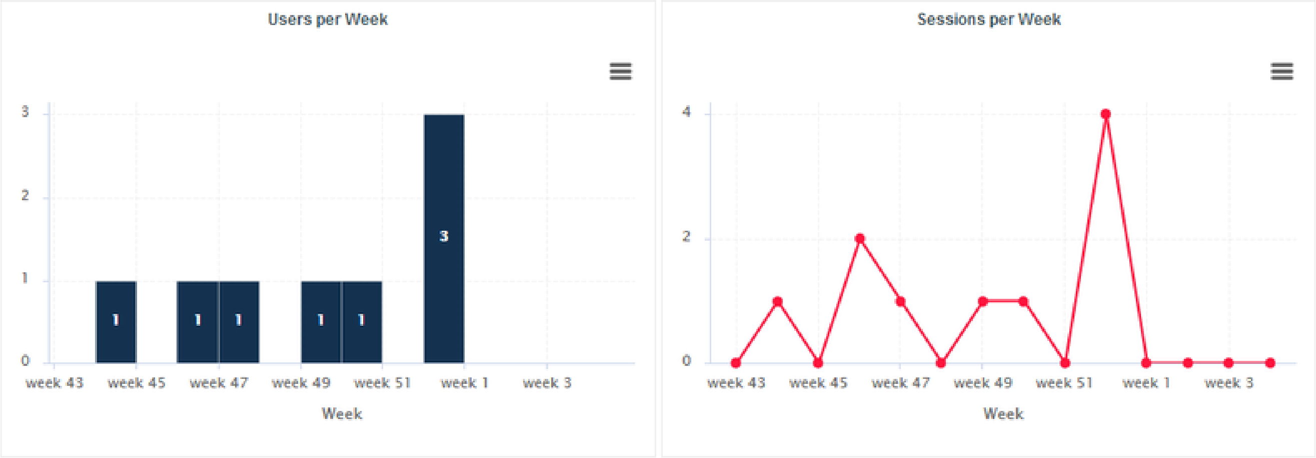
When you select a reporting period, the following information is displayed:

- The trend of the number of views for this project uses the colours from the scale of the root indicator as background to help you correlate the project rating with the number of visits.
- The treemap helps you understand which of the dashboard tabs is the most visited for this project.
- A table summarises the breakdown of views per user, as well as the number of comments left by each user.

Statistics for Model Developers

You can learn more about the usage of particular features of a model by clicking **Models > Statistics**. For each analysis model, find out how many users consult results, which projects are the most popular and which regions and charts of the dashboard are the most useful for users.

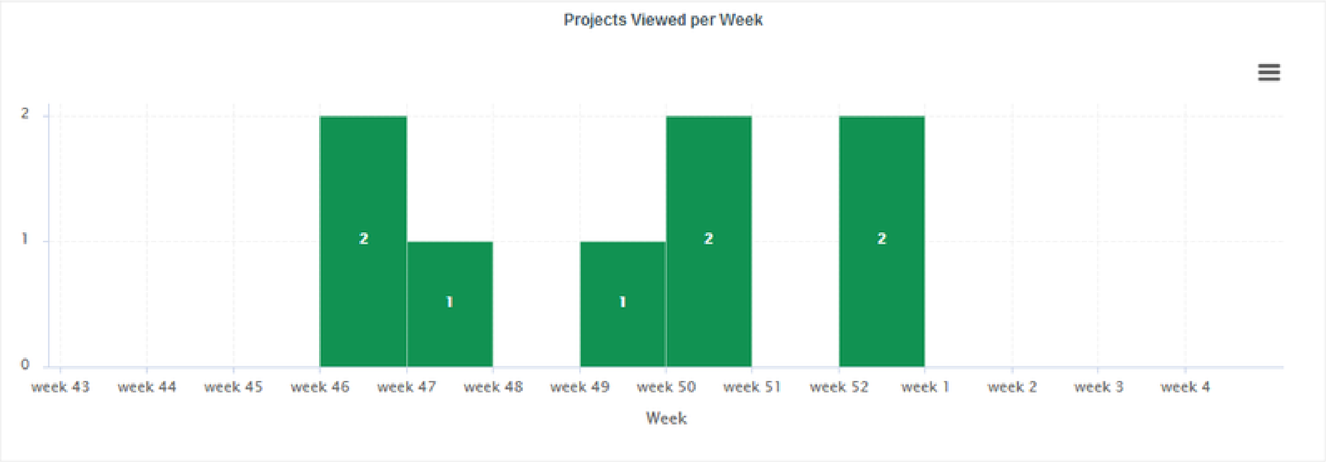
Users



The Users tab displays the information about the number of users and overall connections to the server for projects in this analysis model

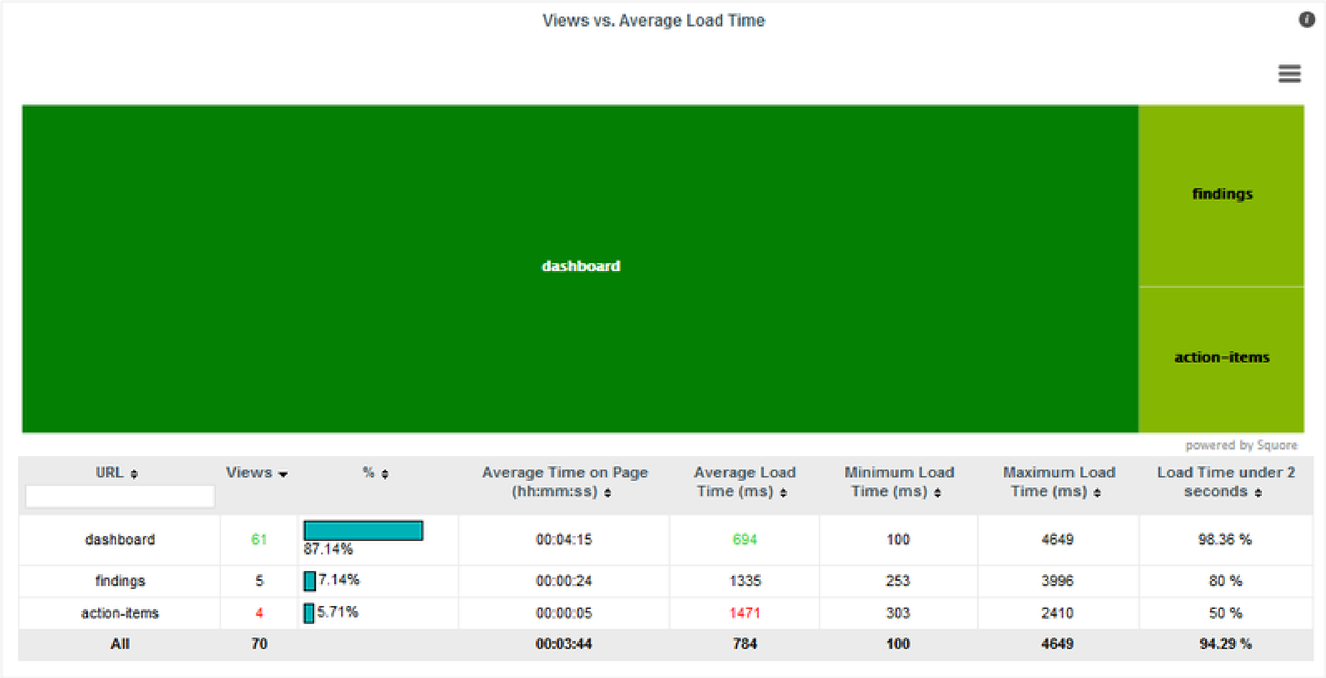
The Users tab displays the information about the number of users and overall connections to the server for projects in this analysis model.

Projects



On the Projects tab, you can check how many projects had visitors over the selected period.

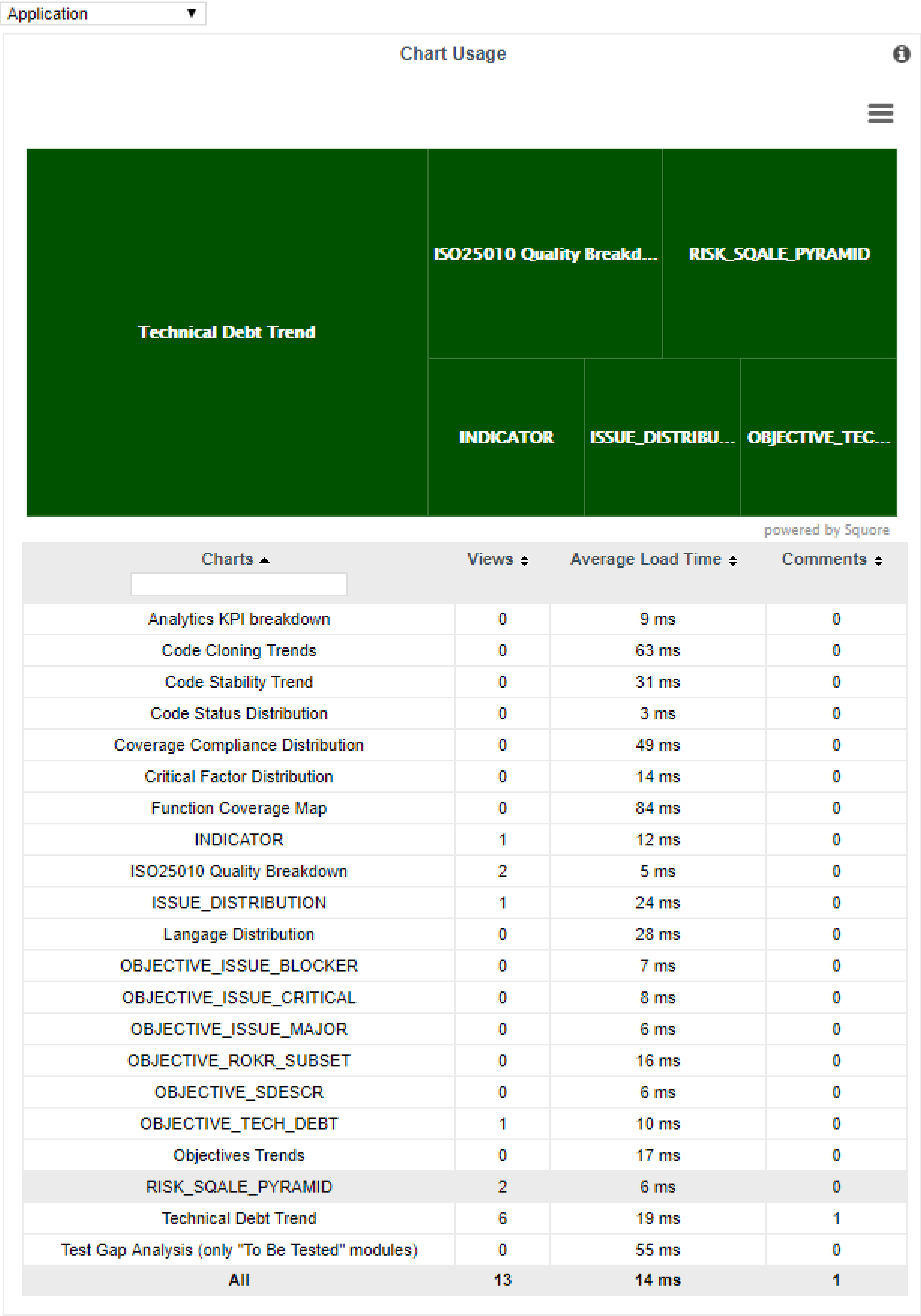
Dashboard



URL ⬍	Views ▾	% ⬍	Average Time on Page (hh:mm:ss) ⬍	Average Load Time (ms) ⬍	Minimum Load Time (ms) ⬍	Maximum Load Time (ms) ⬍	Load Time under 2 seconds ⬍
dashboard	61	87.14%	00:04:15	694	100	4649	98.36 %
findings	5	7.14%	00:00:24	1335	253	3996	80 %
action-items	4	5.71%	00:00:05	1471	303	2410	50 %
All	**70**		**00:03:44**	**784**	**100**	**4649**	**94.29 %**

The Dashboard tab allows analysing the usage of each tab on the dashboard. Each tab is represented in a treemap according to how many views it receives. This information can be used to adjust the default display status of each tab or their availability to end users.

Charts



Charts ▲	Views ⬍	Average Load Time ⬍	Comments ⬍
Analytics KPI breakdown	0	9 ms	0
Code Cloning Trends	0	63 ms	0
Code Stability Trend	0	31 ms	0
Code Status Distribution	0	3 ms	0
Coverage Compliance Distribution	0	49 ms	0
Critical Factor Distribution	0	14 ms	0
Function Coverage Map	0	84 ms	0
INDICATOR	1	12 ms	0
ISO25010 Quality Breakdown	2	5 ms	0
ISSUE_DISTRIBUTION	1	24 ms	0
Langage Distribution	0	28 ms	0
OBJECTIVE_ISSUE_BLOCKER	0	7 ms	0
OBJECTIVE_ISSUE_CRITICAL	0	8 ms	0
OBJECTIVE_ISSUE_MAJOR	0	6 ms	0
OBJECTIVE_ROKR_SUBSET	0	16 ms	0
OBJECTIVE_SDESCR	0	6 ms	0
OBJECTIVE_TECH_DEBT	1	10 ms	0
Objectives Trends	0	17 ms	0
RISK_SQALE_PYRAMID	2	6 ms	0
Technical Debt Trend	6	19 ms	1
Test Gap Analysis (only "To Be Tested" modules)	0	55 ms	0
All	**13**	**14 ms**	**1**

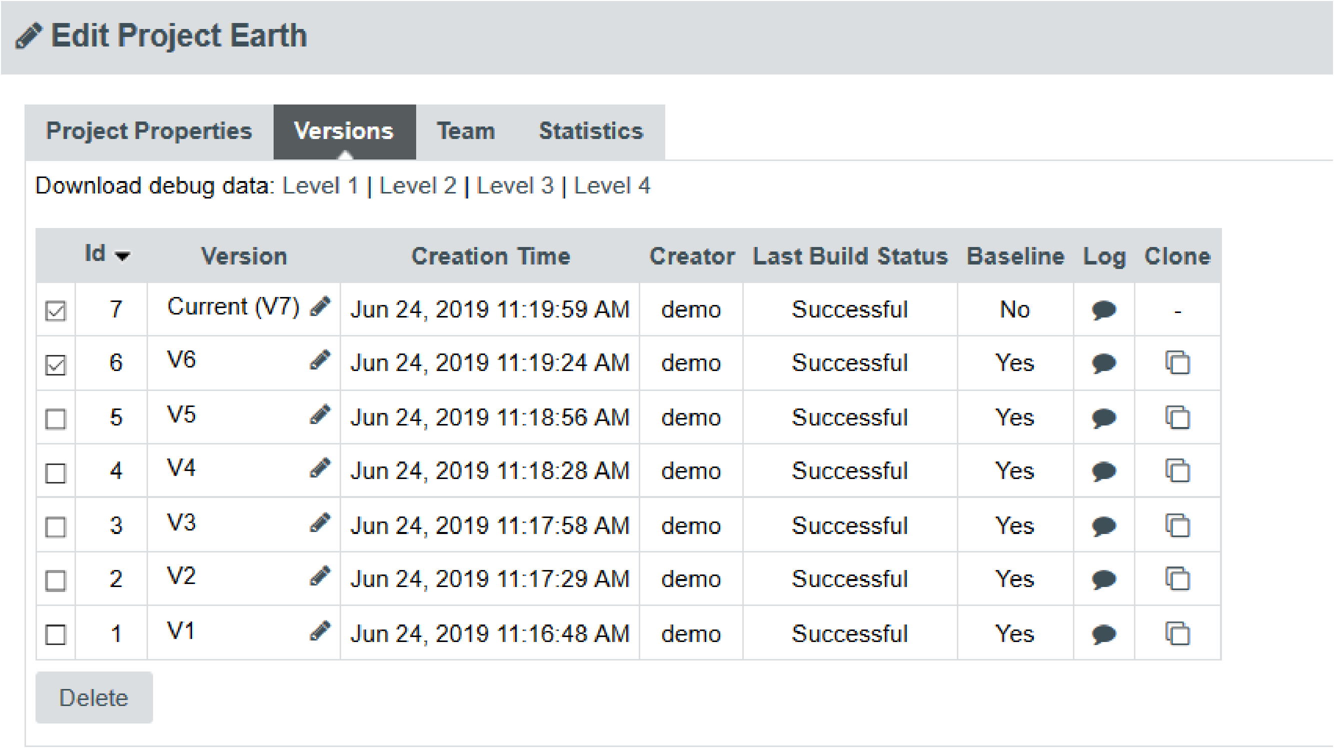
The Charts tab provides information about chart usage in your model: The number of views per chart (per artefact type or for all artefact types), the average loading time and the number of comments.

Chapter 11. Keep it Tidy: Project Maintenance in Squore

Managing Previous Analyses

You can delete or rename one or more of the last versions of a project if needed. This can be done from the **Projects** page if you are the project creator or are a member of a project role that allows for managing the project.

If you want to manage the previous analyses of the Earth project, log in as the demo user and click **Projects**. Click the Manage icon (✎) and open the **Versions** tab to view the list of versions created for this project:



Download debug data: Level 1 | Level 2 | Level 3 | Level 4

	Id ▾	Version	Creation Time	Creator	Last Build Status	Baseline	Log	Clone
☑	7	Current (V7) ✎	Jun 24, 2019 11:19:59 AM	demo	Successful	No	💬	-
☑	6	V6 ✎	Jun 24, 2019 11:19:24 AM	demo	Successful	Yes	💬	⧉
□	5	V5 ✎	Jun 24, 2019 11:18:56 AM	demo	Successful	Yes	💬	⧉
□	4	V4 ✎	Jun 24, 2019 11:18:28 AM	demo	Successful	Yes	💬	⧉
□	3	V3 ✎	Jun 24, 2019 11:17:58 AM	demo	Successful	Yes	💬	⧉
□	2	V2 ✎	Jun 24, 2019 11:17:29 AM	demo	Successful	Yes	💬	⧉
□	1	V1 ✎	Jun 24, 2019 11:16:48 AM	demo	Successful	Yes	💬	⧉

The Versions of the Earth project

The most recent version always appears at the top of the list.

By clicking the pen icon next to the version name, you can rename this analysis. Your changes will immediately be reflected in the Project Portfolios.

In order to delete an analysis, check the box next to the version you want to delete. All versions created after the version you selected will also be checked. Click the **Delete** button to reach a summary page where you can confirm which versions will be deleted, and click **Confirm** to launch the delete process.

If you select to delete all the versions of a project, the entire project will be deleted.

Deleting a Project

Projects can be deleted by their creator or members of a project role that allows to manage projects. In order to delete a project, click **Projects** and click the delete icon (✖) next to the project you want to delete. After confirming the operation, the project is deleted from the Squore database and cannot be restored.

Squore Server Administration

A Squore Administrator can access functionality that does not involve working with projects directly. You can access the **Administration** menu if you need to perform any of the following tasks:

- Create, update, remove and deactivate Squore users (**Administration > Users**)
- Create, update, and remove groups (**Administration > Groups**)
- Create, update, and remove roles (**Administration > Roles**)
- Configure and monitor the Squore Server installation (**Administration > System**)
- View and manage all projects created on Squore Server (**Administration > Projects**)
- Reload the server configuration from disk (**Administration > Reload Configuration**)

For more information about administration functionality, consult the Online Help.

What About Server Maintenance?

Server maintenance, including database backups need to be carried out by a system administrator directly on Squore Server. If you need to know more about the backup options offered by Squore, refer to the Squore Installation and Administration Guide.

Chapter 12. Repository Connectors

Folder Path

Description

The simplest method to analyse source code in Squore is to provide a path to a folder containing your code.

> Remember that the path supplied for the analysis is a path local to the machine running the analysis, which may be different from your local machine. If you analyse source code on your local machine and then send results to the server, you will not be able to view the source code directly in Squore, since it will not have access to the source code on the other machine. A common workaround to this problem is to use UNC paths (\\*Server**Share*, *smb://server/share…*) or a mapped server drive in Windows.

Usage

Folder Path has the following options:

- **Datapath (path, mandatory)**:
  - **Absolute Path**: the absolute path to the folder containing the files you want to include in the analysis. The path specified must be accessible from the server and user must have **Access Server Resources** permission.
  - **Authorized Paths**: a list of server paths accessible for all users, regardless of the **Access Server Resources** permission. This list can only be configured by a Squore administrator : Configure Authorized Server Paths.

The full command line syntax for Folder Path is:

```
-r "type=FROMPATH,path=[text]"
```

Zip Upload

Description

This Repository Connector allows you to upload a zip file containing your sources to analyse. Select a file to upload in the project wizard and it will be extracted and analysed on the server.

> The contents of the zip file are extracted into Squore Server's temp folder. If you want to uploaded files to persist, contact your Squore administrator so that the uploaded zip files and extracted sources are moved to a location that is not deleted at each server restart.

Usage

This Repository Connector is only available from the web UI, not from the command line interface.

Synergy

Description

Rational Synergy is a software tool that provides software configuration management (SCM) capabilities for all artifacts related to software development including source code, documents and images as well as the final built software executable and libraries.

For more details, refer to http://www-03.ibm.com/software/products/en/ratisyne.

> The Synergy Repository Connector assumes that a project already exists and that the Synergy user defined has the right to access it.
>
> The host where the analysis takes place must have Synergy installed and fully functional. Note that, using credentials is only supported on Windows, so use the NO_CREDENTIALS option when Synergy runs on a Linux host (consult IBM's documentation at http://pic.dhe.ibm.com/infocenter/synhelp/v7m2r0/index.jsp?topic=%2Fcom.ibm.rational.synergy.manage.doc%2Ftopics%2Fsc_t_h_start_cli_session.html for more details).
>
> The following is a list of commands used by the Synergy Repository Connector to retrieve sources:
>
> ```
> ccm start -d $db -nogui -m -q [-s $server] [-pw $password] [-n $user
> -pw password]
> ```
>
> ```
> ccm prop "$path@$projectSpec"
> ```
>
> ```
> ccm copy_to_file_system -path $tempFolder -recurse $projectSpec
> ```
>
> ```
> ccm cat "$artefactPath@$projectSpec"
> ```
>
> ```
> ccm stop
> ```

Usage

Synergy has the following options:

- **Server URL (`server`)**: Specify the Synergy server URL, if using a distant server. If specified, the value is used by the Synergy client via the -s parameter.
- **Database (`db`, mandatory)**: Specify the database path to analyse the sources it contains.
- **Project Specification (`projectSpec`, mandatory)**: Specify the project specification for the analysis. Source code contained in this project specification will be analysed recursively.
- **Subfolder (`subFolder`)**: Specify a subfolder name if you want to restrict the scope of the analysis to a particular folder.

- **Include Subprojects (`subProject`, default: yes)**: This option creates work area copies for the specified projects and all subprojects. If this option is not on, subprojects are ignored.

- **Ignore links (`ignoreLinks`, default: no)**: This option is used to ignore links to subprojects. This option is valid only on Linux systems.

- **Authentication: (`useAccountCredentials`, default: NO_CREDENTIALS)**: Note that, as stated in IBM's documentation, using credentials is only supported on Windows. The "No Credentials" option must be used when Synergy runs on a Linux host. For more information, consult http://pic.dhe.ibm.com/infocenter/synhelp/v7m2r0/index.jsp?topic=%2Fcom.ibm.rational.synergy.manage.doc%2Ftopics%2Fsc_t_h_start_cli_session.html.

- **Username (`username`)**:

- **Password (`password`)**:

The full command line syntax for Synergy is:

```
-r
"type=Synergy,server=[text],db=[text],projectSpec=[text],subFolder=[text],subProject=[
multipleChoice],ignoreLinks=[multipleChoice],useAccountCredentials=[multipleChoice],us
ername=[text],password=[password]"
```

ClearCase

Description

IBM Rational ClearCase is a software configuration management solution that provides version control, workspace management, parallel development support, and build auditing. The command executed on the server to check out source code is: $cleartool $view_root_path $view $vob_root_path.

For more details, refer to http://www-03.ibm.com/software/products/en/clearcase.

> The ClearCase tool is configured for Linux by default. It is possible to make it work for Windows by editing the configuration file

Usage

ClearCase has the following options:

- **View root path (`view_root_path`, mandatory, default: /view)**: Specify the absolute path of the ClearCase view.

- **Vob Root Path (`vob_root_path`, mandatory, default: /projets)**: Specify the absolute path of the ClearCase vob.

- **View (`view`)**: Specify the label of the view to analyse sources from. If no view is specified, the current ClearCase view will be used automatically, as retrieved by the command cleartool pwv -s.

- **Server Display View (`server_display_view`)**: When viewing source code from the Explorer after building the project, this parameter is used instead of the view parameter specified earlier. Leave this field empty to use the same value as for view.

- **Sources Path (`sub_path`)**: Specify a path in the view to restrict the scope of the source code to analyse. The value of this field must not contain the vob nor the view. Leave this field empty to analyse the code in the entire view. This parameter is only necessary if you want to restrict to a directory lower than root.

The full command line syntax for ClearCase is:

```
-r
"type=ClearCase,view_root_path=[text],vob_root_path=[text],view=[text],server_display_
view=[text],sub_path=[text]"
```

Git

Description

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

For more details, refer to http://git-scm.com/.

The following is a list of commands used by the Git Repository Connector to retrieve sources:

```
git clone [$username:$password@]$url $tmpFolder
```

```
git checkout $commit
```

```
git log -1 "--format=%H"
```

```
git config --get

remote.origin.url
```

```
git clone [$username:$password@]$url $tmpFolder
```

```
git checkout $commit
```

```
git fetch
```

```
git --git-dir=$gitRoot show $artefactPath
```

Git 1.7.1 is known to fail with a *fatal: HTTP request failed* error on CentOS 6.9. For this OS, it is recommended to upgrade to git 2.9 as provided by software collections on https://www.softwarecollections.org/en/scls/rhscl/rh-git29/ and point to the new binary in *git_config.tcl* or make the change permanent as described on https://access.redhat.com/solutions/527703.

Usage

Git has the following options:

- **URL (`url`, mandatory)**: URL of the git repository to get files from. The local, HTTP(s), SSH and Git protocols are supported.

- **Branch or commit (`commit`)**: This field allows specifying the SHA1 of a commit or a branch name. If a SHA1 is specified, it will be retieved from the default branch. If a branch label is specified, then its latest commit is analysed. Leave this field empty to analyse the latest commit of the default branch.

- **Sub-directory (`subDir`)**: Specify a subfolder name if you want to restrict the analysis to a subpath of the repository root.

- **Authentication (`useAccountCredentials`, default: NO_CREDENTIALS)**: Possible values for authentication are

    - **No credentials**: Used when the underlying Git is open and does not require authetication
    - **Use my Squore credentials**: If the login/password are the same between Squore and the underlying git
    - **Define credentials**: To be prompted for login/password

- **Username (`username`)**:

- **Password (`password`)**:

The full command line syntax for Git is:

```
-r
"type=Git,url=[text],commit=[text],subDir=[text],useAccountCredentials=[multipleChoice
],username=[text],password=[password]"
```

Perforce

Description

The Perforce server manages a central database and a master repository of file versions. Perforce supports both Git clients and clients that use Perforce's own protocol.

For more details, refer to http://www.perforce.com/.

The Perforce repository connector assumes that the specified depot exists on the specified Perforce server, that Squore can access this depot and that the Perforce user defined has the right to access it.

The host where the analysis takes place must have a Perforce command-line client (p4) installed and fully functional.

The P4PORT environment variable is not read by Squore. You have to set it in the form. The path to the p4 command can be configured in the *perforce_conf.tcl* file located in the *configuration/repositoryConnectors/Perforce* folder.

The following is a list of commands used by the Perforce Repository Connector to retrieve sources:

```
p4 -p $p4port [-u username] [-P password] client -i
<$tmpFolder/p4conf.txt
```

```
p4 -p $p4port [-u username] [-P password] -c $clientName sync
"$depot/...@$label"
```

```
p4 -p $p4port [-u username] [-P password] client -d $clientName
```

```
p4 -p $p4port [-u username] [-P password] print -q -o $outputFile
$artefactPath
```

The format of the *p4conf.txt* file is:

```
Client: $clientName

Root: $tmpFolder

Options: noallwrite noclobber nocompress unlocked nomodtime normdir

SubmitOptions: submitunchanged

view:

$depot/... //$clientName/...
```

Usage

Perforce has the following options:

- **P4PORT (`p4port`, mandatory)**: Specify the value of P4PORT using the format [protocol:]host:port (the protocol is optional). This parameter is necessary even if you have specified an environment variable on the machine where the analysis is running.

- **Depot (`depot`, mandatory)**: Specify the name of the depot (and optionnally subforders) containing the sources to be analysed.

- **Revision (`label`)**: Specify a label, changelist or date to retrieve the corresponding revision of the sources. Leave this field empty to analyse the most recent revision fo the sources.
- **Authentication (`useAccountCredentials`, default: NO_CREDENTIALS)**:
- **Username (`username`)**:
- **Password (`password`)**:

The full command line syntax for Perforce is:

```
-r
"type=Perforce,p4port=[text],depot=[text],label=[text],useAccountCredentials=[multiple
Choice],username=[text],password=[password]"
```

SVN

Description

Connecting to an SVN server is supported using svn over ssh, or by using a username and password.

For more details, refer to https://subversion.apache.org/.

The following is a list of commands used by the SVN Repository Connector to retrieve sources (you can edit the common command base or the path to the executable in *<SQUORE_HOME>/configuration/repositoryConnectors/SVN/svn_conf.tcl* if needed):

```
svn info --xml --non-interactive --trust-server-cert --no-auth-cache
[--username $username] [--password $password] [-r $revision] $url
```

```
svn export --force --non-interactive --trust-server-cert --no-auth
-cache [--username $username] [--password $password] [-r $revision]
$url
```

This Repository Connector includes a hybrid SVN mode saves you an extra checkout of your source tree when using the `local_path` attribute. Consult the reference below for more details.

Usage

SVN has the following options:

- **URL (`url`, mandatory)**: Specify the URL of the SVN repository to export and analyse. The following protocols are supported: svn://, svn+ssh://, http://, https:// .

- **Revision (`rev`)**: Specify a revision number in this field, or leave it blank to analyse files at the HEAD revision.

- **External references (`externals`, default: exclude)**: Specify if when extracting sources from SVN the system should also extract external references.

- **Sources are already extracted in (`local_path`)**: Specify a path to a folder where the sources have already been extracted. When using this option, sources are analysed in the specified folder instead of being checked out from SVN. At the end of the analysis, the url and revision numbers are attached to the analysed sources, so that any source code access from the web interface always retrieves files from SVN. This mode is mostly used to save an extra checkout in some continuous integration scenarios.

- **Authentication (`useAccountCredentials`, default: NO_CREDENTIALS)**:

- **Username (`username`)**:

- **Password (`password`)**:

The full command line syntax for SVN is:

```
-r
"type=SVN,url=[text],rev=[text],externals=[multipleChoice],local_path=[directory],useA
ccountCredentials=[multipleChoice],username=[text],password=[password]"
```

Folder (use GNATHub)

Description

Retrieve Sources from a folder on the server, use GNATHub to limit the files (compatible with GNAT Pro versions 7.4.2 up to 18.2).

This Repository Connector will only be available after you configure your server or client *config.xml* with the path to your gnathub executable with a *<path name="gnatub" path="C:\tools\GNAThub\gnathub.exe" />* definition. Consult the Configuration Manual for more information about referencing external executables.

Usage

Folder (use GNATHub) has the following options:

- **Path of the source files (`path`)**: Specify the absolute path to the files you want to include in the analysis. The path specified must be accessible from the server.

- **Path of the gnathub.db file (`gnatdb`)**: Specify the absolute path of the gnathub.db file.

- **Root path for sources in the GNAT DB (`gnat_root`)**: Specify the root path for sources in the GNAT DB

The full command line syntax for Folder (use GNATHub) is:

```
-r "type=GNAThub,path=[text],gnatdb=[text],gnat_root=[text]"
```

PTC Integrity

Description

This Repository Connector allows analysing sources hosted in PTC Integrity, a software system lifecycle management and application lifecycle management platform developed by PTC.

For more details, refer to http://www.ptc.com/products/integrity/.

You can modify some of the settings of this repository connector if the si.exe and mksAPIViewer.exe binaries are not in your path. For versions that do not support the --xmlapi option, you can also turn off this method of retrieving file information. These settings are available by editing *mks_conf.tcl* in the repository connector's configuration folder.

Usage

PTC Integrity has the following options:

- **Server Hostname (`hostname`, mandatory)**: Specify the name of the Integrity server. This value is passed to the command line using the parameter --hostname.

- **Port (`port`)**: Specify the port used to connect to the Integrity server. This value is passed to the command line using the parameter --port.

- **Project (`project`)**: Specify the name of the project containing the sources to be analysed. This value is passed to the command line using the --project parameter.

- **Revision (`revision`)**: Specify the revision number for the sources to be analysed. This value is passed to the command line using the --projectRevision parameter.

- **Scope (`scope`, default: name:**.c,name:*.h)*: Specifies the scope (filter) for the Integrity sandbox extraction. This value is passed to the command line using the --scope parameter.

- **Authentication (`useAccountCredentials`, default: NO_CREDENTIALS)**:

- **Username (`username`)**:

- **Password (`password`)**:

The full command line syntax for PTC Integrity is:

```
-r
"type=MKS,hostname=[text],port=[text],project=[text],revision=[text],scope=[text],useA
ccountCredentials=[multipleChoice],username=[text],password=[password]"
```

CVS

Description

The Concurrent Versions System (CVS), is a client-server free software revision control system in the field of software development.

For more details, refer to http://savannah.nongnu.org/projects/cvs.

> The following is a list of commands used by the CVS Repository Connector to retrieve sources:
>
> ```
> cvs -d $repository export [-r $branch] $project
> ```
>
> ```
> cvs -d $repository co -r $artefactPath -d $tmpFolder
> ```

Usage

CVS has the following options:

- **Repository (`repository`, mandatory)**: Specify the location of the CVS Repository.
- **Project (`project`, mandatory)**: Specify the name of the project to get files from.
- **Tag or Branch (`branch`)**: Specify the tag or branch to get the files from.

The full command line syntax for CVS is:

```
-r "type=CVS,repository=[text],project=[text],branch=[text]"
```

TFS

Description

Team Foundation Server (TFS) is a Microsoft product which provides source code management, reporting, requirements management, project management, automated builds, lab management, testing and release management capabilities. This Repository Connector provides access to the sources hosted in TFS's revision control system.

For more details, refer to https://www.visualstudio.com/products/tfs-overview-vs.

The TFS repository connector (Team Foundation Server - Team Foundation Version Control) assumes that a TFS command-line client is installed and fully functional on the machine where the analysis runs. Two types of clients are supported: Team Explorer Everywhere (the java client, enabled by default) and Visual Studio Client (tf.exe).

Prior to using this repository connector, ensure that you have configured it to use the right client by adjusting settings in *<SQUORE_HOME>/configuration/repositoryConnectors/TFS/tfs_conf.tcl* file.

The Repository Connector form must be filled according to the TFS standard (eg. the Project Path must begin with the '$' character…). Note that this repository connector works with a temporary workspace that is deleted at the end of the analysis. The following is a list of commands used by the TFS Repository Connector to retrieve sources (this example uses the Windows client):

```
tf.exe workspace [/login:$username,$password] /server:$url /noprompt
/new $workspace
```

```
tf.exe workfold [/login:$username,$password] /map $path $tempFolder
/workspace:$workspace
```

```
tf.exe get [/login:$username,$password] /version:$version /recursive
/force $path
```

```
tf.exe workspace [/login:$username,$password] /delete $workspace
```

The following command is used when viewing sources in the web interface:

```
tf.exe view [/login:$username,$password] /server:$artefactPath
```

When using the Java Team Explorer Everywhere client, **/** is replaced by **-** and the **view** command is replaced by **print**.

Usage

TFS has the following options:

- **URL (`URL`, mandatory)**: Specify the URL of the TFS server.
- **Path (`path`, mandatory)**: Path the project to be analysed. This path usually starts with $.
- **Version (`version`)**: Specify the version of the sources to analyse. This field accepts a changeset number, date, or label. Leave the field empty to analyse the most recent revision of the sources.
- **Authentication (`useAccountCredentials`, default: NO_CREDENTIALS)**:
- **Username: (`username`)**:

- **Password (`password`)**:

The full command line syntax for TFS is:

```
-r
"type=TFS,URL=[text],path=[text],version=[text],useAccountCredentials=[multipleChoice]
,username=[text],password=[password]"
```

Using Multiple Nodes

Squore allows using multiple repositories in the same analysis. If your project consists of some code that is spread over two distinct servers or SVN repositories, you can set up your project so that it includes both locations in the project analysis. This is done by labelling each source code node before specifying parameters, as shown below

```
-r "type=FROMPATH,alias=Node1,path=/home/projects/client-code"
-r "type=FROMPATH,alias=Node2,path=/home/projects/common/lib"
```

Note that only alpha-numeric characters are allowed to be used as labels. In the artefact tree, each node will appear as a separate top-level folder with the label provided at project creation.

Using multiple nodes, you can also analyse sources using different Repository Connectors in the same analysis:

```
-r "type=FROMPATH,alias=Node1,path=/home/projects/common-config"
-r "type=SVN,alias=Node2,url=svn+ssh://10.10.0.1/var/svn/project/src,rev=HEAD"
```

Chapter 13. Data Providers

This chapter describe the available Data Providers and the default parameters that they accept via the Command Line Interface.

AntiC

Description

AntiC is a part of the jlint static analysis suite and is launched to analyse C and C++ source code and produce findings.

For more details, refer to http://jlint.sourceforge.net/.

> On Linux, the antiC executable must be compiled manually before you run it for the first time by running the command:
>
> ```
> # cd {squore_home}/addons/tools/Antic_auto/bin/ && gcc antic.c -o antic
> ```

Usage

AntiC has the following options:

- **Source code directory to analyse (`dir`)**: Leave this parameter empty if you want to analyse all sources specified above.

The full command line syntax for AntiC is:

```
-d "type=Antic_auto,dir=[directory]"
```

Automotive Coverage Import

Description

Automotive Coverage Import provides a generic import mechanism for coverage results at function level.

Usage

Automotive Coverage Import has the following options:

- **CSV file (`csv`)**: Enter the path to the CSV containing the coverage data.

  The expected format of each line contained in the file is **PATH;NAME;TESTED_C1;OBJECT_C1;TESTED_MCC;OBJECT_MCC;TESTED_MCDC;OBJECT_MCDC**

The full command line syntax for Automotive Coverage Import is:

```
-d "type=Automotive_Coverage,csv=[file]"
```

Automotive Tag Import

Description

This data provider allows setting values for attributes in the project.

Usage

Automotive Tag Import has the following options:

- **CSV file (`csv`)**: Specify the path to the file containing the metrics.

The full command line syntax for Automotive Tag Import is:

```
-d "type=Automotive_Tag_Import,csv=[file]"
```

BullseyeCoverage Code Coverage Analyzer

Description

BullseyeCoverage is a code coverage analyzer for C++ and C. The coverage report file is used to generate metrics.

For more details, refer to http://www.bullseye.com/.

Usage

BullseyeCoverage Code Coverage Analyzer has the following options:

- **HTML report (`html`)**: Specify the path to the HTML report file generated by BullseyeCoverage.

The full command line syntax for BullseyeCoverage Code Coverage Analyzer is:

```
-d "type=BullseyeCoverage,html=[file]"
```

CANoe

Description

Import data from CANoe XML test results

For more details, refer to https://www.vector.com/int/en/products/products-a-z/software/canoe/.

Usage

CANoe has the following options:

- **Results folder (`dir`)**: Specify the folder containing XML test results files from CANoe.
- **File suffix (`suff`, default: .xml)**: Provide the suffix of CANoe test results files.
- **Create Test artefacts? (`createTests`, default: YES)**: Shoult Test artefacts be created?
- **Test path (`testPath`, default: Tests)**: Define test path (for example Test/HIL Test), by default the value is Tests.

The full command line syntax for CANoe is:

```
-d
"type=CANoe,dir=[directory],suff=[text],createTests=[multipleChoice],testPath=[text]"
```

CPD

Description

CPD is an open source tool which generates Copy/Paste metrics. The dectection of duplicated blocks is set to 100 tokens. CPD provides an XML file which can be imported to generate metrics as well as findings.

For more details, refer to http://pmd.sourceforge.net/pmd-5.3.0/usage/cpd-usage.html.

Usage

CPD has the following options:

- **CPD XML results (`xml`)**: Specify the path to the XML results file generated by CPD. The minimum supported version is PMD/CPD 4.2.5.

The full command line syntax for CPD is:

```
-d "type=CPD,xml=[file]"
```

Cppcheck

Description

Cppcheck is a static analysis tool for C/C++ applications. The tool provides an XML output which can be imported to generate findings.

For more details, refer to http://cppcheck.sourceforge.net/.

Usage

Cppcheck has the following options:

- **Cppcheck XML results (`xml`)**: Specify the path to the XML results file from Cppcheck. Note that the minimum required version of Cppcheck for this data provider is 1.61.

The full command line syntax for Cppcheck is:

```
-d "type=CPPCheck,xml=[file]"
```

Cppcheck (plugin)

Description

Cppcheck is a static analysis tool for C/C++ applications. The tool provides an XML output which can be imported to generate findings.

For more details, refer to http://cppcheck.sourceforge.net/.

On Windows, this data provider requires an extra download to extract the Cppcheck binary in *<SQUORE_HOME>/addons/tools/CPPCheck_auto/* and the MS Visual C++ 2010 Redistributable Package available from http://www.microsoft.com/en-in/download/details.aspx?id=5555. On Linux, you can install the cppcheck application anywhere you want. The path to the Cppcheck binary for Linux can be configured in *config.tcl*. For more information, refer to the Installation and Administration Guide's 'Third-Party Plugins and Applications' section.

Usage

Cppcheck (plugin) has the following options:

- **Source code folder (`dir`)**: Specify the folder containing the source files to analyse. If you want to analyse all of source repositories specified for the project, leave this field empty.

- **Ignore List (`ignores`)**: Specify a semi-colon-separated list of source files or source file directories to exclude from the check. For example: "lib/;folder2/". Leave this field empty to deactivate this option and analyse all files with no exception.

The full command line syntax for Cppcheck (plugin) is:

```
-d "type=CPPCheck_auto,dir=[directory],ignores=[text]"
```

CPPTest

Description

Parasoft C/Ctest is an integrated solution for automating a broad range of best practices proven to improve software development team productivity and software quality for C and C. The tool provides an XML output file which can be imported to generate findings and metrics.

For more details, refer to http://www.parasoft.com/product/cpptest/.

Usage

CPPTest has the following options:

- **Directory which contains the XML results files (`results_dir`)**: Specify the path to the CPPTest results directory. This data provider is compatible with files exported from CPPTest version 7.2.10.34 and up.
- **Results file extensions (`pattern`, default: *.xml)**: Specify the pattern of the results files

The full command line syntax for CPPTest is:

```
-d "type=CPPTest,results_dir=[directory],pattern=[text]"
```

Cantata

Description

Cantata is a Test Coverage tool. It provides an XML output file which can be imported to generate coverage metrics at function level.

For more details, refer to http://www.qa-systems.com/cantata.html.

Usage

Cantata has the following options:

- **Cantata XML results (`xml`)**: Specify the path to the XML results file from Cantata 6.2

The full command line syntax for Cantata is:

```
-d "type=Cantata,xml=[file]"
```

CheckStyle

Description

CheckStyle is an open source tool that verifies that Java applications adhere to certain coding standards. It produces an XML file which can be imported to generate findings.

For more details, refer to http://checkstyle.sourceforge.net/.

Usage

CheckStyle has the following options:

- **CheckStyle results file (`xml`)**: Point to the XML file that contains Checkstyle results. Note that the minimum supported version is Checkstyle 5.3.

The full command line syntax for CheckStyle is:

```
-d "type=CheckStyle,xml=[file]"
```

CheckStyle (plugin)

Description

CheckStyle is an open source tool that verifies that Java applications adhere to certain coding standards. It produces an XML file which can be imported to generate findings.

For more details, refer to http://checkstyle.sourceforge.net/.

This data provider requires an extra download to extract the CheckStyle binary in *<SQUORE_HOME>/addons/tools/CheckStyle_auto/*. For more information, refer to the Installation and Administration Guide's 'Third-Party Plugins and Applications' section. You may also deploy your own version of CheckStyle and make the Data Provider use it by editing *<SQUORE_HOME>/configuration/tools/CheckStyle_auto/config.tcl*.

Usage

CheckStyle (plugin) has the following options:

- **Configuration file (`configFile`)**: A Checkstyle configuration specifies which modules to plug in and apply to Java source files. Modules are structured in a tree whose root is the Checker module. Specify the name of the configuration file only, and the data provider will try to find it in the CheckStyle_auto folder of your custom configuration. If no custom configuration file is found, a default configuration will be used.

- **Xmx (`xmx`, default: 1024m)**: Maximum amount of memory allocated to the java process launching Checkstyle.

- **Excluded directory pattern (`excludedDirectoryPattern`)**: Java regular expression of directories to exclude from CheckStyle, for example: ^test|generated-sources|.*-report$ or ou ^lib$

The full command line syntax for CheckStyle (plugin) is:

```
-d "type=CheckStyle_auto,configFile=[text],xmx=[text],excludedDirectoryPattern=[text]"
```

CheckStyle for SQALE (plugin)

Description

CheckStyle is an open source tool that verifies that Java applications adhere to certain coding standards. It produces an XML file which can be imported to generate findings.

For more details, refer to http://checkstyle.sourceforge.net/.

> This data provider requires an extra download to extract the CheckStyle binary in *<SQUORE_HOME>/addons/tools/CheckStyle_auto_for_SQALE/*. For more information, refer to the Installation and Administration Guide's 'Third-Party Plugins and Applications' section.

Usage

CheckStyle for SQALE (plugin) has the following options:

- **Configuration file (`configFile`, default: config_checkstyle_for_sqale.xml)**: A Checkstyle configuration specifies which modules to plug in and apply to Java source files. Modules are structured in a tree whose root is the Checker module. Specify the name of the configuration file only, and the data provider will try to find it in the CheckStyle_auto folder of your custom configuration. If no custom configuration file is found, a default configuration will be used.

- **Xmx (`xmx`, default: 1024m)**: Maximum amount of memory allocated to the java process launching Checkstyle.

The full command line syntax for CheckStyle for SQALE (plugin) is:

```
-d "type=CheckStyle_auto_for_SQALE,configFile=[text],xmx=[text]"
```

Cobertura format

Description

Cobertura is a free code coverage library for Java. Its XML report file can be imported to generate code coverage metrics for your Java project.

For more details, refer to http://cobertura.github.io/cobertura/.

Usage

Cobertura format has the following options:

- **XML report (`xml`)**: Specify the path to the XML report generated by Cobertura (or by a tool able to produce data in this format).

The full command line syntax for Cobertura format is:

```
-d "type=Cobertura,xml=[file]"
```

CodeSonar

Description

Codesonar is a static analysis tool for C and C++ code designed for zero tolerance defect environments. It provides an XML output file which is imported to generate findings.

For more details, refer to http://www.grammatech.com/codesonar.

Usage

CodeSonar has the following options:

- **XML results file (`xml`)**: Specify the path to the XML results file generated by Codesonar. The minimum version of Codesonar compatible with this data provider is 3.3.

The full command line syntax for CodeSonar is:

```
-d "type=CodeSonar,xml=[file]"
```

Compiler

Description

Compiler allows to import information from compiler logs.

Usage

Compiler has the following options:

- **Compiler output file(s) (`txt`, mandatory)**: Specify the path(s) to CSV compiler log file(s). To provide multiple files click on '+'.

  Each line needs to match the following format: **Path;Line;Rule;Descr** where Rule is one of COMP_ERR, COMPILER_WARN or COMPILER_INFO.

The full command line syntax for Compiler is:

```
-d "type=Compiler,txt=[file]"
```

Coverity

Description

Coverity is a static analysis tool for C, C++, Java and C#. It provides an XML output which can be imported to generate findings.

For more details, refer to http://www.coverity.com/.

Usage

Coverity has the following options:

- **XML results file (`xml`)**: Specify the path to the XML file containing Coverity results.

The full command line syntax for Coverity is:

```
-d "type=Coverity,xml=[file]"
```

ESLint

Description

ESLint is an open source tool that verifies that JavaScript applications adhere to certain coding standards. It produces an XML file which can be imported to generate findings.

For more details, refer to https://eslint.org/.

Usage

ESLint has the following options:

- **ESLint results file (`xml`)**: Point to the XML file that contains ESLint results in Checkstyle format.

The full command line syntax for ESLint is:

```
-d "type=ESLint,xml=[file]"
```

FindBugs-SpotBugs

Description

Findbugs (and its successor SpotBugs) is an open source tool that looks for bugs in Java code. It produces an XML result file which can be imported to generate findings.

For more details, refer to http://findbugs.sourceforge.net/.

Usage

FindBugs-SpotBugs has the following options:

- **XML results file (`xml`)**: Specify the location of the XML file containing Findbugs results. Note that the minimum supported version for FindBugs is 1.3.9, and 3.1.7 to 3.1.12 for SpotBugs.

The full command line syntax for FindBugs-SpotBugs is:

```
-d "type=Findbugs,xml=[file]"
```

FindBugs-SpotBugs (plugin)

Description

FindBugs is an open source tool that looks for bugs in Java code. It produces an XML result file which can be imported to generate findings. Note that the data provider requires an extra download to extract the FindBugs binary in [INSTALLDIR]/addons/tools/Findbugs/. You are free to use FindBugs 3.0 or FindBugs 2.0 depending on what your standard is. For more information, refer to the Installation and Administration Manual's "Third-Party Plugins and Applications" section.This Data Provider also supports SpotBugs (successor to FindBugs), with the same parameters. If you are using SpotBugs, its binary also has to be accessible, in [INSTALLDIR]/addons/tools/Findbugs/.

For more details, refer to http://findbugs.sourceforge.net/.

> This data provider requires an extra download to extract the FindBugs or SpotBugs binary in *<SQUORE_HOME>/addons/tools/Findbugs_auto/*. For more information, refer to the Installation and Administration Guide's 'Third-Party Plugins and Applications' section.

Usage

FindBugs-SpotBugs (plugin) has the following options:

- **Classes (`class_dir`, mandatory)**: Specify the folders and/or jar files for your project in classpath format, or point to a text file that contains one folder or jar file per line.

- **Auxiliary Class path (`auxiliarypath`)**: Specify a list of folders and/or jars in classpath format, or specify the path to a text file that contains one folder or jar per line. This information will be passed to FindBugs or SpotBugs via the -auxclasspath parameter.

- **Memory Allocation (`xmx`, default: 1024m)**: Maximum amount of memory allocated to the java process launching FindBugs or SpotBugs.

The full command line syntax for FindBugs-SpotBugs (plugin) is:

```
-d
"type=Findbugs_auto,class_dir=[file_or_directory],auxiliarypath=[file_or_directory],xm
x=[text]"
```

Function Relaxer

Description

Function Relaxer provides a generic import mechanism for relaxing functions in source code.

Usage

Function Relaxer has the following options:

- **CSV File (`csv`)**:

The full command line syntax for Function Relaxer is:

```
-d "type=Function_Relaxer,csv=[file]"
```

FxCop

Description

FxCop is an application that analyzes managed code assemblies (code that targets the .NET Framework common language runtime) and reports information about the assemblies, such as possible design, localization, performance, and security improvements. FxCop generates an XML results file which can be imported to generate findings.

For more details, refer to https://msdn.microsoft.com/en-us/library/bb429476(v=vs.80).aspx.

Usage

FxCop has the following options:

- **XML results file (`xml`)**: Specify the XML file containing FxCop's analysis results. Note that the minimum supported version of FxCop is 1.35.

The full command line syntax for FxCop is:

```
-d "type=FxCop,xml=[file]"
```

GCov

Description

GCov is a Code coverage program for C application. GCov generates raw text files which can be imported to generate metrics.

For more details, refer to http://gcc.gnu.org/onlinedocs/gcc/Gcov.html.

Usage

GCov has the following options:

- **Directory containing results files (`dir`)**: Specify the path of the root directory containing the GCov results files.

- **Results files extension (`ext`, default: *.c.gcov)**: Specify the file extension of GCov results files.

The full command line syntax for GCov is:

```
-d "type=GCov,dir=[directory],ext=[text]"
```

GNATcheck

Description

GNATcheck is an extensible rule-based tool that allows developers to completely define a coding standard. The results are output to a log file or an XML file that can be imported to generate findings.

For more details, refer to http://www.adacore.com/gnatpro/toolsuite/gnatcheck/.

Usage

GNATcheck has the following options:

- **Log or XML file (`txt`)**: Specify the path to the log file or the XML file generated by the GNATcheck run.

The full command line syntax for GNATcheck is:

```
-d "type=GnatCheck,txt=[file]"
```

GNATCompiler

Description

GNATCompiler is a free-software compiler for the Ada programming language which forms part of the GNU Compiler Collection. It supports all versions of the language, i.e. Ada 2012, Ada 2005, Ada 95 and Ada 83. It creates a log file that can be imported to generate findings.

For more details, refer to http://www.adacore.com/gnatpro/toolsuite/compilation/.

Usage

GNATCompiler has the following options:

- **Log file (`log`)**: Specify the path to the log file containing the compiler warnings.

The full command line syntax for GNATCompiler is:

```
-d "type=GnatCompiler,log=[file]"
```

JSHint

Description

JSHint is an open source tool that verifies that JavaScript applications adhere to certain coding standards. It produces an XML file which can be imported to generate findings.

For more details, refer to http://jshint.com/.

Usage

JSHint has the following options:

- **JSHint results file (Checkstyle formatted): (`xml`)**: Point to the XML file that contains JSHint results Checkstyle formatted.

The full command line syntax for JSHint is:

```
-d "type=JSHint,xml=[file]"
```

JUnit Format

Description

JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. JUnit XML result files are imported as test artefacts and links to tested classes are generated in the project.

For more details, refer to http://junit.org/.

Usage

JUnit Format has the following options:

- **Results folder (`resultDir`, mandatory)**: Specify the path to the folder containing the JUnit results (or by a tool able to produce data in this format). The data provider will parse subfolders recursively. Note that the minimum support version of JUnit is 4.10.

- **File Pattern (`filePattern`, mandatory, default: TEST-**.xml)*: Specify the pattern for files to read reports from.

- **Root Artefact (`root`, mandatory, default: tests[type=TEST_FOLDER]/junit[type=TEST_FOLDER])**: Specify the name and type of the artefact under which the test artefacts will be created.

The full command line syntax for JUnit Format is:

```
-d "type=JUnit,resultDir=[directory],filePattern=[text],root=[text]"
```

JaCoCo

Description

JaCoCo is a free code coverage library for Java. Its XML report file can be imported to generate code coverage metrics for your Java project.

For more details, refer to http://www.eclemma.org/jacoco/.

Usage

JaCoCo has the following options:

- **XML report (`xml`, mandatory)**: Specify the path to the XML report generated by JaCoCo. Note that the folder containing the XML file must also contain JaCoCo's report DTD file, available from http://www.eclemma.org/jacoco/trunk/coverage/report.dtd. XML report files are supported from version 0.6.5.

The full command line syntax for JaCoCo is:

```
-d "type=Jacoco,xml=[file]"
```

Klocwork

Description

Klocwork is a static analysis tool. Its XML result file can be imported to generate findings.

For more details, refer to http://www.klocwork.com.

Usage

Klocwork has the following options:

- **XML results file (`xml`)**: Specify the path to the XML results file exported from Klocwork. Note that Klocwork version 9.6.1 is the minimum required version.

The full command line syntax for Klocwork is:

```
-d "type=Klocwork,xml=[file]"
```

Klocwork MISRA

Description

Klocwork is a static analysis tool. Its XML result file can be imported to generate findings.

For more details, refer to http://www.klocwork.com.

Usage

Klocwork MISRA has the following options:

- **XML results file (`xml`)**: Specify the path to the XML results file exported from Klocwork. Note that Klocwork version 9.6.1 is the minimum required version.

The full command line syntax for Klocwork MISRA is:

```
-d "type=Klocwork_misra,xml=[file]"
```

Rational Logiscope

Description

The Logiscope suite allows the evaluation of source code quality in order to reduce maintenance cost, error correction or test effort. It can be applied to verify C, C++, Java and Ada languages and produces a CSV results file that can be imported to generate findings.

For more details, refer to http://www.kalimetrix.com/en/logiscope.

Usage

Rational Logiscope has the following options:

- **RuleChecker results file (`csv`)**: Specify the path to the CSV results file from Logiscope.

The full command line syntax for Rational Logiscope is:

```
-d "type=Logiscope,csv=[file]"
```

MSTest

Description

MS-Test automates the process of testing Windows applications. It combines a Windows development language, Basic, with a testing-oriented API.

For more details, refer to https://en.wikipedia.org/wiki/Visual_Test.

Usage

MSTest has the following options:

- **MSTest results directory (`resultDir`)**: Specify the path to the results directory generated by MSTest.

- **Test result file pattern (`filePattern`)**: Specify the pattern of files to extract Test data from.

The full command line syntax for MSTest is:

```
-d "type=MSTest,resultDir=[directory],filePattern=[text]"
```

MSTest Code Coverage

Description

MSTest is a code coverage library for C#. Its XML report file can be imported to generate code coverage metrics for your C# project.

Usage

MSTest Code Coverage has the following options:

- **XML report (`xml`)**: Specify the path to the XML report generated by MSTest Visual Studio 2017.

The full command line syntax for MSTest Code Coverage is:

```
-d "type=MSTest_Coverage,xml=[file]"
```

MemUsage

Description

Usage

MemUsage has the following options:

- **Memory Usage excel file (`excel`)**:

The full command line syntax for MemUsage is:

```
-d "type=MemUsage,excel=[file]"
```

NCover

Description

NCover is a Code coverage program for C# application. NCover generates an XML results file which can be imported to generate metrics.

For more details, refer to http://www.ncover.com/.

Usage

NCover has the following options:

- **XML results file (`xml`)**: Specify the location of the XML results file generated by NCover. Note that the minimum supported version is NCover 3.0.

The full command line syntax for NCover is:

```
-d "type=NCover,xml=[file]"
```

Oracle PLSQL compiler Warning checker

Description

This data provider reads an Oracle compiler log file and imports the warnings as findings. Findings extracted from the log file are filtered using a prefix parameter.

For more details, refer to http://www.oracle.com/.

Usage

Oracle PLSQL compiler Warning checker has the following options:

- **Compiler log file (`log`)**:
- **Prefixes (`prefix`)**: Prefixes and their replacements are specified as pairs using the syntax [prefix1|node1;prefix2|node2]. Leave this field empty to disable filtering.

  The parsing algorithm looks for lines fitting this pattern:

  [PATH;SCHEMA;ARTE_ID;ARTE_TYPE;LINE;COL;SEVERITY_TYPE;WARNING_ID;SEVERITY_ID;DESCR] and keeps lines where [PATH] begins with one of the input prefixes. In each kept [PATH], [prefix] is replaced by [node]. If [node] is empty, [prefix] is removed from [PATH], but not replaced. Some valid syntaxes for prefix:

  One prefix to remove: svn://aaaa:12345/valid/path/from/svn

  One prefix to replace: svn://aaaa:12345/valid/path/from/svn|node1

  Two prefixes to remove: svn://aaaa:12345/valid/path/from/svn|;svn://bbbb:12345/valid/path/from/other_svn|

  Two prefixes to remove: svn://aaaa:12345/valid/path/from/svn;svn://bbbb:12345/valid/path/from/other_svn

  Two prefixes to replace: svn://aaaa:12345/valid/path/from/svn|node1;svn://bbbb:12345/valid/path/from/other_svn|node2

The full command line syntax for Oracle PLSQL compiler Warning checker is:

```
-d "type=Oracle_PLSQLCompiler,log=[file],prefix=[text]"
```

MISRA Rule Checking using PC-lint

Description

PC-lint is a static code analyser. The PC-lint data provider reads PC-lint log file(s) and imports MISRA violations as findings.

For more details, refer to http://www.gimpel.com/html/pcl.htm.

Usage

MISRA Rule Checking using PC-lint has the following options:

- **Log file or folder (`logDir`)**: Specify the path to the folder containing the PC-lint log files, or to a single log file.
- **Extensions to exclude (`excludedExtensions`, default: .h;.H)**: Specify the file extensions to exclude from the reported violations.

The full command line syntax for MISRA Rule Checking using PC-lint is:

```
-d "type=PC_Lint_MISRA,logDir=[file_or_directory],excludedExtensions=[text]"
```

PMD

Description

PMD scans Java source code and looks for potential problems like possible bugs, dead code, sub-optimal code, overcomplicated expressions, duplicate code... The XML results file it generates is read to create findings.

For more details, refer to http://pmd.sourceforge.net.

Usage

PMD has the following options:

- **XML results file (`xml`)**: Specify the path to the PMD XML results file. Note that the minimum supported version of PMD for this data provider is 4.2.5.

The full command line syntax for PMD is:

```
-d "type=PMD,xml=[file]"
```

PMD (plugin)

Description

PMD scans Java source code and looks for potential problems like possible bugs, dead code, sub-optimal code, overcomplicated expressions, duplicate code ... The XML results file it generates is read to create findings.

For more details, refer to http://pmd.sourceforge.net.

This data provider requires an extra download to extract the PMD binary in *<SQUORE_HOME>/addons/tools/PMD_auto/*. For more information, refer to the Installation and Administration Guide's 'Third-Party Plugins and Applications' section. You may also deploy your own version of PMD and make the Data Provider use it by editing *<SQUORE_HOME>/configuration/tools/PMD_auto/config.tcl*.

Usage

PMD (plugin) has the following options:

- **Ruleset file (`configFile`)**: Specify the path to the PMD XML ruleset you want to use for this analysis. If you do not specify a ruleset, the default one from INSTALLDIR/addons/tools/PMD_auto will be used.

The full command line syntax for PMD (plugin) is:

```
-d "type=PMD_auto,configFile=[file]"
```

Polyspace

Description

Polyspace is a static analysis tool which includes a MISRA checker. It produces an XML output which can be imported to generate findings. Polyspace Verifier detects RTE (RunTime Error) such as Division by zero, Illegal Deferencement Pointer, Out of bound array index… Such information is turned into statistical measures at function level. Number of Red (justified/non-justified), Number of Grey (justified/non-justified), Number of Orange (justified/non-justified), Number of Green.

For more details, refer to http://www.mathworks.com/products/polyspace/index.html.

Usage

Polyspace has the following options:

- **DocBook results file (`xml`)**: Specify the path to the DocBook (main XML file) generated by Polyspace.

- **Ignore source file path (`ignoreSourceFilePath`, default: false)**: Removes all path elements when doing the mapping between files in Squore project and files in the Pomyspace report. Becareful this can work only if file names in Squore project are unique.

The full command line syntax for Polyspace is:

```
-d "type=Polyspace,xml=[file],ignoreSourceFilePath=[booleanChoice]"
```

MISRA Rule Checking with QAC

Description

QAC identifies problems in C source code caused by language usage that is dangerous, overly complex, non-portable, difficult to maintain, or simply diverges from coding standards. Its CSV results file can be imported to generate findings.

For more details, refer to http://www.phaedsys.com/principals/programmingresearch/pr-qac.html.

Usage

MISRA Rule Checking with QAC has the following options:

- **Code Folder (`logDir`)**: Specify the path to the folder that contains the annotated files to process.

  For the findings to be successfully linked to their corresponding artefact, several requirements have to be met:

  - The annotated file name should be [Original source file name].txt

  e.g. The annotation of file "controller.c" should be called "controller.c.txt"

  - The annotated file location in the annotated directory should match the associated source file location in the source directory.

  e.g. The annotation for source file "[SOURCE_DIR]/subDir1/subDir2/controller.c" should be located in "[ANNOTATIONS_DIR]/subDir1/subDir2/controller.c.txt"

  The previous comment suggests that the source and annotated directory are different.

  However, these directories can of course be identical, which ensures that locations of source and annotated files are the same.

- **Extension (`ext`, default: html)**: Specify the extension used by QAC to create annotated files.

- **Force import of all QAC violations (not only MISRA) (`force_all_import`, default: false)**: Force the import of all QAC findings (not only the MISRA violations)

The full command line syntax for MISRA Rule Checking with QAC is:

```
-d "type=QAC_MISRA,logDir=[directory],ext=[text],force_all_import=[booleanChoice]"
```

Rational Test RealTime

Description

Rational Test RealTime is a cross-platform solution for component testing and runtime analysis of embedded software. This Data Provider extracts coverage results, as well as tests and their status

For more details, refer to http://www-01.ibm.com/software/awdtools/test/realtime/.

Usage

Rational Test RealTime has the following options:

- **.xrd folder (logDir)**: Specify the path to the folder containing the .xrd files generated by RTRT.

- **Excluded file extensions (excludedExtensions, default: .h;.H)**:

- **Do you want to include FE (Function and Exit) in MCDC computation? (include_fe_in_mcdc, default: false)**:

- **Generate Test artefacts and structure from .xrd files? (generateTests, default: false)**:

The full command line syntax for Rational Test RealTime is:

```
-d
"type=RTRT,logDir=[directory],excludedExtensions=[text],include_fe_in_mcdc=[booleanCho
ice],generateTests=[booleanChoice]"
```

ReqIF

Description

RIF/ReqIF (Requirements Interchange Format) is an XML file format that can be used to exchange requirements, along with its associated metadata, between software tools from different vendors.

For more details, refer to http://www.omg.org/spec/ReqIF/.

Usage

ReqIF has the following options:

- **Reqif Directory (dir)**: Specify the directory which contains the Reqif files

- **Spec Object Type (objType, default: *AUTO*)**: Specify the SPEC_OBJECT_TYPE property LONG-NAME to be used to process the ReqIf file. Using the _AUTO_ value will let the Data Provider extract the value fro the ReqIf file, and assumes that there is only one such definition.

The full command line syntax for ReqIF is:

```
-d "type=ReqIf,dir=[directory],objType=[text]"
```

SQL Code Guard

Description

SQL Code Guard is a free solution for SQL Server that provides fast and comprehensive static analysis for T-Sql code, shows code complexity and objects dependencies.

For more details, refer to http://www.sqlcodeguard.com.

Usage

SQL Code Guard has the following options:

- **XML results (`xml`)**: Specify the path to the XML file containing SQL Code Guard results.

The full command line syntax for SQL Code Guard is:

```
-d "type=SQLCodeGuard,xml=[file]"
```

Squan Sources

Description

Squan Sources provides basic-level analysis of your source code.

For more details, refer to https://www.vector.com/squore.

The analyser can output info and warning messages in the build logs. Recent additions to those logs include better handling of structures in C code, which will produce these messages:

- [Analyzer] Unknown syntax declaration for function XXXXX at line yyy to indicate that we whould have found a function but, probably due to preprocessing directives, we are not able to parse it.
- [Analyzer] Unbalanced () blocks found in the file. Probably due to preprocessing directives, parenthesis in the file are not well balanced.
- [Analyzer] Unbalanced {} blocks found in the file. Probably due to preprocessing directives, curly brackets in the file are not well balanced.

You can specify the languages for your source code by passing pairs of language and extensions to the `languages` parameter. Extensions are case-sensitive and cannot be used for two different languages. For example, a project mixing php and javascript files can be analysed with:

```
--dp "type=SQuORE,languages=php:.php;javascript:.js,.JS"
```

In order to launch an analysis using all the available languages by default, do not specify the `languages` parameter in your command line.

Usage

Squan Sources has the following options:

- **Languages (`languages`, default: ada;c;cpp;csharp;cobol;java;fortran77;fortran90;php;python;swift;vbnet)**: Check the boxes for the languages used in the specified source repositories. Adjust the list of file extensions as necessary. Note that two languages cannot use the same file extension, and that the list of extensions is case-sensitive. Tip: Leave all the boxes unchecked and Squan Sources will auto-detect the language parser to use.
- **Force full analysis (`rebuild_all`, default: false)**: Analyses are incremental by default. Check this box if you want to force the source code parser to analyse all files instead of only the ones

that have changed since the previous analysis. This is useful if you added new rule files or text parsing rules and you want to re-evaluate all files based on your modifications.

- **Generate control graphs (`genCG`, default: true)**: This option allows generating a control graph for every function in your code. The control graph is visible in the dashboard of the function when the analysis completes.
- **Use qualified names (`qualified`, default: false)**: Note: This option cannot be modified in subsequent runs after you create the first version of your project.
- **Limit analysis depth (`depth`, default: false)**: Use this option to limit the depth of the analysis to file-level only. This means that Squan Sources will not create any class or function artefacts for your project.
- **Add a 'Source Code' node (`scnode`, default: false)**: Using this options groups all source nodes under a common source code node instead of directly under the APPLICATION node. This is useful if other data providers group non-code artefacts like tests or requirements together under their own top-level node. This option can only be set when you create a new project and cannot be modified when creating a new version of your project.
- **'Source Code' node label (`scnode_name`, default: Source Code)**: Specify a custom label for your main source code node. Note: this option is not modifiable. It only applies to projects where you use the "Add a 'Source Code' node" option. When left blank, it defaults to "Source Code".
- **Compact folders (`compact_folder`, default: true)**: When using this option, folders with only one son are aggregates together. This avoids creating many unnecessary levels in the artefact tree to get to the first level of files in your project. This option cannot be changed after you have created the first version of your project.
- **Content exclusion via regexp (`pattern`)**: Specify a PERL regular expression to automatically exclude files from the analysis if their contents match the regular expression. Leave this field empty to disable content-based file exclusion.
- **File Filtering (`files_choice`, default: Exclude)**: Specify a pattern and an action to take for matching file names. Leave the pattern empty to disable file filtering.
- **pattern (`pattern_files`)**: Use a shell-like wildcard e.g. '*-test.c'.

  - * Matches any sequence of characters in string, including a null string.
  - ? Matches any single character in string.
  - [chars] Matches any character in the set given by chars. If a sequence of the form x-y appears in chars, then any character between x and y, inclusive, will match. On Windows, this is used with the -nocase option, meaning that the end points of the range are converted to lower case first. Whereas [A-z] matches '_' when matching case-sensitively ('_' falls between the 'Z' and 'a'), with -nocase this is considered like [A-Za-z].
  - \x Matches the single character x. This provides a way of avoiding the special interpretation of the characters *?[] in pattern.

  Tip: Use ';' to separate multiple patterns.

  How to specify a file:

  - By providing its name, containing or not a pattern
  - By providing its name and its path, both containing or not a pattern

  e.g.

  - *D??l?g.* : will match MyDialog.java, WinDowlog.c, … anywhere in the project
  - */[Dd]ialog/*D??l?g.* : will match src/java/Dialog/MyDialog.java, src/c/dialog/WinDowlog.c, but not src/Dlg/c/WinDowlog.c

- **Folder Filtering (`dir_choice`, default: Exclude)**: Specify a pattern and an action to take for matching folder names. Leave the pattern empty to disable folder filtering.
- **pattern (`pattern_dir`)**: Use a shell-like wildcard e.g. 'Test_*'.
  - * Matches any sequence of characters in string, including a null string.
  - ? Matches any single character in string.
  - [chars] Matches any character in the set given by chars. If a sequence of the form x-y appears in chars, then any character between x and y, inclusive, will match. On Windows, this is used with the -nocase option, meaning that the end points of the range are converted to lower case first. Whereas [A-z] matches '_' when matching case-sensitively ('_' falls between the 'Z' and 'a'), with -nocase this is considered like [A-Za-z].
  - \x Matches the single character x. This provides a way of avoiding the special interpretation of the characters *?[] in pattern.

  Tip: Use ';' to separate multiple patterns.

  A directory can be specified:

  - By providing its name, containing or not a pattern
  - By providing its name and its path, both containing or not a pattern. In that case the full path has to match.

  e.g.

  - source? : will match directories source, sources, ... anywhere in the project
  - src/tests : will not match any directory because the full path can not match
  - */src/tests : will match java/src/tests, native/c/src/tests, ...

  To get the root path of the project it is possible to use the nodes variables ($src, $Node1, ...). Refers to "Using Data Provider Input Files From Version Control" in the Getting Started to learn more.

  e.g. $src/source/tests will match only the directory source/tests if it is a root directory of the project.
- **Exclude files whose size exceeds (`size_limit`, default: 500000)**: Provide the size in bytes above which files are excluded automatically from the Squore project (Big files are usually generated files or test files). Leave this field empty to deactivate this option.
- **Detect algorithmic cloning (`clAlg`, default: true)**: When checking this box, Squan Sources launches a cloning detection tool capable of finding algorithmic cloning in your code.
- **Detect text cloning (`clTxt`, default: true)**: When checking this box, Squan Sources launches a cloning detection tool capable of finding text duplication in your code.
- **Ignore blank lines (`clIgnBlk`, default: true)**: When checking this box, blanks lines are ignored when searching for text duplication
- **Ignore comment blocks (`clIgnCmt`, default: true)**: When checking this box, blocks of comments are ignored when searching for text duplication
- **Minimum size of duplicated blocks (`clRSlen`, default: 10)**: This threshold defines the minimum size (number of lines) of blocks that can be reported as cloned.
- **Textual Cloning fault ratio (`clFR`, default: 0.1)**: This threshold defines how much cloning between two artefacts is necessary for them to be considered as clones by the text duplication tool. For example, a fault ratio of 0.1 means that two artefacts are considered clones if less than 10% of their contents differ.

- **Algorithmic cloning fault ratio (`clAlgFR`, default: 0.1)**: This threshold defines how much cloning between two artefacts is necessary for them to be considered as clones by the algorithmic cloning detection tool.

- **Compute Textual stability (`genTs`, default: true)**: This option allows keeping track of the stability of the code analysed for each version. The computed stability is available on the dashboard as a metric called and can be interpreted as 0% meaning completely changed and 100% meaning not changed at all.

- **Compute Algorithmic stability (`genAs`, default: true)**: This option allows keeping track of the stability of the code analysed for each version. The computed stability is available on the dashboard as a metric called Stability Index (SI) and can be interpreted as 0% meaning completely changed and 100% meaning not changed at all.

- **Detect artefact renaming (`clRen`, default: true)**: This option allows Squan Sources to detect artefacts that have been moved since the previous version, ensuring that the stability metrics of the previous artefact are passed to the new one. This is typically useful if you have moved a file to a different folder in your source tree and do not want to lose the previous metrics generated for this file. If you do not use this option, moved artefacts will be considered as new artefacts.

- **Mark relaxed or confirmed findings as suspicious (`susp`, default: MODIFIED_BEFORE)**: This option sets the suspicious flag on relaxed or confirmed findings depending of the selected option. Applies on source code artifacts only.

- **Accept Relaxation from source code comment (`relax`, default: true)**: **Relaxing Violations in Code**

  Squore interprets comments formatted in one of these three ways:

  - Inline Relaxation

  This syntax is used to relax violations on the current line.

  some code; /* **%RELAX<keys>** : Text to justify the relaxation */

  &nbsp;

  - Relax Next Line

  This syntax is used to relax a violation on the first following line that is not a comment. In the example the text of the justification will be: "Text to justify the relaxation the text of the justification continues while lines are made of comments only"

  /* **>RELAX<keys>** : Text to justify the relaxation */

  /* the text of the justification continues while */

  /* lines are made of comments only */

  some code;

  &nbsp;

  - Block Relaxation

  This syntax is used to relax violations in an entire block of code.

  /* **{{ RELAX<keys>** : Text to justify the relaxation */

  /* like for format 2 text can be on more than one line */

int my_func() {

&nbsp;&nbsp;&nbsp;/* contains many violations */

&nbsp;&nbsp;&nbsp;&#8230;&#8203;

}

/* **}} RELAX<keys>** */

<keys> can be one of the following:

- <*>: relax all violations
- <MNEMO>: relax violations of the rule MNEMO
- <MNEMO1,MNEMO2,...,MNEMOn>: relax violations of rules MNEMO1 and MNEMO2 ... and MNEMOn

- **Additional parameters (`additional_param`)**: These additional parameters can be used to pass instructions to external processes started by this data provider. This value is generally left empty in most cases.

The full command line syntax for Squan Sources is:

```
-d
"type=SQuORE,languages=[multipleChoice],rebuild_all=[booleanChoice],genCG=[booleanChoi
ce],qualified=[booleanChoice],depth=[booleanChoice],scnode=[booleanChoice],scnode_name
=[text],compact_folder=[booleanChoice],pattern=[text],files_choice=[multipleChoice],pa
ttern_files=[text],dir_choice=[multipleChoice],pattern_dir=[text],size_limit=[text],cl
Alg=[booleanChoice],clTxt=[booleanChoice],clIgnBlk=[booleanChoice],clIgnCmt=[booleanCh
oice],clRSlen=[text],clFR=[text],clAlgFR=[text],genTs=[booleanChoice],genAs=[booleanCh
oice],clRen=[booleanChoice],susp=[multipleChoice],relax=[booleanChoice],additional_par
am=[text]"
```

Squore Import

Description

Squore Import is a data provider used to import the results of another data provider analysis. It is generally only used for debugging purposes.

For more details, refer to support@vector.com.

Usage

Squore Import has the following options:

- **XML folder (`inputDir`)**: Specify the folder that contains the squore_data_*.xml files that you want to import.

The full command line syntax for Squore Import is:

```
-d "type=SQuOREImport,inputDir=[directory]"
```

Squore Virtual Project

Description

Squore Virtual Project is a data provider that can use the output of several projects to compile metrics in a meta-project composed of the import sub-projects.

For more details, refer to support@vector.com.

Usage

Squore Virtual Project has the following options:

- **Paths to output.xml files (`output`)**: Specify the paths to all the output.xml files you want to include in the virtual project. Separate paths using ';'.

The full command line syntax for Squore Virtual Project is:

```
-d "type=SQuOREVirtualProject,output=[file]"
```

StyleCop

Description

StyleCop is a C# code analysis tool. Its XML output is imported to generate findings.

For more details, refer to https://stylecop.codeplex.com/.

Usage

StyleCop has the following options:

- **XML results file (`xml`)**: Specify the path to the StyleCop XML results file. The minimum version compatible with this data provider is 4.7.

The full command line syntax for StyleCop is:

```
-d "type=StyleCop,xml=[file]"
```

StyleCop (plugin)

Description

StyleCop is a C# code analysis tool. Its XML output is imported to generate findings.

For more details, refer to https://stylecop.codeplex.com/.

> Note that this data provider is not supported on Linux. On windows, this data provider requires an extra download to extract the StyleCop binary in *<SQUORE_HOME>/addons/tools/StyleCop_auto/* and .NET framework 3.5 to be installed on your machine (run *Net.SF.StyleCopCmd.Console.exe* manually once to install .NET automatically). For more information, refer to the Installation and Administration Guide's 'Third-Party Plugins and Applications' section.

Usage

StyleCop (plugin) has the following options:

- **Solution (`sln`)**: Specify the path to the .sln file to analyse. Leave empty to analyse all .sln found in the source repository.

The full command line syntax for StyleCop (plugin) is:

```
-d "type=StyleCop_auto,sln=[file]"
```

Tessy

Description

Tessy is a tool automating module/unit testing of embedded software written in dialects of C/C++. Tessy generates an XML results file which can be imported to generate metrics. This data provider supports importing files that have a xml_version="1.0" attribute in their header.

For more details, refer to https://www.hitex.com/en/tools/tessy/.

Usage

Tessy has the following options:

- **Results folder (`resultDir`)**: Specify the top folder containing XML result files from Tessy. Note that this data provider will recursively scan sub-folders looking for index.xml files to aggregate results.

The full command line syntax for Tessy is:

```
-d "type=Tessy,resultDir=[directory]"
```

VectorCAST

Description

The VectorCAST Data Provider extracts coverage results, as well as tests and their status

For more details, refer to https://www.vectorcast.com/.

Usage

VectorCAST has the following options:

- **HTML Report (`html_report`)**: Specify the path to the HTML report which contains the test results.

- **Source code file extension (`file_extension`, default: .c)**: Source code file extension

- **Create test artefacts from HTML report (`generateTests`, default: false)**:

- **Sub Folder for test results (`sub_root`)**: Sub Folder for test results.

The full command line syntax for VectorCAST is:

```
-d
"type=VectorCAST,html_report=[file_or_directory],file_extension=[text],generateTests=[
booleanChoice],sub_root=[text]"
```

VectorCAST API

Description

The VectorCAST Data Provider extracts coverage results, as well as tests and their status

For more details, refer to https://www.vectorcast.com/.

Usage

VectorCAST API has the following options:

- **Sub Folder for test results (`sub_root`, default: Tests)**: Sub Folder for test results.

- **Generate the Squore Report? (`generate_report`, mandatory)**: Do you need to generate the Squore Report on the fly?

- **VectorCAST Project File or Directory (**.vce or **.vcp) (`project_file_list`)**: Specify the file or directory which contains all the vectorCAST projects to import.

- **Squore Report Folder (**.sqc) (`squore_report_folder`)*: Specify the folder which contains all the Squore report (*.sqc files)

- **Don't Be "case sensitve" (`case_sensitve_option`, default: true)**: Don't Be "case sensitve"

The full command line syntax for VectorCAST API is:

```
-d
"type=VectorCAST_API,sub_root=[text],generate_report=[multipleChoice],project_file_lis
t=[file_or_directory],squore_report_folder=[directory],case_sensitve_option=[booleanCh
oice]"
```

Vector Trace Items

Description

Import Trace Items in Vector generic format as Requirements in Squore

For more details, refer to https://www.vector.com/int/en/products/products-a-z/software/vTESTstudio/.

Usage

Vector Trace Items has the following options:

- **Trace Items folder (`dir`)**: Specify the folder containing Trace Items (Requirements) files

- **Trace Items file suffix (`suff`, default: .vti-tso)**: Provide the suffix of Trace Items (Requirements) files.

- **Default status (`default_status`, default: VERIFIED)**: It is possible to specify the defauls status for Requirements imported in Squore.

- **Planned Trace Items folder (`dirPlanned`)**: Specify the folder containing Planned Trace Items files.

- **Filter on Requirements (`filter`)**: The filter is a way to keep Requirements which properties match a certain pattern.

  Syntax:<PROPERTY_NAME>?regex=<REGEX>

  Examples:

  - ***No filters are provided***... If no filters are provided, all Requirements from vTESTstudio are shown in Squore (default behavior)
  - **Property 1?regex=V_.***... Only keep Requirements where 'Property 1' starts with 'V_'
  - **Property 1?regex=V_.*;Property 2?regex=.*VALID.***... Only keep Requirements where 'Property 1' starts with 'V_', **AND** 'Property 2' contains 'VALID'

- **Requirements grouping (`grouping`)**: Grouping is a way to structure Requirements in Squore by the value of given properties, in the order they are provided.

  Examples:Suppose Requirements have:

  - an 'Origin' property ('Internal', 'External')
  - and a 'Criticity' property ('A', 'B', 'C, 'D')

  Possible values for grouping:

  - **grouping is empty** ... If no grouping is provided, Requirement will be shown in Squore with the same structure as in vTESTstudio (default behavior)

- **grouping ='Origin'** ... In addition to the original structure, Requirements will be broken down by origin ('Internal, 'External', or 'Unknown' if the 'Origin' property is absent or empty)
- **grouping ='Origin;Criticity'** ... Same as before, but the Requirements will be broken down by Origin, **THEN** by Criticity ('A', 'B', 'C, 'D', or 'Unknown' if the 'Criticity' property is absent or empty)

The full command line syntax for Vector Trace Items is:

```
-d
"type=Vector_TraceItems,dir=[directory],suff=[text],default_status=[multipleChoice],di
rPlanned=[directory],filter=[text],grouping=[text]"
```

Bauhaus

Description

Import Findings from Bauhaus

For more details, refer to http://www.axivion.com.

Usage

Bauhaus has the following options:

- **CSV File (`csv`)**: Specify the CSV file which contains the findings results (MISRA, Coding Style...)

The full command line syntax for Bauhaus is:

```
-d "type=bauhaus,csv=[file]"
```

CodeSniffer

Description

CodeSniffer is a rulecker for PHP and Javascript

For more details, refer to http://www.squizlabs.com/php-codesniffer.

Usage

CodeSniffer has the following options:

- **CodeSniffer results file (checkstyle formatted XML) (`xml`)**: Point to the XML file that contains CodeSniffer results.

The full command line syntax for CodeSniffer is:

```
-d "type=codesniffer,xml=[file]"
```

Configuration Checker

Description

Use this tool to check for duplicated files or XML Elements between a custom configuration and the standard configuration.

Usage

Configuration Checker has the following options:

- **Standard Configuration Path (`s`)**:
- **Custom Configurations Path (`p`)**:

The full command line syntax for Configuration Checker is:

```
-d "type=conf-checker,s=[directory],p=[directory]"
```

CSV Coverage Import

Description

CSV Coverage Import provides a generic import mechanism for coverage results at function level

Usage

CSV Coverage Import has the following options:

- **CSV file (`csv`)**: Enter the path to the CSV containing the coverage data.

  The expected format of each line contained in the file is **PATH;NAME;TESTED_C1;OBJECT_C1;TESTED_MCC;OBJECT_MCC;TESTED_MCDC;OBJECT_MCDC;TCOV_MCC;TCOV_MCDC;TCOV_C1**

The full command line syntax for CSV Coverage Import is:

```
-d "type=csv_coverage,csv=[file]"
```

CSV Findings

Description

CSV Findings is a generic tool that allows importing findings into the project.

Usage

CSV Findings has the following options:

- **CSV File(s) (`csv`)**: Specify the path(s) to your CSV file(s) containing findings. To provide multiple files click on '+'. Each line in the file must use the following format and the file should include the following header:

  FILE;FUNCTION;RULE_ID;MESSAGE;LINE;COL;STATUS;STATUS_MESSAGE;TOOL

The full command line syntax for CSV Findings is:

```
-d "type=csv_findings,csv=[file]"
```

CSV Import

Description

Imports artefacts, metrics, findings, textual information and links from one or more CSV files. The expected CSV format for each of the input files is described in the user manuals in the csv_import framework reference.

Consult csv_import Reference for more details about the expected CSV format.

Usage

CSV Import has the following options:

- **CSV Separator (`separator`, default: ;)**: Specify the CSV Separator used in the CSV file.

- **CSV Delimiter (`delimiter`, default: ")**: CSV Delimiter is used when the separator is used inside a cell value. If a delimiter is used as a char in a cell it has to be doubled.

  The ' char is not allowed as a delimiter.

- **Artefact Path Separator (`pathSeparator`, default: /)**: Specify the character used as a separator in an artefact's path in the input CSV file.

- **Case-sensitive artefact lookup (`pathAreCaseSensitive`, default: true)**: When this option is turned on, artefacts in the CSV file are matched with existing source code artefacts in a case-sensitive manner.

- **Ignore source file path (`ignoreSourceFilePath`, default: false)**: When ignoring source file path it is your responsbility to ensure that file names are unique in the project.

- **Create missing files (`createMissingFile`, default: false)**: Automatically creates the artefacts declared in the CSV file if they do not exist.

- **Ignore finding if artefact not found (`ignoreIfArtefactNotFound`, default: true)**: If a finding can not be attached to any artefact then it is either ignored (checked) or it is attached to the project node instead (unchecked).

- **Unknown rule ID (`unknownRuleId`)**: For findings of a type that is not in your ruleset, set a default rule ID. The value for this parameter must be a valid rule ID from your analysis model.

- **Measure ID for orphan artifacts count (`orphanArteCountId`)**: To save the total count of orphan findings as a metric at application level, specify the ID of the measure to use in your analysis model.

- **Measure ID for unknown rules count (`orphanRulesCountId`)**: To save the total count of unknown rules as a metric at application level, Specify the ID of the measure to use in your analysis model.

- **Information ID receiving the list of unknown rules IDs (`orphanRulesListId`)**: To save the list of unknown rule IDs as textual information at application level, specify the ID of the textual information to use in your analysis model.

- **CSV File (`csv`)**: Specify the path to the input CSV file containing artefacts, metrics, findings, textual information, links and keys.

- **Metrics CSV File (`metrics`)**: Specify the path to the CSV file containing metrics.

- **Infos CSV File (`infos`)**: Specify the path to the CSV file containing textual information.

- **Findings CSV File (`findings`)**: Specify the path to the CSV file containing findings.

- **Keys CSV File (`keys`)**: Specify the path to the CSV file containing artefact keys.

- **Links CSV File (`links`)**: Specify the path to the CSV file containing links.

- **Reports artefacts mapping problem as (`level`, default: info)**: When an artefact referenced in the CSV file can not be found in the project, reports the problem as an information or as a warning.

The full command line syntax for CSV Import is:

```
-d
"type=csv_import,separator=[text],delimiter=[text],pathSeparator=[text],pathAreCaseSensitive=[booleanChoice],ignoreSourceFilePath=[booleanChoice],createMissingFile=[booleanChoice],ignoreIfArtefactNotFound=[booleanChoice],unknownRuleId=[text],orphanArteCountId=[text],orphanRulesCountId=[text],orphanRulesListId=[text],csv=[file],metrics=[file],infos=[file],findings=[file],keys=[file],links=[file],level=[multipleChoice]"
```

CSV Tag Import

Description

This data provider allows setting values for attributes in the project.

Usage

CSV Tag Import has the following options:

- **CSV file (`csv`)**: Specify the path to the file containing the metrics.

The full command line syntax for CSV Tag Import is:

```
-d "type=csv_tag_import,csv=[file]"
```

Generic Findings XML Import

Description

Generic Findings XML Import

Usage

Generic Findings XML Import has the following options:

- **XML File (`xml`)**: Specify the XML file which contains the findings results (MISRA, Coding Style...)
- **"Issue" mapping definition (`issue`)**:
- **"Rule Id" mapping definition (`id_rule`)**:
- **"Message" mapping definition (`message`)**:
- **"File" mapping definition (`file`)**:
- **"Line" mapping definition (`line`)**:
- **"Justification" mapping definition (`justification`)**:

The full command line syntax for Generic Findings XML Import is:

```
-d
"type=findings_xml,xml=[file],issue=[text],id_rule=[text],message=[text],file=[text],l
ine=[text],justification=[text]"
```

GNAThub

Description

Import data from GNAThub. GNAThub integrates and aggregates the results of AdaCore's various static and dynamic analysis tools (GNATmetric, GNATcheck, GNATcoverage, CodePeer). Compatible with GNAT Pro versions 7.4.2 up to 18.2.

For more details, refer to https://www.adacore.com/gnatpro/toolsuite/gnatdashboard.

This Data Provider will only be available after you configure your server or client *config.xml* with the path to your gnathub executable with a *<path name="gnatub" path="C:\tools\GNAThub\gnathub.exe" />* definition. Consult the Configuration Manual for more information about referencing external executables.

Usage

GNAThub has the following options:

- **Path of the gnathub.db file (`gnatdb`)**: Specify the absolute path of the gnathub.db file.

The full command line syntax for GNAThub is:

```
-d "type=gnathub,gnatdb=[file]"
```

CPU Data Import

Description

CPU Data Import provides a generic import mechanism for CPU data from a CSV or Excel file.

Usage

CPU Data Import has the following options:

- **Root node name (`root_node`, default: Resources)**: Specify the name of root node in the artefact tree.
- **Data File (`xls_file`)**: Specify the path to the file containing CPU information.
- **Sheet Name (`xls_sheetname`)**: Specify the name of the Excel sheet that contains the CPU list.
- **CPU Column name (`xls_key`)**: Specify the header name of the column which contains the CPU key.
- **Grouping Structure (`xls_groups`)**: Specify the headers for Grouping Structure, separated by ";".
- **Filtering (`xls_filters`)**: Specify the list of Header for filtering

  For example: **"column_name_1=regex1;column_name_2=regex2;**
- **Specify the CSV separator (`csv_separator`, default: ;)**: Specify the CSV separator
- **"CPU Loop Time" Column name (`cpu_loop_column_name`, default: Total Loop Time [ms])**: Specify the column name of the CPU Loop Time (Ex: "Total Loop Time [ms]")
- **"Average Idle Time per loop" Column name (`cpu_idle_column_name`, default: Average idle Time per loop [ms])**: Specify the column name of the Average Idle Time per loop (Ex: "Average idle Time per loop [ms]")
- **"Worst Case Idle Time per loop" Column name (`cpu_worst_column_name`, default: Worse case idle Time per loop [ms])**: Specify the column name of the Worst Case Idle Time per loop (Ex: "Worse case idle Time per loop [ms]")
- **Create an output file (`createOutput`, default: true)**: Create an output file

The full command line syntax for CPU Data Import is:

```
-d
"type=import_cpu,root_node=[text],xls_file=[file],xls_sheetname=[text],xls_key=[text],
xls_groups=[text],xls_filters=[text],csv_separator=[text],cpu_loop_column_name=[text],
cpu_idle_column_name=[text],cpu_worst_column_name=[text],createOutput=[booleanChoice]"
```

Excel Import

Description

Excel Import

Usage

Excel Import has the following options:

- **Input file (`input_file`)**: Specify the Excel input file
- **Sheetname (`sheetname`)**: Sheetname to read data from
- **Artefact Type (`artefact_type`)**: Artefact Type used by Squore Analysis model.

  Example: TEST
- **Artefact Type container (`artefact_type_container`)**: Artefact Type container used by Squore Analysis model.

  Example: TEST_FOLDER
- **Artefact unique ID (`artefact_uid`)**: **Optional unless you want to use links to these artefacts.**

  This is the artefact unique ID, to be used by links, from this Data Provider, or another Data Provider.Examples:

  - ${ID}
  - T_${Name}
  - ${Name} ${Descr}

  Note:${NAME} designates the column called NAME
- **Links to this artefact (`artefact_link`)**: Specify how to create links between this artefact and other artefacts with the following format:

  *<LINK_TYPE>?direction=<IN OR OUT>&column=<COLUMN_NAME>&separator=<SEPARATOR>*Examples:

  - **TESTED_BY?column=Test**

  A 'TESTED_BY' link will be created with the UID found in column 'Test'

  - **IMPLEMENTED_BY?direction=IN&column=Implements**

  An 'IMPLEMENTED_BY' link will be created with the UID found in column 'Implements'. Since the optional 'direction' attribute is provided, it will be set as 'IN' (default value is 'OUT')

  - **TESTED_BY?column=Tests&separator=','**

'TESTED_BY' links will be created with all UIDs found in column 'Tests', separated by a comma

  - **TESTED_BY?column=Tests&separator=',';REFINED_BY?column=DownLinks&separator=','**

'TESTED_BY' and 'REFINED_BY' links will be created with UIDs found in columns 'Tests' and 'DownLinks' respectively

- **Artefact name (`artefact_name`)**: Artefact name as displayed in Squore. Examples:

  - ${ID}
  - T_${Name}
  - ${Name} ${Descr}

  Note:${NAME} designates the column called NAME

- **Path to the artefact (`path_list`): Optional. If not used, artefacts extracted from the Excel file will be directly added to the Squore root.**

  To specify the path in Squore of artefacts exracted from the Excel file, using the following format:

  *<COLUMN_NAME>?map=[<REGEX_1>:<GROUP_NAME_1>,... ,<REGEX_N>:<GROUP_NAME_N>]&groupByDate=<YES>&format=<dd-mm-YYYY>*Examples:

  - **Area**

  Artefacts will be regrouped by the value found in the 'Area' column

  - **Area?map=[A*:Area A,B*:Area B]**

  Artefacts will be regrouped into two groups:'Area A', for all values of 'Area' column starting with letter 'A', and 'Area B' for letter 'B'.

  - **Started on?groupByDate=Yes&format=YYYY/mm/dd**

  Artefacts will be regrouped by the date found in column 'Started on', using the format 'YYYY/mm/dd'

  Note:Date patterns are based on SimpleDateFormat Java class specifications.

- **Textual data to extract (`info_list`)**: **Optional.**

  To specify the list of textual data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&map=[<REGEX_1>:<TEXT_1>,... ,<REGEX_N>:<TEXT_N>]*Examples:

  - **ZONE_ID?column=Zone**

  Textual data found in column 'Zone' will be associated to metric ZONE_ID

  - **ZONE_ID?column=Zone;OWNER?column=Belongs to**

  Textual data found in columns 'Zone' and 'Belongs to' will be associated to metric ZONE_ID and OWNER respectively

  - **ORIGIN?column=Comes from,map=[Cust*:External,Sub-contractor*:External,Support:Internal,Dev:Internal]**

  _Textual data found in column 'Comes from' will be associated to metric ORIGIN:

- With value 'External' if the column starts with 'Cust' or 'Sub-contractor'
- With value 'Internal' if the column equals 'Support' or 'Dev'

_

- **Started on?groupByDate=Yes&format=YYYY/mm/dd**

Artefacts will be regrouped by the date found in column 'Started on', using the format 'YYYY/mm/dd'

- **Numerical metrics to extract (`metric_list`): Optional.**

  To specify the list of numerical data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&extract=<REGEX_EXRACT>&map=[<REGEX_1>:<VALUE_1>,...,<REGEX_N>:<VALUE_N>]*Examples:

  - **PRIORITY?column=Priority level**

  Numerical values found in column 'Priority level' will be associated to metric PRIORITY

  - **SEVERITY?column=Severity level,extract=S_**

  Numerical values found in column 'Severity level' will be associated to metric SEVERITY, after having extracted (removed) the string 'S_', because in this example, column 'Severity level' contains for example 'S_1', 'S_4', etc., and we want to obtain '1', '4', etc.

  - **STATUS?column=State&map=[passed:0,Passed:0,Pass:0,*nconclusive*:1,failed:2,Failed:2,FAIL:2]**

  _Textual values found in column 'State' will be mapped to numerical values using these rules:

  - For values containing 'passed', 'Passed', 'Pass'
  - For values containing 'nconclusive'
  - For values containing 'failed', 'Failed, 'FAIL'

  _

- **Date metrics to extract (`date_list`): Optional.**

  To specify the list of date data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&format=<DATE_FORMAT>*Examples:

  - **CREATION_DATE?column=Created on**

  Date values found in column 'Created on' will be associated to metric CREATION_DATE, using the default dd-MMM-yyyy format

  - **LAST_UPDATE?column=Updated on&format=yyyy/mm/dd**

  Date values found in column 'Created on' will be associated to metric CREATION_DATE, using the yyyy/mm/dd format

  Note:Date patterns are based on SimpleDateFormat Java class specifications.

- **Filters to set the list of artefacts to keep (`filter_list`): Optional.**

  If specified only artefacts complying with the provided filters are kept. Use the following format:

*<COLUMN_NAME>?regex=<REGEX>*Examples:

- **Name?regex=^ST***

Only create artefacts for which column 'Name' starts with 'ST'

- **Name?regex=^ST*;Region?regex=Europe**

Same as before, but restrict to artefacts where column 'Region' is 'Europe'

- **Import data via UID only (`import_data_via_uid_only`, default: 0)**: Specify this option if you want to add metric/info to an artefact which created in another Data Provider
- **Header row index (`initial_row`, default: 0)**: Specify the line number where headers are defined.Note:Indexes start at value '0', e.g. the 4th line has index 3.

The full command line syntax for Excel Import is:

```
-d
"type=import_excel,input_file=[file],sheetname=[text],artefact_type=[text],artefact_ty
pe_container=[text],artefact_uid=[text],artefact_link=[text],artefact_name=[text],path
_list=[text],info_list=[text],metric_list=[text],date_list=[text],filter_list=[text],i
mport_data_via_uid_only=[text],initial_row=[text]"
```

Memory Data Import

Description

Memory Data Import provides a generic import mechanism for memory data from a CSV or Excel file.

Usage

Memory Data Import has the following options:

- **Root node name (`root_node`, default: Resources)**: Specify the name of root node in the artefact tree.
- **Data File (`xls_file`)**: Specify the path to the file containing Memory information.
- **Sheet Name (`xls_sheetname`)**: Specify the name of the Excel sheet that contains the Memory list.
- **Memory Column name (`xls_key`)**: Specify the header name of the column which contains the Memory key.
- **Grouping Structure (`xls_groups`)**: Specify the headers for Grouping Structure, separated by ";".
- **Filtering (`xls_filters`)**: Specify the list of Header for filtering

  For example: **"column_name_1=regex1;column_name_2=regex2;**
- **Specify the CSV separator (`csv_separator`, default: ;)**: Specify the CSV separator
- **Memory size column name (`memory_size_column_name`, default: Total)**: Specify the header name

of the column which contains the memory size.

- **Used memory column name (memory_used_column_name, default: Used)**: Specify the header name of the column which contains the used memory.

- **Memory type column name (memory_type_column_name, default: Type)**: Specify the header name of the column which contains the memory type.

- **ROM memory type name (memory_type_rom_name, default: ROM)**: Specify the name used for ROM memory.

- **RAM memory type name (memory_type_ram_name, default: RAM)**: Specify the name used for RAM memory.

- **NVM memory type name (memory_type_nvm_name, default: NVM)**: Specify the name used for NVM memory.

- **Create an output file (createOutput, default: true)**: Create an output file

The full command line syntax for Memory Data Import is:

```
-d
"type=import_memory,root_node=[text],xls_file=[file],xls_sheetname=[text],xls_key=[tex
t],xls_groups=[text],xls_filters=[text],csv_separator=[text],memory_size_column_name=[
text],memory_used_column_name=[text],memory_type_column_name=[text],memory_type_rom_na
me=[text],memory_type_ram_name=[text],memory_type_nvm_name=[text],createOutput=[boolea
nChoice]"
```

Requirement Data Import

Description

Requirement Data Import provides a generic import mechanism for requirements from a CSV.

Requirement Data Import provides fields so you can map all your requirements and spread them over the following statuses: Proposed, Analyzed, Approved, Implemented, Verified, Postponed, Deleted, Rejected. Overlapping statuses will cause an error, but if a requirement's status is not declared in the definition, the requirement will still be imported, and a finding will be created.

Usage

Requirement Data Import has the following options:

- **Root Node (root_node, default: Requirements)**: Specify the name of the node to attach requirements to.

- **Data File (input_file)**: Specify the path to the CSV file containing requirements.

- **Sheet Name (xls_sheetname)**: Specify the sheet name that contains the requirement list.

- **Requirement ID (artefact_id)**: Specify the header name of the column which contains the requirement ID.

- **Requirement version (`version`)**: Specify the header name of the column which contains the requirement version.

- **Linked Requirements IDs which satisfy this requirement (`link_satisfied_by`)**: Specify the header name of the column which contains the requirements IDs which satisfy this requirement.

- **Linked Test ID verifying this requirement (`link_tested_by`)**: Specify the header name of the column which contains the linked test ID verifying this requirement.

- **Linked Ticket ID associated to this requirement (`link_ticket`)**: Specify the header name of the column which contains the linked Ticket ID corresponding to an issue or enhancement request.

- **Requirement Name (`artefact_name`)**: Specify the pattern used to build the name of the requirement. The name can use any information collected from the CSV file as a parameter.

  Example: **${ID} : ${Summary}**

- **Requirement UID (`artefact_uid`)**: Specify the pattern used to build the requirement Unique ID. The UID can use any information collected from the CSV file as a parameter.

  Example: **TK#${ID}**

- **Grouping Structure (`artefact_groups`)**: Specify the headers for Grouping Structure, separated by ";".

  For example: **"column_name_1=regex1;column_name_2=regex2;**

- **Filtering (`artefact_filters`)**: Specify the list of Header for filtering

  For example: **"column_name_1=regex1;column_name_2=regex2;**

- **Applicable Requirement Pattern (`definition_applicable`)**: Specify the pattern applied to define requirements as Applicable. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Applicable=Yes**

- **Proposed Requirement Pattern (`definition_proposed`)**: Specify the pattern applied to define requirements as proposed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Proposed**

- **Analyzed Requirement Pattern (`definition_analyzed`)**: Specify the pattern applied to define requirements as analyzed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Examples:

  - **Status=Analyzed**
  - **Status=[Analyzed|Introduced]**
  - **Status=Analyzed;Decision=[final;revised]**

- **Approved Requirement Pattern (`definition_approved`)**: Specify the pattern applied to define requirements as approved. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Proposed**

- **Implemented Pattern (`definition_implemented`)**: Specify the pattern applied to define requirements as Implemented. This field accepts a regular expression to match one or more column headers with a list of possible values.

Example: **Status=Implemented**

- **Verified Requirement Pattern (`definition_verified`)**: Specify the pattern applied to define requirements as Verified. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Verified**

- **Postponed Requirement Pattern (`definition_postponed`)**: Specify the pattern applied to define requirements as Postponed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=postponed**

- **Deleted Requirement Pattern (`definition_deleted`)**: Specify the pattern applied to define requirements as deleted. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Deleted**

- **Rejected Requirement Pattern (`definition_rejected`)**: Specify the pattern applied to define requirements as rejected. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Rejected**

- **'Very high' Requirement priority Pattern (`definition_priority_very_high`)**: Specify the pattern applied to define requirements priority as 'Very High' (usually associated to value '1'). This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Priority=1**

- **'High' Requirement priority Pattern (`definition_priority_high`)**: Specify the pattern applied to define requirements priority as 'High' (usually associated to value '2'). This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Priority=2**

- **'Medium' Requirement priority Pattern (`definition_priority_medium`)**: Specify the pattern applied to define requirements priority as 'Medium' (usually associated to value '3'). This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Priority=3**

- **'Low' Requirement priority Pattern (`definition_priority_low`)**: Specify the pattern applied to define requirements priority as 'Low' (usually associated to value '4'). This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Priority=4**

- **'Met' Compliance Pattern (`definition_met`)**: Specify the pattern applied to define requirement Compliance as 'Met'. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Compliance=Met**

- **'Partially Met' Compliance Pattern (`definition_partially_met`)**: Specify the pattern applied to define requirement Compliance as 'Partially Met'. This field accepts a regular expression to match one or more column headers with a list of possible values.

Example: **Compliance=Partially Met**

- **'Not Met' Compliance Pattern (`definition_not_met`)**: Specify the pattern applied to define requirement Compliance as 'Not Met'. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Compliance=Not Met**

- **'Inspection' Test Method Pattern (`definition_inspection`)**: Specify the pattern applied to define requirement Test method as 'Inspection'. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **IADT Method=Inspection**

- **'Analysis' Test Method Pattern (`definition_analysis`)**: Specify the pattern applied to define requirement Test method as 'Analysis'. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **IADT Method=Analysis**

- **'Demonstration' Test Method Pattern (`definition_demonstration`)**: Specify the pattern applied to define requirement Test method as 'Demonstration'. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **IADT Method=Demonstration**

- **'Test' Test Method Pattern (`definition_test`)**: Specify the pattern applied to define requirement Test method as 'Test'. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **IADT Method=Test**

- **Creation Date Column (`creation_date`)**: Enter the name of the column containing the creation date of the requirement.

  Accepted formats are detailed here.

- **Last Update Column (`last_updated`)**: Enter the name of the column containing the last modification date of the requirement.

  Accepted formats are detailed here.

- **URL (`url`)**: Specify the pattern used to build the requirement URL. The URL can use any information collected from the CSV file as a parameter.

  Example: **https://example.com/bugs/${ID}**

- **Description Column (`description`)**: Specify the header of the column containing the description of the requirement.

- **Priority Column (`priority`)**: Specify the header of the column containing priority data.

- **'A' critical factor Pattern (`definition_crit_factor_A`)**: Specify the pattern applied to define requirement critical factor as 'A' (low). This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Criticity=A**.

- **'B' critical factor Pattern (`definition_crit_factor_B`)**: Specify the pattern applied to define requirement critical factor as 'B' (medium). This field accepts a regular expression to match one or more column headers with a list of possible values.

Example: **Criticity=B**.

- **'C' critical factor Pattern (`definition_crit_factor_C`)**: Specify the pattern applied to define requirement critical factor as 'C' (high). This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Criticity=C**.

- **'D' critical factor Pattern (`definition_crit_factor_D`)**: Specify the pattern applied to define requirement critical factor as 'D' (highest). This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Criticity=D**.

- **CSV Separator (`csv_separator`)**: Specify the character used in the CSV file to separate columns.

- **Information Fields (`informations`)**: Specify the list of extra textual information to import from the CSV file. This parameter expects a list of headers separated by ";" characters.

  For example: **Company;Country;Resolution**

- **Save Output (`createOutput`)**:

The full command line syntax for Requirement Data Import is:

```
-d
"type=import_req,root_node=[text],input_file=[file],xls_sheetname=[text],artefact_id=[
text],version=[text],link_satisfied_by=[text],link_tested_by=[text],link_ticket=[text]
,artefact_name=[text],artefact_uid=[text],artefact_groups=[text],artefact_filters=[tex
t],definition_applicable=[text],definition_proposed=[text],definition_analyzed=[text],
definition_approved=[text],definition_implemented=[text],definition_verified=[text],de
finition_postponed=[text],definition_deleted=[text],definition_rejected=[text],definit
ion_priority_very_high=[text],definition_priority_high=[text],definition_priority_medi
um=[text],definition_priority_low=[text],definition_met=[text],definition_partially_me
t=[text],definition_not_met=[text],definition_inspection=[text],definition_analysis=[t
ext],definition_demonstration=[text],definition_test=[text],creation_date=[text],last_
updated=[text],url=[text],description=[text],priority=[text],definition_crit_factor_A=
[text],definition_crit_factor_B=[text],definition_crit_factor_C=[text],definition_crit
_factor_D=[text],csv_separator=[text],informations=[text],createOutput=[booleanChoice]
"
```

Requirement ASIL via Excel Import

Description

Requirement ASIL via Excel Import

Usage

Requirement ASIL via Excel Import has the following options:

- **Input file (`input_file`)**: Specify the Excel input file
- **Sheetname (`sheetname`)**: Sheetname to read data from
- **Artefact name (`artefact_name`)**: Artefact name as displayed in Squore. Examples:
  - ${ID}
  - T_${Name}
  - ${Name} ${Descr}

  Note:${NAME} designates the column called NAME
- **Path to the artefact (`path_list`): Optional. If not used, artefacts extracted from the Excel file will be directly added to the Squore root.**

  To specify the path in Squore of artefacts exracted from the Excel file, using the following format:

  *<COLUMN_NAME>?map=[<REGEX_1>:<GROUP_NAME_1>,... ,<REGEX_N>:<GROUP_NAME_N>]&groupByDate=<YES>&format=<dd-mm-YYYY>*Examples:

  - **Area**

  Artefacts will be regrouped by the value found in the 'Area' column

  - **Area?map=[A*:Area A,B*:Area B]**

  Artefacts will be regrouped into two groups:'Area A', for all values of 'Area' column starting with letter 'A', and 'Area B' for letter 'B'.

  - **Started on?groupByDate=Yes&format=YYYY/mm/dd**

  Artefacts will be regrouped by the date found in column 'Started on', using the format 'YYYY/mm/dd'

  Note:Date patterns are based on SimpleDateFormat Java class specifications.
- **Textual data to extract (`info_list`): Optional.**

  To specify the list of textual data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&map=[<REGEX_1>:<TEXT_1>,... ,<REGEX_N>:<TEXT_N>]*Examples:

  - **ZONE_ID?column=Zone**

  Textual data found in column 'Zone' will be associated to metric ZONE_ID

  - **ZONE_ID?column=Zone;OWNER?column=Belongs to**

  Textual data found in columns 'Zone' and 'Belongs to' will be associated to metric ZONE_ID and OWNER respectively

  - **ORIGIN?column=Comes from,map=[Cust*:External,Sub-contractor*:External,Support:Internal,Dev:Internal]**

  _Textual data found in column 'Comes from' will be associated to metric ORIGIN:

    - With value 'External' if the column starts with 'Cust' or 'Sub-contractor'
    - With value 'Internal' if the column equals 'Support' or 'Dev'

  _

    - **Started on?groupByDate=Yes&format=YYYY/mm/dd**

  Artefacts will be regrouped by the date found in column 'Started on', using the format 'YYYY/mm/dd'

- **Numerical metrics to extract (`metric_list`): Optional.**

  To specify the list of numerical data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&extract=<REGEX_EXRACT>&map=[<REGEX_1>:<VALUE_1>,...,<REGEX_N>:<VALUE_N>]*Examples:

    - **PRIORITY?column=Priority level**

  Numerical values found in column 'Priority level' will be associated to metric PRIORITY

    - **SEVERITY?column=Severity level,extract=S_**

  Numerical values found in column 'Severity level' will be associated to metric SEVERITY, after having extracted (removed) the string 'S_', because in this example, column 'Severity level' contains for example 'S_1', 'S_4', etc., and we want to obtain '1', '4', etc.

    - **STATUS?column=State&map=[passed:0,Passed:0,Pass:0,*nconclusive*:1,failed:2,Failed:2,FAIL:2]**

  _Textual values found in column 'State' will be mapped to numerical values using these rules:

    - For values containing 'passed', 'Passed', 'Pass'
    - For values containing 'nconclusive'
    - For values containing 'failed', 'Failed, 'FAIL'

  _

- **Artefact unique ID (`artefact_uid`): Optional unless you want to use links to these artefacts.**

  This is the artefact unique ID, to be used by links, from this Data Provider, or another Data Provider.Examples:

    - ${ID}
    - T_${Name}
    - ${Name} ${Descr}

  Note:${NAME} designates the column called NAME

The full command line syntax for Requirement ASIL via Excel Import is:

```
-d
"type=import_req_asil,input_file=[file],sheetname=[text],artefact_name=[text],path_lis
t=[text],info_list=[text],metric_list=[text],artefact_uid=[text]"
```

Stack Data Import

Description

Stack Data Import provides a generic import mechanism for stack data from a CSV or Excel file.

Usage

Stack Data Import has the following options:

- **Root node name (`root_node`, default: Resources)**: Specify the name of root node in the artefact tree.
- **Data File (`xls_file`)**: Specify the path to the file containing Stack information.
- **Sheet Name (`xls_sheetname`)**: Specify the sheetname that contains the Stack list.
- **Stack Column name (`xls_key`)**: Specify the header name of the column which contains the Stack key.
- **Grouping Structure (`xls_groups`)**: Specify the headers for Grouping Structure, separated by ";".
- **Filtering (`xls_filters`)**: Specify the list of Header for filtering

  For example: **"column_name_1=regex1;column_name_2=regex2;**
- **Specify the CSV separator (`csv_separator`, default: ;)**: Specify the CSV separator
- **Stack size column (`stack_size_column_name`, default: Stack Size [Bytes])**: Specify the name of the column of Stack Size
- **Stack Average column (`stack_average_column_name`, default: Average Stack Size used [Bytes])**: Specify the name of the column of Stack Average
- **Stack Worst column (`stack_worst_column_name`, default: Worse Case Stack Size used [Bytes])**: Specify the name of the column of Stack Worst
- **Create an output file (`createOutput`, default: true)**: Create an output file

The full command line syntax for Stack Data Import is:

```
-d
"type=import_stack,root_node=[text],xls_file=[file],xls_sheetname=[text],xls_key=[text
],xls_groups=[text],xls_filters=[text],csv_separator=[text],stack_size_column_name=[te
xt],stack_average_column_name=[text],stack_worst_column_name=[text],createOutput=[bool
eanChoice]"
```

Test Data Import

Description

Test Data Import provides a generic import mechanism for tests from a CSV, Excel or JSON file.

Additionnally, it generates findings when the imported tests have an unknown status or type.

> This Data Provider provides fields so you can map all your tests and spread them over the following statuses: Failed, Inconclusive, Passd. Overlapping statuses and types will cause an error, but if a test status is not declared in the definition, the test will still be imported, and a finding will be created.

Usage

Test Data Import has the following options:

- **Root Node (`root_node`, default: Tests)**: Specify the name of the node to attach tests to.
- **Data File (`input_file`)**: Specify the path to the CSV, Excel or JSON file containing tests.
- **Excel Sheet Name (`xls_sheetname`)**: Specify the sheet name that contains the test list if your import file is in Excel format.
- **TestID (`artefact_id`)**: Specify the header name of the column which contains the test ID.
- **Linear Index Column (`linear_idx`)**: Specify the column name of the Linear Index (=Linear Index is used to order unit or integration tests in matrix graph).
- **Test Name (`artefact_name`)**: Specify the pattern used to build the name of the test. The name can use any information collected from the CSV file as a parameter.

  Example: **${ID} : ${Summary}**
- **Test UID (`artefact_uid`)**: Specify the pattern used to build the test Unique ID. The UID can use any information collected from the CSV file as a parameter.

  Example: **TST#${ID}**
- **Grouping Structure (`artefact_groups`)**: Specify the headers for Grouping Structure, separated by ";".

  For example: **"column_name_1=regex1;column_name_2=regex2;**
- **Filtering (`artefact_filters`)**: Specify the list of Header for filtering

  For example: **"column_name_1=regex1;column_name_2=regex2;**
- **Failed Test Pattern (`definition_failed`)**: Specify the pattern applied to define tests as failed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Failed**
- **Inconcusive Test Pattern (`definition_inconclusive`)**: Specify the pattern applied to define tests as inconclusive. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=[Inconclusive|Unfinished]**
- **Passed Test Pattern (`definition_passed`)**: Specify the pattern applied to define tests as passed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Passed**
- **Date when the test was executed (`execution_date`)**: Enter the name of the column containing

the execution date of the test.

Accepted formats are detailed here.

- **Unit of test duration (`execution_duration_unit`, default: ms)**: Enter the unit used for the test duration. Possible values are 's' (seconds) or 'ms' (milliseconds), default is 'ms')

- **Duration of the test (`execution_duration`)**: Enter duration of the test, in milliseconds.

- **TODO Pattern (`in_todo_list`)**: Specify the pattern applied to include tests in the TODO list. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Active=Yes**

- **Creation Date Column (`creation_date`)**: Enter the name of the column containing the creation date of the test.

  Accepted formats are detailed here.

- **Last Updated Date Column (`last_updated_date`)**: Enter the name of the column containing the last updated date of the test.

  Accepted formats are detailed here.

- **URL (`url`)**: Specify the pattern used to build the test URL. The URL can use any information collected from the CSV file as a parameter.

  Example: **https://example.com/tests/${ID}**

- **Description Column (`description`)**: Specify the header of the column containing the description of the test.

- **Category Column (`category`)**: Specify the header of the column containing the category of the test.

- **Priority Column (`priority`)**: Specify the header of the column containing priority data.

- **CSV Separator (`csv_separator`)**: Specify the character used in the CSV file to separate columns.

- **Information Fields (`informations`)**: Specify the list of extra textual information to import from the CSV file. This parameter expects a list of headers separated by ";" characters.

  For example: **Architecture;Responsible;Target**

- **Save Output (`createOutput`)**:

The full command line syntax for Test Data Import is:

```
-d
"type=import_test,root_node=[text],input_file=[file],xls_sheetname=[text],artefact_id=
[text],linear_idx=[text],artefact_name=[text],artefact_uid=[text],artefact_groups=[tex
t],artefact_filters=[text],definition_failed=[text],definition_inconclusive=[text],def
inition_passed=[text],execution_date=[text],execution_duration_unit=[multipleChoice],e
xecution_duration=[text],in_todo_list=[text],creation_date=[text],last_updated_date=[t
ext],url=[text],description=[text],category=[text],priority=[text],csv_separator=[text
],informations=[text],createOutput=[booleanChoice]"
```

Test Excel Import

Description

Test Excel Import

Usage

Test Excel Import has the following options:

- **Input file (`input_file`)**: Specify the Excel input file
- **Sheetname (`sheetname`)**: Sheetname to read data from
- **Artefact name (`artefact_name`)**: Artefact name as displayed in Squore. Examples:

  - ${ID}
  - T_${Name}
  - ${Name} ${Descr}

  Note:${NAME} designates the column called NAME

- **Path to the artefact (`path_list`): Optional. If not used, artefacts extracted from the Excel file will be directly added to the Squore root.**

  To specify the path in Squore of artefacts exracted from the Excel file, using the following format:

  *<COLUMN_NAME>?map=[<REGEX_1>:<GROUP_NAME_1>,... ,<REGEX_N>:<GROUP_NAME_N>]&groupByDate=<YES>&format=<dd-mm-YYYY>*Examples:

  - **Area**

  Artefacts will be regrouped by the value found in the 'Area' column

  - **Area?map=[A*:Area A,B*:Area B]**

  Artefacts will be regrouped into two groups:'Area A', for all values of 'Area' column starting with letter 'A', and 'Area B' for letter 'B'.

  - **Started on?groupByDate=Yes&format=YYYY/mm/dd**

  Artefacts will be regrouped by the date found in column 'Started on', using the format 'YYYY/mm/dd'

  Note:Date patterns are based on SimpleDateFormat Java class specifications.

- **Textual data to extract (`info_list`): Optional.**

  To specify the list of textual data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&map=[<REGEX_1>:<TEXT_1>,... ,<REGEX_N>:<TEXT_N>]*Examples:

  - **ZONE_ID?column=Zone**

  Textual data found in column 'Zone' will be associated to metric ZONE_ID

  - **ZONE_ID?column=Zone;OWNER?column=Belongs to**

  Textual data found in columns 'Zone' and 'Belongs to' will be associated to metric ZONE_ID and OWNER respectively

  - **ORIGIN?column=Comes from,map=[Cust*:External,Sub-contractor*:External,Support:Internal,Dev:Internal]**

  _Textual data found in column 'Comes from' will be associated to metric ORIGIN:

  - With value 'External' if the column starts with 'Cust' or 'Sub-contractor'
  - With value 'Internal' if the column equals 'Support' or 'Dev'

  _

  - **Started on?groupByDate=Yes&format=YYYY/mm/dd**

  Artefacts will be regrouped by the date found in column 'Started on', using the format 'YYYY/mm/dd'

- **Numerical metrics to extract (`metric_list`): Optional.**

  To specify the list of numerical data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&extract=<REGEX_EXRACT>&map=[<REGEX_1>:<VALUE_1>,...,<REGEX_N>:<VALUE_N>]*Examples:

  - **PRIORITY?column=Priority level**

  Numerical values found in column 'Priority level' will be associated to metric PRIORITY

  - **SEVERITY?column=Severity level,extract=S_**

  Numerical values found in column 'Severity level' will be associated to metric SEVERITY, after having extracted (removed) the string 'S_', because in this example, column 'Severity level' contains for example 'S_1', 'S_4', etc., and we want to obtain '1', '4', etc.

  - **STATUS?column=State&map=[passed:0,Passed:0,Pass:0,*nconclusive*:1,failed:2,Failed:2,FAIL:2]**

  _Textual values found in column 'State' will be mapped to numerical values using these rules:

  - For values containing 'passed', 'Passed', 'Pass'
  - For values containing 'nconclusive'
  - For values containing 'failed', 'Failed, 'FAIL'

  _

- **Date metrics to extract (`date_list`): Optional.**

  To specify the list of date data to extract from the Excel file, using the following format:

  *<METRIC_ID>?column=<COLUMN_NAME>&format=<DATE_FORMAT>*Examples:

  - **CREATION_DATE?column=Created on**

  Date values found in column 'Created on' will be associated to metric CREATION_DATE, using the default dd-MMM-yyyy format

  - **LAST_UPDATE?column=Updated on&format=yyyy/mm/dd**

Date values found in column 'Created on' will be associated to metric CREATION_DATE, using the yyyy/mm/dd format

Note:Date patterns are based on SimpleDateFormat Java class specifications.

- **Filters to set the list of artefacts to keep (`filter_list`): Optional.**

  If specified only artefacts complying with the provided filters are kept. Use the following format:

  *<COLUMN_NAME>?regex=<REGEX>*Examples:

  - **Name?regex=^ST***

  Only create artefacts for which column 'Name' starts with 'ST'

  - **Name?regex=^ST*;Region?regex=Europe**

  Same as before, but restrict to artefacts where column 'Region' is 'Europe'

- **Artefact unique ID (`artefact_uid`): Optional unless you want to use links to these artefacts.**

  This is the artefact unique ID, to be used by links, from this Data Provider, or another Data Provider.Examples:

  - ${ID}
  - T_${Name}
  - ${Name} ${Descr}

  Note:${NAME} designates the column called NAME

- **Links to this artefact (`artefact_link`)**: Specify how to create links between this artefact and other artefacts with the following format:

  *<LINK_TYPE>?direction=<IN OR OUT>&column=<COLUMN_NAME>&separator=<SEPARATOR>*Examples:

  - **TESTED_BY?column=Test**

  A 'TESTED_BY' link will be created with the UID found in column 'Test'

  - **IMPLEMENTED_BY?direction=IN&column=Implements**

  An 'IMPLEMENTED_BY' link will be created with the UID found in column 'Implements'. Since the optional 'direction' attribute is provided, it will be set as 'IN' (default value is 'OUT')

  - **TESTED_BY?column=Tests&separator=','**

  'TESTED_BY' links will be created with all UIDs found in column 'Tests', separated by a comma

  - **TESTED_BY?column=Tests&separator=',';REFINED_BY?column=DownLinks&separator=','**

  'TESTED_BY' and 'REFINED_BY' links will be created with UIDs found in columns 'Tests' and 'DownLinks' respectively

The full command line syntax for Test Excel Import is:

```
-d
"type=import_test_excel,input_file=[file],sheetname=[text],artefact_name=[text],path_l
ist=[text],info_list=[text],metric_list=[text],date_list=[text],filter_list=[text],art
efact_uid=[text],artefact_link=[text]"
```

Ticket Data Import

Description

Ticket Data Import provides a generic import mechanism for tickets from a CSV, Excel or JSON file. Additionnally, it generates findings when the imported tickets have an unknown status or type.

> This Data Provider provides fields so you can map all your tickets as Enhancements and defects and spread them over the following statuses: Open, In Implementation, In Verification, Closed. Overlapping statuses and types will cause an error, but if a ticket's type or status is not declared in the definition, the ticket will still be imported, and a finding will be created.

Usage

Ticket Data Import has the following options:

- **Root Node (`root_node`, default: Tickets)**: Specify the name of the node to attach tickets to.

- **Data File (`input_file`)**: Specify the path to the CSV, Excel or JSON file containing tickets.

- **Excel Sheet Name (`xls_sheetname`)**: Specify the sheet name that contains the ticket list if your import file is in Excel format.

- **Ticket ID (`artefact_id`)**: Specify the header name of the column which contains the ticket ID.

- **Ticket Name (`artefact_name`)**: Specify the pattern used to build the name of the ticket. The name can use any information collected from the CSV file as a parameter.

  Example: **${ID} : ${Summary}**

- **Ticket UID (`artefact_uid`)**: Specify the pattern used to build the ticket Unique ID. The UID can use any information collected from the CSV file as a parameter.

  Example: **TK#${ID}**

- **Grouping Structure (`artefact_groups`)**: Specify the headers for Grouping Structure, separated by ";".

  For example: **"column_name_1=regex1;column_name_2=regex2;**

- **Filtering (`artefact_filters`)**: Specify the list of Header for filtering

  For example: **"column_name_1=regex1;column_name_2=regex2;**

- **Open Ticket Pattern (`definition_open`)**: Specify the pattern applied to define tickets as open. This field accepts a regular expression to match one or more column headers with a list of possible values.

Example: **Status=[Open|New]**

- **In Development Ticket Pattern (`definition_rd_progress`)**: Specify the pattern applied to define tickets as in development. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Implementing**

- **Fixed Ticket Pattern (`definition_vv_progress`)**: Specify the pattern applied to define tickets as fixed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Verifying;Resolution=[fixed;removed]**

- **Closed Ticket Pattern (`definition_close`)**: Specify the pattern applied to define tickets as closed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Closed**

- **Defect Pattern (`definition_defect`)**: Specify the pattern applied to define tickets as defects. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Type=Bug**

- **Enhancement Pattern (`definition_enhancement`)**: Specify the pattern applied to define tickets as enhancements. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Type=Enhancement**

- **TODO Pattern (`in_todo_list`)**: Specify the pattern applied to include tickets in the TODO list. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Sprint=2018-23**

- **Creation Date Column (`creation_date`)**: Enter the name of the column containing the creation date of the ticket.

  Accepted formats are detailed here.

- **Due Date Column (`due_date`)**: Enter the name of the column containing the due date of the ticket.

  Accepted formats are detailed here.

- **Last Updated Date Column (`last_updated_date`)**: Enter the name of the column containing the last updated date of the ticket.

  Accepted formats are detailed here.

- **Closure Date Column (`closure_date`)**: Enter the name of the column containing the closure date of the ticket.

  Accepted formats are detailed here.

- **URL (`url`)**: Specify the pattern used to build the ticket URL. The URL can use any information collected from the CSV file as a parameter.

  Example: **https://example.com/bugs/${ID}**

- **Description Column (`description`)**: Specify the header of the column containing the description of the ticket.

- **Category Column (`category`)**: Specify the header of the column containing the category of the ticket.

- **Reporter Column (`reporter`)**: Specify the header of the column containing the reporter of the ticket.

- **Handler Column (`handler`)**: Specify the header of the column containing the handler of the ticket.

- **Priority Column (`priority`)**: Specify the header of the column containing priority data.

- **Severity Column (`severity`)**: Specify the header of the column containing severity data.

- **CSV Separator (`csv_separator`)**: Specify the character used in the CSV file to separate columns.

- **Information Fields (`informations`)**: Specify the list of extra textual information to import from the CSV file. This parameter expects a list of headers separated by ";" characters.

  For example: **Company;Country;Resolution**

- **Save Output (`createOutput`)**:

The full command line syntax for Ticket Data Import is:

```
-d
"type=import_ticket,root_node=[text],input_file=[file],xls_sheetname=[text],artefact_i
d=[text],artefact_name=[text],artefact_uid=[text],artefact_groups=[text],artefact_filt
ers=[text],definition_open=[text],definition_rd_progress=[text],definition_vv_progress
=[text],definition_close=[text],definition_defect=[text],definition_enhancement=[text]
,in_todo_list=[text],creation_date=[text],due_date=[text],last_updated_date=[text],clo
sure_date=[text],url=[text],description=[text],category=[text],reporter=[text],handler
=[text],priority=[text],severity=[text],csv_separator=[text],informations=[text],creat
eOutput=[booleanChoice]"
```

Jira

Description

This Data Provider extracts tickets and their attributes from a Jira instance to create ticket artefacts in your project.

For more details, refer to https://www.atlassian.com/software/jira.

> The extracted JSON from Jira is then passed to the Ticket Data Import Data Provider (described in Ticket Data Import). Finer configuration of the data passed from this Data Provider to Ticket Data Import is available by editing (or overriding) *<SQUORE_HOME>/addons/tools/jira/jira_config.xml*.

Usage

Jira has the following options:

- **Jira REST API URL (`url`, mandatory)**: The URL used to connect to yout Jira instance's REST API URL (e.g: https://jira.domain.com/rest/api/2)

- **Jira User login (`login`, mandatory)**: Specyfy your Jira User login.

- **Jira User password (`pwd`, mandatory)**: Specify your Jira User password.

- **Number of queried tickets (`max_results`, mandatory, default: -1)**: Maximum number of queried tickets returned by the query (default is -1, meaning 'retrieve all tickets').

- **Additional Fields (`additional_fields`, default: environment,votes)**: List additional fields to be exported from Jira.

  This field accepts a comma-separated list of field names that are added to the export request URL, for example **fixVersions,versions**

- **Grouping Structure (`artefact_groups`, default: fields/components[0]/name)**: Specify the headers for Grouping Structure, separated by ";".

  For example: **"column_name_1=regex1;column_name_2=regex2;**

- **Creation Date Field (`creation_date`, default: fields/created)**: Enter the name of the column containing the creation date of the ticket.

  For example: **column_name{format="dd/mm/yyyy"})**.

  If format is not specified, the following is used by default: **dd/mm/yyyy**.

- **Closure Date Field (`closure_date`, default: fields/resolutiondate)**: Enter the name of the column containing the closure date of the ticket.

  For example: **column_name{format="dd/mm/yyyy"})**.

  If format is not specified, the following is used by default: **dd/mm/yyyy**.

- **Due Date Field (`due_date`, default: fields/duedate)**: Enter the name of the column containing the due date of the ticket.

  For example: **column_name{format="dd/mm/yyyy"})**.

  If format is not specified, the following is used by default: **dd/mm/yyyy**.

- **Last Updated Date Field (`last_updated_date`, default: fields/updated)**: Enter the name of the column containing the last updated date of the ticket.

  For example: **column_name{format="dd/mm/yyyy"})**.

  If format is not specified, the following is used by default: **dd/mm/yyyy**.

- **Category (`category`, default: fields/components[0]/name)**: Specify the path to the field that will contain the ticket category.

- **Priority (`priority`, default: fields/priority/name)**: Specify the path to the field that will contain the ticket priority.

- **JQL Request (`jql_request`)**: Specify a JQL request (see JIRA documentation) in order to limit the number of elements sent by the JIRA server.

For example: **project=MyProject**.This parameter is optional.

- **Filtering (`artefact_filters`, default: fields/issuetype/name=(Task|Bug|Improvement|New Feature))**: Specify the list of Header for filtering

  For example: **"column_name_1=regex1;column_name_2=regex2;**

- **Information Fields (`informations`, default: fields/environment;fields/votes/votes)**: Specify a semicolon-separated list of paths to fields you want to extract from the Jira JSON export to be added as textual information for the ticket artefacts.

  For example: **fields/fixVersions[0]/name;fields/versions[0]/name**

- **Open Ticket Pattern (`definition_open`, default: fields/status/name=[To Do|Open|Reopened])**: Specify the pattern applied to define tickets as open. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=[Open|New]**

- **In Development Ticket Pattern (`definition_rd_progress`, default: fields/status/name=[In Progress|In Review])**: Specify the pattern applied to define tickets as in development. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Implementing**

- **Fixed Ticket Pattern (`definition_vv_progress`, default: fields/status/name=[Verified])**: Specify the pattern applied to define tickets as fixed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Verifying;Resolution=[fixed;removed]**

- **Closed Ticket Pattern (`definition_close`, default: fields/status/name=[Resolved|Closed|Done])**: Specify the pattern applied to define tickets as closed. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Status=Closed**

- **Defect Pattern (`definition_defect`, default: fields/issuetype/name=[Bug])**: Specify the pattern applied to define tickets as defects. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Type=Bug**

- **Enhancement Pattern (`definition_enhancement`, default: fields/issuetype/name=[Improvement|New Feature])**: Specify the pattern applied to define tickets as enhancements. This field accepts a regular expression to match one or more column headers with a list of possible values.

  Example: **Type=Enhancement**

- **Todo list regex (`in_todo_list`, default: fields/status/name=.)***: Todo list regex (ticket which fit the regex will be considered as part of the TODO list for the analysis)

The full command line syntax for Jira is:

```
-d
"type=jira,url=[text],login=[text],pwd=[password],max_results=[text],additional_fields
=[text],artefact_groups=[text],creation_date=[text],closure_date=[text],due_date=[text
],last_updated_date=[text],category=[text],priority=[text],jql_request=[text],artefact
_filters=[text],informations=[text],definition_open=[text],definition_rd_progress=[tex
t],definition_vv_progress=[text],definition_close=[text],definition_defect=[text],defi
nition_enhancement=[text],in_todo_list=[text]"
```

Mantis

Description

The Mantis Data Provider extracts tickets and their attributes from a Mantis installation and creates ticket artefacts.

Prerequisites:

- This Data Provider queries Mantis tickets using the Mantis BT REST API. An **API token** is required to access this API.
- The Mantis server should be configured to avoid filtering 'Authorization' headers.

See http://docs.php.net/manual/en/features.http-auth.php#114877 for further details.

For more details, refer to https://www.mantisbt.com.

> The extracted JSON from Mantis BT is then passed to the Ticket Data Import Data Provider (described in Ticket Data Import). Finer configuration of the data passed from this Data Provider to Ticket Data Import is available by editing (or overriding) *<SQUORE_HOME>/addons/tools/mantis/mantis_config.xml*.

Usage

Mantis has the following options:

- **Mantis URL (`url`, mandatory)**: Specify the URL of the Mantis instance (e.g: https://www.mantisbt.org/bugs/api/rest)
- **Mantis API Token (`api_token`, mandatory)**: Copy the Mantis API Token generated from your Account Settings in Mantis.
- **Number of queried tickets (`max_results`, mandatory, default: 50)**: Maximum number of queried tickets returned by the query (default is 50. value=-1 means 'retrieve all tickets').

The full command line syntax for Mantis is:

```
-d "type=mantis,url=[text],api_token=[text],max_results=[text]"
```

OSLC

Description

OSLC-CM allows retrieving information from Change Management systems following the OSLC standard. Metrics and artefacts are created by connecting to the OSLC system and retrieving issues with the specified query.

For more details, refer to http://open-services.net/.

Usage

OSLC has the following options:

- **Change Server (`server`)**: Specify the URL of the project you want to query on the OSLC server. Typically the URL will look like this: http://myserver:8600/change/oslc/db/3454a67f-656ddd4348e5/role/User/

- **Query (`query`)**: Specify the query to send to the OSLC server (e.g.: release="9TDE/TDE_00_01_00_00"). It is passed to the request URL via the ?oslc_cm.query= parameter.

- **Query Properties (`properties`, default: request_type,problem_number,crstatus,severity,submission_area,functionality…)**: Specify the properties to add to the query. They are passed to the OSLC query URL using the ?oslc_cm.properties= parameter.

- **Login (`login`)**:

- **Password (`password`)**:

The full command line syntax for OSLC is:

```
-d
"type=oslc_cm,server=[text],query=[text],properties=[text],login=[text],password=[pass
word]"
```

pep8

Description

pep8 is a tool to check your Python code against some of the style conventions in PEP 88. Its CSV report file is imported to generate findings.

For more details, refer to https://pypi.python.org/pypi/pep8.

Usage

pep8 has the following options:

- **CSV results file (`csv`)**: Specify the path to the CSV report file created by pep8.

The full command line syntax for pep8 is:

```
-d "type=pep8,csv=[file]"
```

pycodestyle / pep8 (plugin)

Description

Style Guide for Python Code. Pep8 results are imported to produce findings on Python code. This data provider requires having pycodestyle or pep8 installed on the machine running the analysis and the pycodestyle or pep8 command to be available in the path. It is compatible with pycodestyle 2.4 or pep8 1.7 and may also work with older versions.

For more details, refer to https://pypi.org/project/pycodestyle.

Usage

pycodestyle / pep8 (plugin) has the following options:

- **Source code directory to analyse (`dir`)**: Leave this field empty to analyse all sources.

The full command line syntax for pycodestyle / pep8 (plugin) is:

```
-d "type=pep8_auto,dir=[directory]"
```

PHP Code Coverage

Description

Library that provides collection, processing, and rendering functionality for PHP code coverage information.

For more details, refer to https://github.com/sebastianbergmann/php-code-coverage.

Usage

PHP Code Coverage has the following options:

- **Report file or folder (`html_report`)**: Specify the path to the HTML report folder or file which contains the coverage results.

The full command line syntax for PHP Code Coverage is:

```
-d "type=phpcodecoverage,html_report=[file_or_directory]"
```

pylint

Description

Pylint is a Python source code analyzer which looks for programming errors, helps enforcing a coding standard and sniffs for some code smells (as defined in Martin Fowler's Refactoring book). Pylint results are imported to generate findings for Python code.

For more details, refer to http://www.pylint.org/.

Usage

pylint has the following options:

- **CSV results file (`csv`)**: Specify the path to the CSV file containing pylint results. Note that the minimum version supported is 1.1.0.

The full command line syntax for pylint is:

```
-d "type=pylint,csv=[file]"
```

pylint (plugin)

Description

Coding Guide for Python Code. Pylint results are imported to produce findings on Python code. This data provider requires having pylint installed on the machine running the analysis and the pylint command to be available in the path. It is known to work with pylint 1.7.0 and may also work with older versions.

Usage

pylint (plugin) has the following options:

- **Source code directory to analyse (`dir`)**: Leave this field empty to analyse all sources.

The full command line syntax for pylint (plugin) is:

```
-d "type=pylint_auto,dir=[directory]"
```

QAC 8.2

Description

QA-C is a static analysis tool for MISRA checking.

For more details, refer to http://www.programmingresearch.com/static-analysis-software/qac-qacpp-static-analyzers/.

Usage

QAC 8.2 has the following options:

- **QAC output file(s) (`txt`, mandatory)**: Specify the path(s) to the .tab file(s) to extract findings from. To provide multiple files click on '+'

- **Eliminate duplicated findings (`eliminate_duplicate`, default: false)**: When 2 occurences of the same finding (same rule, same file, same line, same description) is found, only one is reported.

The full command line syntax for QAC 8.2 is:

```
-d "type=qac,txt=[file],eliminate_duplicate=[booleanChoice]"
```

QAC 8.2 CERT Import

Description

QA-C is a static analysis tool for MISRA and CERT checking.

For more details, refer to http://www.programmingresearch.com/static-analysis-software/qac-qacpp-static-analyzers/.

Usage

QAC 8.2 CERT Import has the following options:

- **QAC CERT output file(s) (`txt`, mandatory)**: Specify the path(s) to the .tab file(s) to extract findings from. To provide multiple files click on '+'

- **Eliminate duplicated findings (`eliminate_duplicate`, default: false)**: When 2 occurences of the same finding (same rule, same file, same line, same description) is found, only one is reported.

The full command line syntax for QAC 8.2 CERT Import is:

```
-d "type=qac_cert,txt=[file],eliminate_duplicate=[booleanChoice]"
```

SonarQube

Description

This data provider imports findings from SonarQube. Note that versions prior to 6.2 may not be supported.

For more details, refer to https://www.sonarqube.org/.

Usage

SonarQube has the following options:

- **SonarQube Location (`sonar`, default: http://127.0.0.1:9000)**: Specify the URL of the SonarQube installation to work with (for example: http://localhost:9000)

- **SonarQube Component Key (`key`)**:

- **Version Name (`version`)**:

- **Login (`login`)**:

- **Password (`password`)**:

The full command line syntax for SonarQube is:

```
-d
"type=sonarqube,sonar=[text],key=[text],version=[text],login=[text],password=[password
]"
```

Testwell CTC++

Description

Import data from Testwell CTC++ XML results

For more details, refer to http://www.testwell.fi/ctcdesc.html.

Usage

Testwell CTC++ has the following options:

- **Results folder (`dir`)**: Specify the folder containing XML test results files from Testwell CTC++.

The full command line syntax for Testwell CTC++ is:

```
-d "type=testwell_ctc,dir=[directory]"
```

vTESTstudio Traceability

Description

Import vTESTstudio traeability Matrix information

For more details, refer to https://www.vector.com/int/en/products/products-a-z/software/vTESTstudio/.

Usage

vTESTstudio Traceability has the following options:

- **Traceability matrix file path (`file`)**: Specify the absolute path to the vTESTstudio traceability matrix file (.vti-tso format)
- **Test path (`testPath`, default: Tests)**: Define test path (for example Test/HIL Test), by default the value is Tests.

The full command line syntax for vTESTstudio Traceability is:

```
-d "type=vTestStudio_Traceability,file=[file],testPath=[text]"
```

PC Lint MISRA 2012

Description

PC Lint MISRA 2012 (via XML import)

Usage

PC Lint MISRA 2012 has the following options:

- **XML File (`xml`)**: Specify the XML file which contains the findings results (MISRA, Coding Style...)

The full command line syntax for PC Lint MISRA 2012 is:

```
-d "type=vectorCAST_Lint,xml=[file]"
```

Adding More Languages to Squan Sources

Squan Sources can handle files written in languages that are not officially supported with a bit of extra configuration. In this mode, only a basic analysis of the file is carried out so that an artefact is created in the project and findings can be attached to it. A subset of the base metrics from Squan Sources is optionally recorded for the artefact so that line counting, stability and text duplication metrics are available at file level for the new language.

The example below shows how you can add TypeScript files to your analysis:

1. Copy *<SQUORE_HOME>/configuration/tools/SQuORE/form.xml* and its *.properties* files into your own configuration
2. Edit *form.xml* to add a new language key and associated file extensions:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <tags baseName="SQuORE" ...>
       <tag type="multipleChoice" key="languages" ... defaultValue="...;typescript">
           ...
           <value key="typescript" option=".ts,.TS" />
       </tag>
   </tags>
   ```

   Files with extensions matching the **typescript** language will be added to your project as TYPESCRIPT_FILE artefacts

3. Edit the `defaultValue` of the `additional_param` field to specify how Squan Sources should count source code lines and comment lines in the new language, based on another language officially supported by Squore. This step is optional, and is only needed if you want the to record basic line counting metrics for the artefacts.

```
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="SQuORE" ...>
    ...
    <tag type="text" key="additional_param" defaultValue="typescript=javascript" />
    ...
</tags>
```

Lines in TypeScript files will be counted as they would for Javascript code.

4. Add translations for the new language key to show in the web UI in Squan Sources's *form_en.properties*

```
OPT.typescript.NAME=TypeScript
```

5. Add translations for the new artefact type and new LANGUAGE information value in one of the properties files imported by your Description Bundle:

```
T.TYPESCRIPT_FILE.NAME=TypeScript File

INFO_VALUE.LANGUAGE.TYPESCRIPT.NAME=Typescript
INFO_VALUE.LANGUAGE.TYPESCRIPT.COLOR=#2b7489
```

6. The new artefact type should also be declared as a type in your model. The easiest way to do this is to add it to the **GENERIC_FILE** alias in your analysis model, which is pre-configured to record the line counting metrics for new artefacts. You should also define a root indicator for you new artefact type. The following snippet shows a minimal configuration using a dummy indicator:

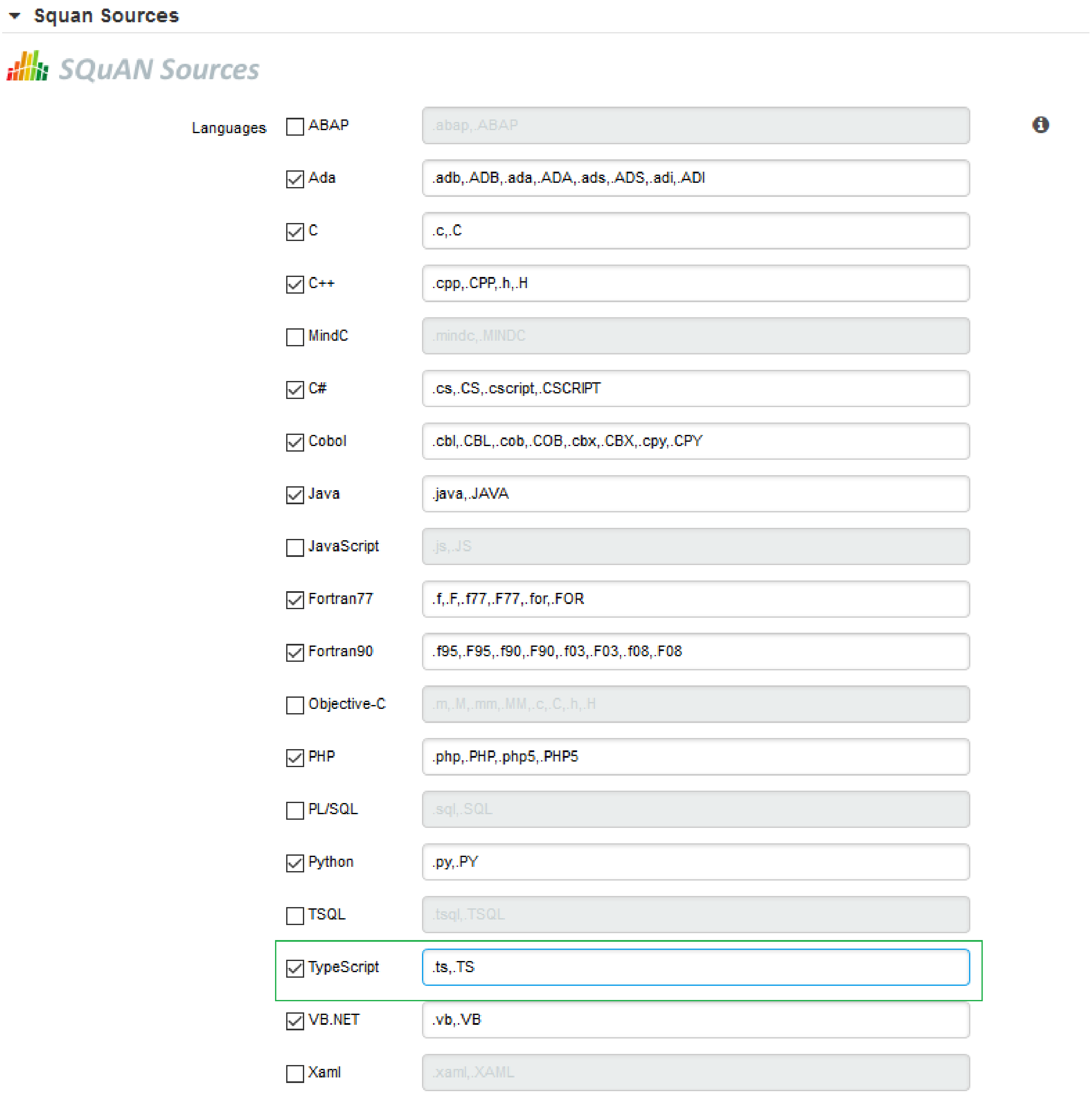
```
<!-- <configuration>/MyModel/Analysis/Bundle.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<Bundle>
...
    <ArtefactType id="GENERIC_FILE" heirs="TYPESCRIPT_FILE" />

    <RootIndicator artefactTypes="TYPESCRIPT_FILE" indicatorId="DUMMY" />
    <Indicator indicatorId="DUMMY" scaleId="SCALE_INFO" targetArtefactTypes=
"TYPESCRIPT_FILE" displayTypes="IMAGE" />

    <Measure measureId="DUMMY">
        <Computation targetArtefactTypes="TYPESCRIPT_FILE" result="0" />
    </Measure>
...
</Bundle>
```

Make sure that this declaration appears in your analysis model before the inclusion of *import.xml* so it overrides the default analysis model.

Don't forget to add translations for your dummy indicator to avoid warnings in the Model Validator:

```
DUMMY.NAME= Generic Indicator
DUMMY.DESCR= This is an indicator for additional languages in Squan Sources. It
does not rate files in any way.
```

7. Reload your configuration and analyse a project, checking the box for TypeScript in Squan Sources's options to get Typescrypt artefacts in your project.



▾ Squan Sources

SQuAN Sources

Languages	☐ ABAP	.abap,.ABAP
	☑ Ada	.adb,.ADB,.ada,.ADA,.ads,.ADS,.adi,.ADI
	☑ C	.c,.C
	☑ C++	.cpp,.CPP,.h,.H
	☐ MindC	.mindc,.MINDC
	☑ C#	.cs,.CS,.cscript,.CSCRIPT
	☑ Cobol	.cbl,.CBL,.cob,.COB,.cbx,.CBX,.cpy,.CPY
	☑ Java	.java,.JAVA
	☐ JavaScript	.js,.JS
	☑ Fortran77	.f,.F,.f77,.F77,.for,.FOR
	☑ Fortran90	.f95,.F95,.f90,.F90,.f03,.F03,.f08,.F08
	☐ Objective-C	.m,.M,.mm,.MM,.c,.C,.h,.H
	☑ PHP	.php,.PHP,.php5,.PHP5
	☐ PL/SQL	.sql,.SQL
	☑ Python	.py,.PY
	☐ TSQL	.tsql,.TSQL
	☑ TypeScript	.ts,.TS
	☑ VB.NET	.vb,.VB
	☐ Xaml	.xaml,.XAML

The new option for TypeScript files in Squan Sources

If you are launchin an analysis from the command line, use the language key defined in step 2 to analyse TypeScript files:

```
-d
"type=SQuORE,languages=typescript,additional_param=typescript=javasc
ript"
```

8. After the analysis finishes and you can see your artefacts in the tree, use the Dashboard Editor to build a dashboard for your new artefact type.
9. Finally, create a handler for the source code viewer to display your new file type into your configuration folder, by copying *<SQUORE_HOME>/configuration/sources/javascript_file.properties* into your own configuration as *<SQUORE_HOME>/configuration/sources/typescript_file.properties*.

Advanced COBOL Parsing

By default, Squan Sources generates artefacts for all PROGRAMs in COBOL source files. It is possible to configure the parser to also generate artefacts for all SECTIONs and PARAGRAPHs in your source code. This feature can be enabled with the following steps:

1. Open *<SQUORE_HOME>/configuration/tools/SQuORE/Analyzer/artifacts/cobol/ArtifactsList.txt*
2. Edit the list of artefacts to generate and add the section and paragraph types:

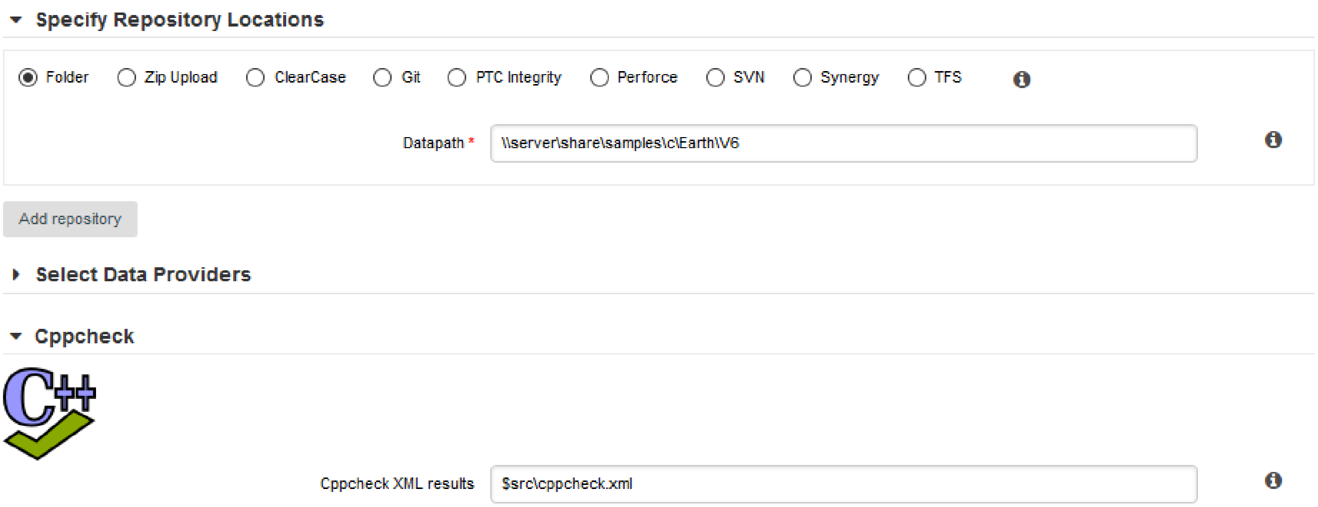
```
program
section
paragraph
```

3. Save your changes

If you create a new project, you will see the new artefacts straight away. For already-existing projects, make sure to launch a new analysis and check Squan Sources's **Force full analysis** option to parse the entire code again and generate the new artefacts.

Using Data Provider Input Files From Version Control

Input files for Squore's Data Providers, like source code, can be located in your version control system. When this is the case, you need to specify a variable in the input field for the Data Provider instead of an absolute path to the input file.



Specify Repository Locations

● Folder ○ Zip Upload ○ ClearCase ○ Git ○ PTC Integrity ○ Perforce ○ SVN ○ Synergy ○ TFS

Datapath * \\server\share\samples\c\Earth\V6

Cppcheck

Cppcheck XML results $src\cppcheck.xml

A Data Provider using an input file extracted from a remote repository

The variable to use varies depending on your scenario:

- **You have only one node of source code in your project**

In this case, the variable to use is **$src**.

- **You have more than one node of source code in your project**

  In this case, you need to tell Squore in which node the input file is located. This is done using a variable that has the same name as the alias you defined for the source code node in the previous step of the wizard. For example, if your nodes are labelled *Node1* and *Node2* (the default names), then you can refer to them using the **$Node1** and **$Node2** variables.

> When using these variables from the command line on a linux system, the *$* symbol must be escaped:
>
> ```
> -d "type=PMD,configFile=\$src/pmd_data.xml"
> ```

Providing a catalog file to a Data Provider for Offline XSL Transformations

When transforming an XML results file with an XSL stylesheet, the XML parser used by Squore will try to validate the XML file against the DTD declared in the XML header. In cases where the XSL transformation is running on a machine with no internet access, this can result in the execution of the Data Provider failing with a *No route to host* error message.

You can fix this issue by modifying the data provider to use a catalog file that will provide an alternate location for the DTD used to validate the XML. This feature can be used by all Data Providers that include an XSL transformation [1: The list includes:] .

The following example adds this functionality to the Cobertura Data Provider:

1. Add a catalog.xml file in the Data Provider's configuration folder:

```
<configuration>/tools/cobertura/catalog.xml:
<?xml version="1.0"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
    <rewriteSystem systemIdStartString="http://cobertura.sourceforge.net/xml"
rewritePrefix="./DTD"/>
</catalog>
```

2. Copy the dtd that the XML needs to validate again inside a *DTD* folder in *<configuration>/tools/cobertura/*.

The catalog file will be used the next time the Data Provider is executed and the DTD declaration will dynamically be changed from:

```
<!DOCTYPE coverage SYSTEM "http://cobertura.sourceforge.net/xml/coverage-04.dtd">
```

to:

```
<!DOCTYPE coverage SYSTEM "<configuration>/tools/cobertura/DTD/coverage-04.dtd">">
```

For more information about how to write your catalog file, refer to https://xerces.apache.org/xerces2-j/faq-xcatalogs.html.

Creating a *form.xml* for your own Data Providers, Repository Connectors and Export Definitions

All Data Providers are utilities that run during an analysis. They usually take an input file to parse or parameters specified by the user to generate output files containing violations or metrics to add to your project. Here is a non-exhaustive list of what some of them do:

- Use XSLT files to transform XML files
- Read information from Microsoft Excel files
- Parse HTML test results
- Query web services
- Export data from OSLC systems
- Launch external processes

Repository Connectors are based on the same model and are used to specifically retrieve source code and other data from source code management systems.

Export Definitions use the same *form.xml* specification to offer custom export formats to users from the web interface, dumping data from highlight definitions into a specified, custom format.

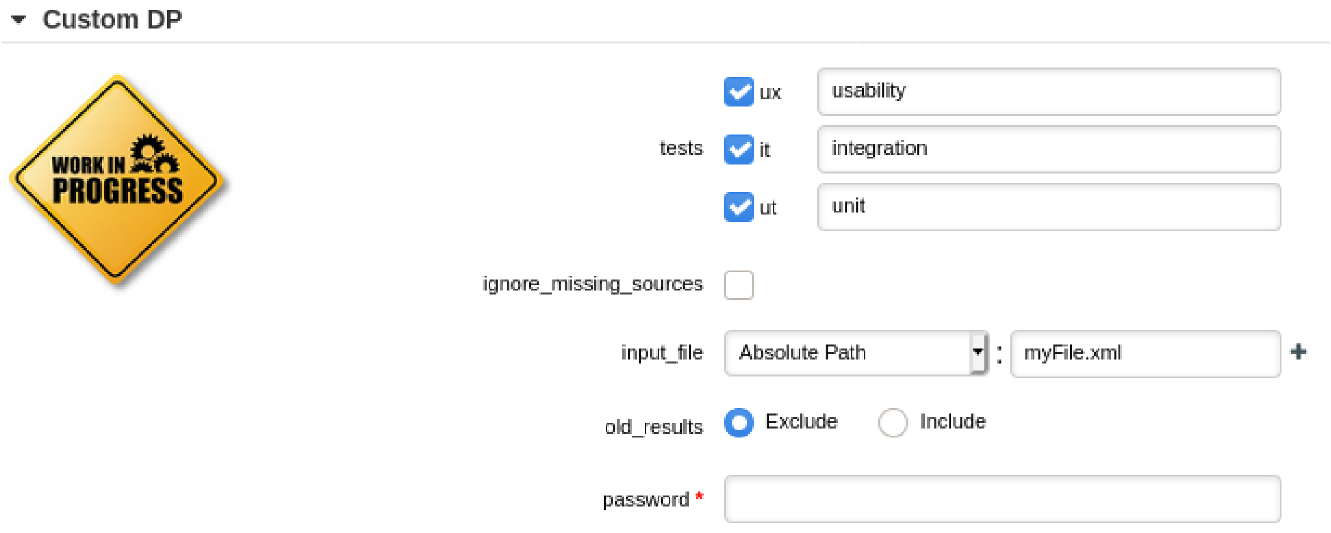
Read on to learn about how to configure your Data Provider, make it available in the web interface, and then understand how to implement the scripted part of a Data Provider that is executed during an analysis.

After you understand how to build a a Data Provider using a *form.xml* file, you can apply this knowledge to building Repository Connectors and Export Definitions, as described in Creating Repository Connectors and Creating Export Definitions.

You can find the XML schema for *form.xml* in form.xsd.

Defining Data Provider Parameters

A Data Provider's parameters are defined in a file called *form.xml*. The following is an example of *form.xml* for a Data Provider extending the GenericPerl framework:



Custom DP

tests ☑ ux usability

☑ it integration

☑ ut unit

ignore_missing_sources □

input_file Absolute Path : myFile.xml ✚

old_results ● Exclude ○ Include

password *

CustomDP parameters

```
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="GenericPerl" needSources="true" image="CustomDP.png"
projectStatusOnFailure="ERROR">
    <tag type="multipleChoice" displayType="checkbox" key="tests" optionTitle=" ">
        <value key="ux" option="usability"/>
        <value key="it" option="integration"/>
        <value key="ut" option="unit"/>
    </tag>
    <tag type="booleanChoice" key="ignore_missing_sources" defaultValue="false" />
    <tag type="file" key="input_file" defaultValue="myFile.xml" multi="true"/>
    <tag type="multipleChoice" key="old_results" style="margin-left:10px" displayType
="radioButton" defaultValue="Exclude">
        <value key="Exclude" />
        <value key="Include" />
    </tag>
    <tag type="text" key="java_path" defaultValue="/usr/bin/java" hide="true" />
    <tag type="password" required="true" key="password" />
</tags>
```

The `tags` element accepts the following attributes:

- `baseName` **(mandatory if you are not using an `exec-phase`)** indicates on which framework you are basing this Data Provider. The value of this attribute must match a folder from the *addons* folder of your installation.
- `needSources` **(optional, default: false)** allows specifying whether the Data Provider requires sources or not. When set to true, an error will be displayed if you try to select this Data Provider without adding any Repository Connector location to your project.
- `image` **(optional, default: none)** allows displaying a logo in the web UI for the Data Provider
- `projectStatusOnFailure` **(optional, default: ERROR)** defines what status the project ends in when this Data Provider produces an error. The following values are allowed:
    - **IGNORE**
    - **WARNING**
    - **ERROR**

- `projectStatusOnWarning` **(optional, default: WARNING)** defines what status the project ends in when this Data Provider produces a warning. The following values are allowed:
  - **IGNORE**
  - **WARNING**
  - **ERROR**

Each `tag` element is a Data Provider option and allows the following attributes:

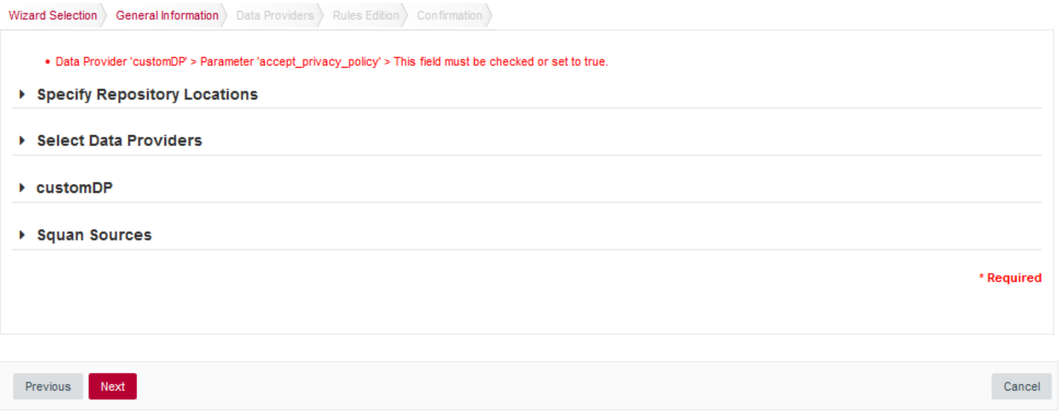
- `key` **(mandatory)** is the option's key that will be passed to the perl script, or can be used to specify the parameter's value from the command line
- `type` **(mandatory)** defines the type of the parameter. The following values are accepted:
  - **text** for free text entry
  - **file** for file path with native permission and validity checks
  - **directory** for directory path with native permission and validity checks
  - **file_or_directory** for file or directory path with native permission and validity checks
  - **password** for password fields
  - **booleanChoice** for a boolean
  - **multipleChoice** for offering a selection of predefined values

> Predefined values are specified with a `value` element with a mandatory `key` attribute and an optional `option` attribute that allows modifying the value of the option from the UI. The input field for each `option` attribute is only displayed if the parent `tag` contains an `optionTitle` attribute.

- `displayType` **(optional)** allows specifying how to display a `multipleChoice` parameter by using one of:
  - **comboBox**
  - **radioButton**
  - **checkbox**
- `multi` **(optional, default: false)** allows for dynamic addition/removal of multiple files or directories path
- `defaultValue` **(optional, default: empty)** is the value used for the parameter when not specified
- `hide` **(optional, default: false)** allows hiding a parameter from the web UI, which is useful when combining it with a default value
- `changeable` **(optional, default: true)** allows making a parameter configurable only when creating the project but read-only for following analyses when set to true
- `style` **(optional, default: empty)** allows setting basic css for the attribute in the web UI
- `required` **(optional, default: false)** allows showing a red asterisk next to the field in the web UI to make it visibly required

You can use a required tag of type **booleanchoice** to ensure that users must check a box in the web UI or set its value to *true* when building from the command line in order to proceed with the analysis.

```
<tag type="booleanChoice" required="true" key="
accept_privacy_policy" />
```



- Data Provider 'customDP' > Parameter 'accept_privacy_policy' > This field must be checked or set to true.

Specify Repository Locations

Select Data Providers

customDP

Squan Sources

* Required

Clicking the ***Next*** *button without checking a required checkbox displays an error*

Hiding your Data Provider elements in the web UI

You can associate to your tag element the displayIf element:

The `displayIf` (optional)* This element allows the user to define conditions on the tagged field to make it visible in the web UI.

The `displayIf` element accepts logical conditions. These conditions are designed as containers that can contains the following elements:

- `and` **(optional, applied by default)** all conditions defined in the "and" container must be true in order to display tagged items of the data-provider
- `or` **(optional)** one condition defined in the "or" container must be true in order to hide tagged items of the data-provider

The `displayIf` elements and conditionnal containers can accepts the following elements:

- `equals` **(optional)** this element is associated to a tag element defined in the "key" and "value" attributes. The tagged element must contains the value specified in the value attribute in order to be displayed in the web UI
- `notEmpty` **(optional)** the tag element defined in the "key" attribute has to be filed in order to display the data-provider element in the web UI

You can use the displayIf condition in a `tag` element in order to display the tagged field following conditions you have defined.

Syntax example:

```
    <tag type="text" key="config_file" hide="true"/>
    <tag type="text" key="url" required="true" >
        <displayIf>

            <notEmpty key="max_results" />

        </displayIf>
    </tag>
    <tag type="text" key="api_token" required="true" >
        <displayIf>
            <or> <!-- Conditionnal containers can be stacked  -->
                <equals key="max_results" value="100"/>
                <notEmpty key="url" />
            </or>
        </displayIf>
    </tag>
    <tag type="text" key="max_results" required="true" defaultValue="50" />
```

Localising your Data Provider

In order to display your Data Provider parameters in different languages in the web UI, yout Data Provider's *form.xml* does not contain any hard-coded strings. Instead, Squore uses each parameter's `key` attribute to dynamically retrieve a translation from a *form_xx.properties* file located next to *form.xml*.

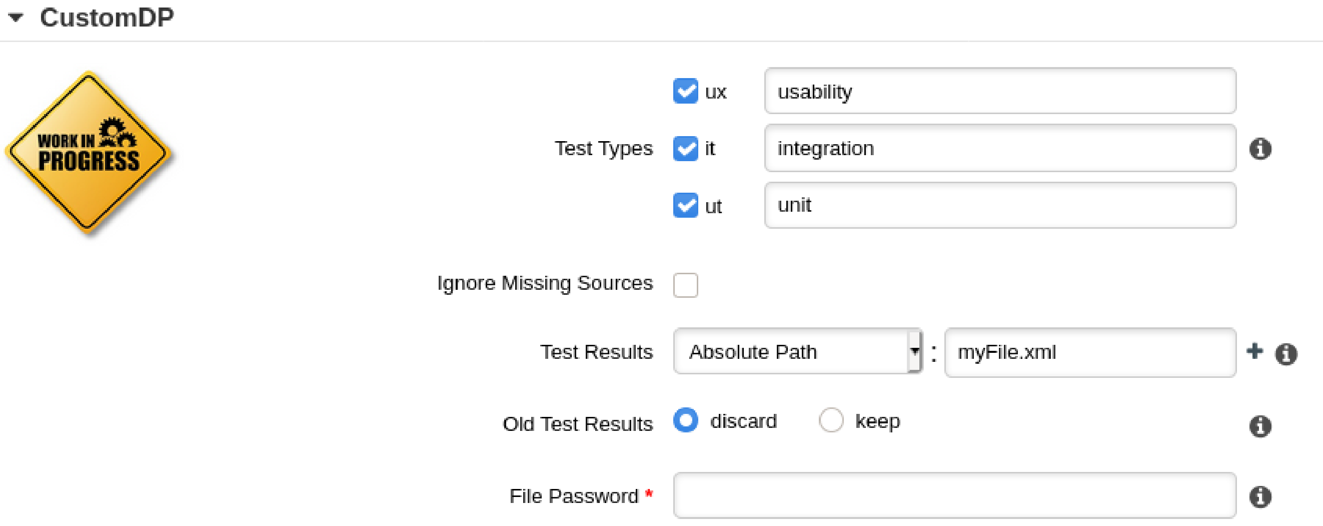
When you create a Data Provider, it is mandatory to include at least an English version of the strings in a file called *form_en.properties*. You are free to add other languages as needed. Here is a sample *.properties* for for the CustomDP you created in the previous section:

```
FORM.GENERAL.NAME = CustomDP
FORM.DASHBOARD.NAME = Test Status
FORM.GENERAL.DESCR = CustomDP imports test results for my project
FORM.GENERAL.URL  = http://example.com/CustomDP

TAG.tests.NAME = Test Types
TAG.tests.DESCR = Check the boxes next to the types of test results contained in the
results

TAG.ignore_missing_sources.NAME = Ignore Missing Sources

TAG.input_file.NAME = Test Results
TAG.input_file.DESCR = Specify the absolute path to the file containing the test
results

TAG.old_results.NAME = Old Test Results
TAG.old_results.DESCR = If the previous analysis contained results that are not in
this results file, what do you want to do with the old results?
OPT.Exclude.NAME = discard
OPT.Include.NAME = keep

TAG.password.NAME = File Password
TAG.password.DESCR = Specify the password to decrypt the test results file
```

The syntax for the *.properties* file is as follows:

- **FORM.GENERAL.NAME** is the display name of the Data Provider in the project wizard
- **FORM.DASHBOARD.NAME** is the display name of the Data Provider in the Explorer
- **FORM.GENERAL.DESCR** is the description displayed in the Data Provider's tooltip in the web UI
- **FORM.GENERAL.URL** is a reference URL for the Data Provider. Note that it is not displayed in ther web UI yet.
- **TAG.tag_name.NAME** allows setting the display name of a parameter
- **TAG.tag_name.DESCR** is a help text displayed in a tooltip next to the Data Provider option in the web UI
- **OPT.option_name.NAME** allows setting the display name of an option

Using the *form_en.properties* above for CustomDP results in the following being displayed in the web UI when launching an analysis:



▾ CustomDP

Test Types ☑ ux usability
☑ it integration
☑ ut unit

Ignore Missing Sources □

Test Results Absolute Path : myFile.xml

Old Test Results ● discard ○ keep

File Password *

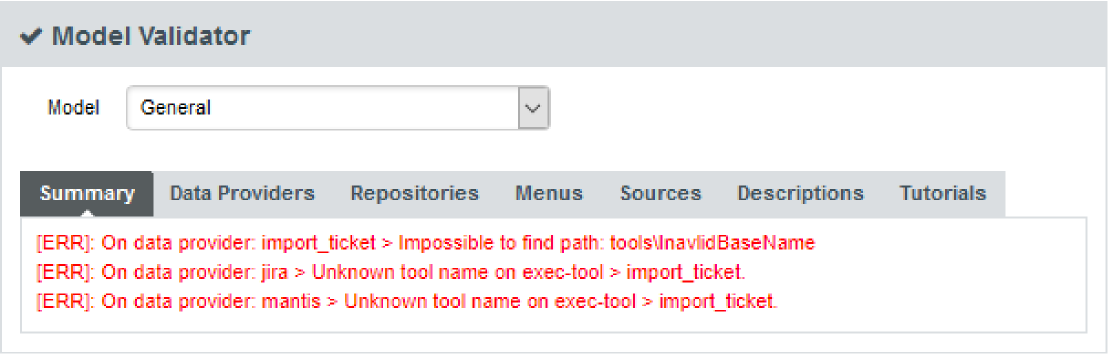
CustomDP pulling translations from a .properties file

Not all wizards display all Data Providers by default. If your Data Provider does not appear after refreshing your configuration, make sure that your wizard bundle allows displaying all Data Providers by reviewing the `tools` element of *Bundle.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>
<Bundle>
    <Wizard ... >
        ...
        <tools all="true">
            ...
        </tools>
        ...
    </Wizard>
</Bundle>
```

For more information about the wizard bundle, consult the the chapter called "Project Wizards" in the Configuration Guide.

If you have made this change and your Data Provider still does not appear in your wizard, consult the Validator to find out if it was disabled because of an error in its configuration.



✓ Model Validator

Model General

```
[ERR]: On data provider: import_ticket > Impossible to find path: tools\InavlidBaseName
[ERR]: On data provider: jira > Unknown tool name on exec-tool > import_ticket.
[ERR]: On data provider: mantis > Unknown tool name on exec-tool > import_ticket.
```

The General section of the Validator shows errors in your Data Providers

Running your Data Provider

Now that you have a new Data Provider available in the web interface (and the command line), this section will show you how to use these parameters and pass them to one or more scripts or executables in order to eventually write data in the format that Squore expects to import during the analysis.

At the end of a Data Provider execution, Squore expects a file named *input-data.xml* to be written in a specific location. The syntax of the XML file to generate is as follows:

```
<!-- input-data.xml syntax -->
<bundle version="2">
    <artifact [local-key=""] [local-parent=""|parent=""] >
        <artifact [id="<guid-stable-in-time-also-used-as-a-key>"] name="Component"
type="REQ" [location=""] >
            <info name|n="DESCR" value="The description of the object"/>
            <key value="3452-e89b-ff82"/>
            <metric name="TEST_KO" value="2"/>
            <finding name="AR120" loc="xxx" p0="The message" />
            <link name="TEST" local-src=""|src=""|local-dst=""|dst="" />
                <artifact id="" name="SubComponent" type="REQ">
                ...
                </artifact>
        </artifact>
    </artifact>

    <artifact id="" local-key="" name="" type="" local-parent=""|parent="" [location=
""] />
    ...

    <link name="" local-src=""|src="" local-dst=""|dst="" />
    ...

    <info local-ref=""|ref="" name="" value=""/>
    ...

    <metric local-ref=""|ref="" name="" value=""/>
    ...

    <finding local-ref=""|ref="" [location=""] p0="" />
    <finding local-ref=""|ref="" [location=""] p0="">
        <location local-ref=""|ref="" [location=""] />
        ...
        <relax status="RELAXED_DEROGATION|RELAXED_LEGACY|RELAXED_FALSE_POSITIVE"
><![CDATA[My Comment]]></relax>
    </finding>
    ...
</bundle>
```

You can find the XML schema for *input-data.xml* in input-data-2.xsd.

Your Data Provider is configured by adding an exec-phase element with a mandatory id="add-data" attribute in *form.xml*.

The basic syntax of an exec-phase can be seen below:

```
<exec-phase id="add-data">
    <exec name="tcl|perl|java" | executable="/path/to/bin" | executable=
"executable_name" failOnError="true|false" failOnStdErr="true|false" warn="[WARN]"
error="[ERROR|ERR]" fatal="[FATAL]">
        <arg value="${<function>(<args>)}"/>
        <arg value="-freeText" />
        <arg value="${<predefinedVars>}" />
        <arg value="versions" />
        <arg value="-myTag"/>
        <arg tag="myTag"/>
        <env key="MY_VAR" value="SOME_VALUE"/>
    </exec>
    <exec ... />
    <exec-tool name="another_data_provider">
        <param key="<tagName>" value="<value>" />
        <param key="<tagName>" tag="<tag>" />
        <param ... />
    </exec-tool>
    <exec-tool ... >
        ...
    </exec-tool>
</exec-phase>
```

You can also use Groovy in order to configure your Data Provider.

The basic syntax of a Groovy exec name is indicated below:

```
<exec name="java">
            <arg value="${javaClasspath(poi,groovy,jackson)}"/>
            <arg value="groovy.lang.GroovyShell" />
            <arg value="${getConfigFile(to_excel.groovy)}"/>
            <arg value="${getSharedAddonsFile(GroovyScriptUtils.groovy)}"/>
...
```

Only the exec name section is different. The syntax of the others sections of your Data Provider is still the same.

Executables

The exec-phase element accepts one or more launches of scripts or executables specified in an exec child element, that can receive arguments and environment variables specified via arg and env elements.

There are four built-in languages for executables:

- **tcl**

- **perl**
- **java**
- **Groovy**

The scripts are launched using the tcl, perl, or java runtimes defined in your Squore installation. This is also the case for Groovy, which is handled by Java engine.

The following attributes of the `exec` element allow you to control error handling:

- `failOnError` **(optional, default: true)** marks the Data Provider execution as failed if the executable returns an error code
- `failOnStdErr` **(optional, default: true)** marks the Data Provider execution as failed if the executable prints something to stdErr during the execution
- `warn` **,** `error` **and** `fatal` **(optional, default: see code block above**) allow you to define patterns to look for in the executable's standard output to fine-tune the result of the execution.

Other executables can be called, as long as they are available on the system's PATH, or configured in *config.xml*

Given the following *config.xml*:

```
<!-- config.xml (server or cli) -->
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<squore type="server" version="1.3">
    <paths>
        <path name="python" path="C:\Python\python.exe" />
        <path name="git" path="C:\Git\bin\git.exe" />
    </paths>
...
</squore>
```

git and python can be called in your Data Provider as follows:

```
<exec-phase id="add-data">
    <exec name="git">
        ...
    </exec>
    <exec name="python">
        ...
    </exec>
</exec-phase>
```

Arguments

Argument values can be:

1. Free text passed in a `value` tag, useful to specify a parameter for your script

```
<exec name="perl">
    <arg value="-V" />
</exec>
```

2. A tag key declared in *form.xml* passed as a `tag ` attribute to retrieve the input specified by the user. If no input was specified, you can define a `defaultValue`:

   ```
   <arg tag="maxValue" defaultValue="50" />
   <arg tag="configFile" defaultValue="${getConfigFile(default.xml)}" />
   ```

3. One of the predefined functions

   - **${getOutputFile(<relative/path/to/file>,<abortIfMissing>)}** returns the absolute path of an *input-data.xml* file output by an `exec-phase` . *failIfMissing* is an optional boolean which aborts the execution when set to **true** if the file is missing.
   - **${getTemporaryFile(<relative/path/to/file>)}** returns the absolute path of a temporary file created by an `exec` (only for *add-data* and *repo-add-data* phases)
   - **${getAddonsFile(<relative/path/to/file>)}** returns the absolute path of a file in the Data Provider's addons folder
   - **${getConfigFile(<relative/path/to/file>)}** returns the absolute path of a file in the Data Provider's configuration folder
   - **${getSharedAddonsFile(<relative/path/to/file>)}** returns the absolute path of a file in Data Provider's addons/shared folder, if not returns the absolute path of a file in addons/shared folder
   - **${path(<executable_name>)}** returns the absolute path of an executable configured in *config.xml*, or just the executable name if the executable is available from the system's PATH.

     ```
     <exec name="...">
         <arg value="-git_path" />
         <arg value="${path(git)}" />
     ```

   - **${javaClasspath(poi,groovy,jackson,abc.jar,xyz.jar)}** adds the specified list of jars to the classpath for java execution.

     Squore will look for the jars in the *addons/lib* folder of your configuration and return a classpath parameter for the desired runtime environment (*-cp="..." for java*)

     > 💡
     >
     > **poi** is a shortcut for *poi-ooxml-3.17.jar,poi-3.17.jar,poi-ooxml-schemas-3.17.jar,xmlbeans-2.6.0.jar,commons-collections4-4.1.jar* and configures the environment necessary to use Apache POI when creating custom Export Definitions, as described in Creating Export Definitions.
     >
     > **groovy** is a shortcut for groovy-3.0.1.jar, groovy-json-3.0.1.jar and groovy-xml-3.0.1.jar libraries needed to run Groovy scripts
     >
     > **jackson** is a shortcut for jackson-core-2.6.3.jar, jackson-databind-2.6.3.jar and jackson-annotations-2.6.0.jar libraries needed to parse Json file

4. One of the predefined variables

   - **${tmpDirectory}** to get an absolute path to a temp folder to create files

  - **${sourcesList}** to get a list of the aliases and locations containing the data extracted by the repository connectors used in the analysis
  - **${outputDirectory}** to get the absolute path of folder where the Data Provider needs to write the final *input-data.xml*

Conditions

You can use condition statements in the `exec` and `exec-tool` elements in order to parametrize the execution of your Data Providers. The `execute-if` element is used as follow :

```
<exec-phase id="add-data">
    <exec name="java">
        <executeIf>
            <equals key="outputFile" value="" />   <!-- Execute this Java process only
if the output file is not provided. -->
        </executeIf>
        ...
    </exec>
</exec-phase>
```

The `execute-if` element uses the same syntax as the `displayIf` element : Hiding your Data Provider elements in the web UI.

Calling Other Data Providers

You can call and pass parameters to other Data Providers after your `exec-phase` using an `exec-tool` element. The `exec-tool` element uses a mandatory `name` which is the name of the folder containing the other Data Provider to launch in your configuration folder and supports passing the parameters expected by the other Data Provider via one or more `param` elements where:

- `key` is the name of the parameter expected by the other Data Provider (as defined in its *form.xml*)
- `value` allows passing free text
- `tag` allows passing the value of your own Data Provider's tag value to the other Data Provider and can be combined with a `defaultValue` attribute in case no value was specified by the user for the tag

As an example, the following Data Provider generates a CSV file that is then passed to the pep8 Data Provider:

```
<exec-phase id="add-data">
    <exec name="python">
        <arg value="consolidate-reports-recursive.py" />
        <arg value="-folders" />
        <arg tag="root_folder" />
        <arg value="-outputFile" />
        <arg value="output.csv" />
    </exec>
    <exec-tool name="pep8">
        <param key="csv" value="${getOutputFile(output.csv)}" />
        <param key="separator" tag="separator" defaultValue=";" />
    </exec-tool>
</exec-phase>
```

In this other example, a perl script is launched to retrieves issues from a ticketing system and the export data is passed to the **import_ticket** Data Provider:

```
<exec-phase id="add-data">
    <exec name="perl">
        <arg value="${getConfigFile(export_ticket.pl)}" />
        <arg value="-url" />
        <arg tag="url" />
        <arg value="-login" />
        <arg tag="login" />
        <arg value="-pwd" />
        <arg tag="pwd" />
        <arg value="-outputFile" />
        <arg value="${getOutputFile(exportdata.csv,false)}" />
    </exec>
    <exec-tool name="import_ticket">
        <param key="input_file" value="${getOutputFile(exportdata.csv)}" />
        <param key="csv_separator" value=";" />
    </exec-tool>
</exec-phase>
```

If your Data Provider uses a perl script, Squore provides a small library that makes it easy to retrieve script arguments called **SQuORE::Args**. Using it as part of your script, you can retrieve arguments using the **get_tag_value()** function, as shown below:

```
# name: export_ticket.pl
# description: exports issues to a CSV file
use SQuORE::Args;
# ...
# ...
my $url = get_tag_value("url");
my $login = get_tag_value("login");
my $pwd = get_tag_value("pwd");
my $outputFile = get_tag_value("outputFile");
# ...
exit 0;
```

Using the Squore toolkit

If you want your Data Provider to use the Squore toolkit to retrieve references to artefacts, the following variables are available (in the *add-data* and *repo-add-data* phases only):

- **${tclToolkitDirectory}:** the directory of the toolkit tcl code to execute
- **${squanOutputDirectory}:** the directory of containing the results of the execution of Squan Sources

In order to use the toolkit, your `exec` must use the tcl language. As an example, here is a sample `exec-phase` and associated tcl file to get you started:

```
 <!-- form.xml -->
<exec-phase id="repo-add-data">
    <exec name="tcl">
        <arg value="${getAddonsFile(repo-add-data.tcl)}" />
        <arg value="${tclToolkitFile}" />
        <arg value="${squanOutputDirectory}" />
        <arg value="${outputDirectory}" />
        <arg tag="xxx" />
    </exec>
</exec-phase>
```

```
#repo-add-data.tcl:
set toolkitFile [lindex $argv 0]
set sqOutputDir [lindex $argv 1]
set outputDir [lindex $argv 2]
set xxx [lindex $argv 3]

# Initialise the toolkit
puts "Initializing toolkit"
source $toolkitFile
toolkit::initialize $sqOutputDir $outputDir

# Execute your code
puts "Main execution"
# yout code here
# ...

# Generate xml files (artefacts)
puts "Generating xml files"
toolkit::generate $outputDir {artefacts}
```

Finding More Examples

If you want to find more examples of working Data Providers that use this syntax, check the following Data Providers in Squore's default configuration folder:

- **conf-checker** calls a jar file to write an XML file in Squore's exchange format
- **import_ticket** parses a file to translate it into a format that can then be passed to **csv_import** to import the tickets into Squore
- **jira** retrieves data from Jira and passes it to **import_ticket**

Built-in Data Provider Frameworks

In order to help you import data into Squore, the following Data Provider frameworks are provided and can write a valid *input-data.xml* file for you:

1. **csv_import**

   The csv_import framework allows you to write Data Providers that produce CSV files and then pass them on to the framework to be converted to an XML format that Squore understands. This framework allows you to import metrics, findings, textual information and links as well as generate your own artefacts. It is fully linked to the source code parser and therefore allows to locate existing source code artefacts generated by the source code parser. Refer to the full csv_import Reference for more information.

2. **xml**

   The xml framework is a sample implementation of a Data Provider that allows you to directly import an XML file or run it through an XSL transformation to that it matches the input format expected by Squore (input-data.xml). This framework therefore allows you to import metrics, findings, textual information and links as well as generate your own artefacts. Refer to the full xml Reference for more information.

💡 If you are looking for the legacy Data Provider frameworks from previous versions of Squore, consult Legacy Frameworks.

The legacy Data Provider frameworks are still supported, however using the new frameworks is recommended for developping new Data Providers, as they are more flexible and provide more functionality to interact with source code artefacts.

Creating Repository Connectors

The same syntax used to create Data Providers can be used to create Repository Connectors, and therefore instruct Squore to get source code from SCMs. Instead of using an `exec-phase` with the `id="add-data"`, your Repository Connector should define the following phases:

- `id="import"` defines how you extract source code and make it available to Squan Sources so it can be analysed. This phase is expected to return a path to a folder containing the sources to analyse or a *data.properties* file listing the path to the folder containing source and various other properties to be used in other executions:

  ```
  directory=/path/to/sources-to-analyse
  data.<key1>=<value1>
  data.<key2>=<value2>
  ```

  This phase is executed once per source code node in the project and allows you to use the following additional variables: **${outputSourceDirectory} is the folder containing the sources to analyse** **${alias}** is the alias used for the source code node (empty if there is only one source code node)

- `id="repo-add-data"` is similar to the *add-data* phase described for Data Providers in Running your Data Provider and is expected to produce an *input-data.xml*. The only difference in the case of a Repository Connector is that this phase is executed once per source code node in the analysis.

- `id="display"` is the phase that is called when users request to view the source code for an artefact from the web UI. This phase is expected to return a *data.properties* file with the following keys:

  ```
  filePath=/path/to/source/file
  displayPath=<Artefact Display Path (optional)>
  ```

  The contents of *filePath* will be loaded in the source code viewer, while the value of *displayPath* will be used as the file path displayed in the header of the source code viewer.

  This phase allows you to use the following additional variables:

- **${scaInfo}** is text to display in the title bar of the source code viewer in the web interface
- **${artefactName}** is the name of the file to display
- **${artefactPath}** is the path (without the alias) of the file to display

  During the **display** phase, you can retrieve any data set during the **import** phase for the repository using the **${getImportData(<key1>)}** function

Additional attributes are available for the `tags` element of a Repository Connector:

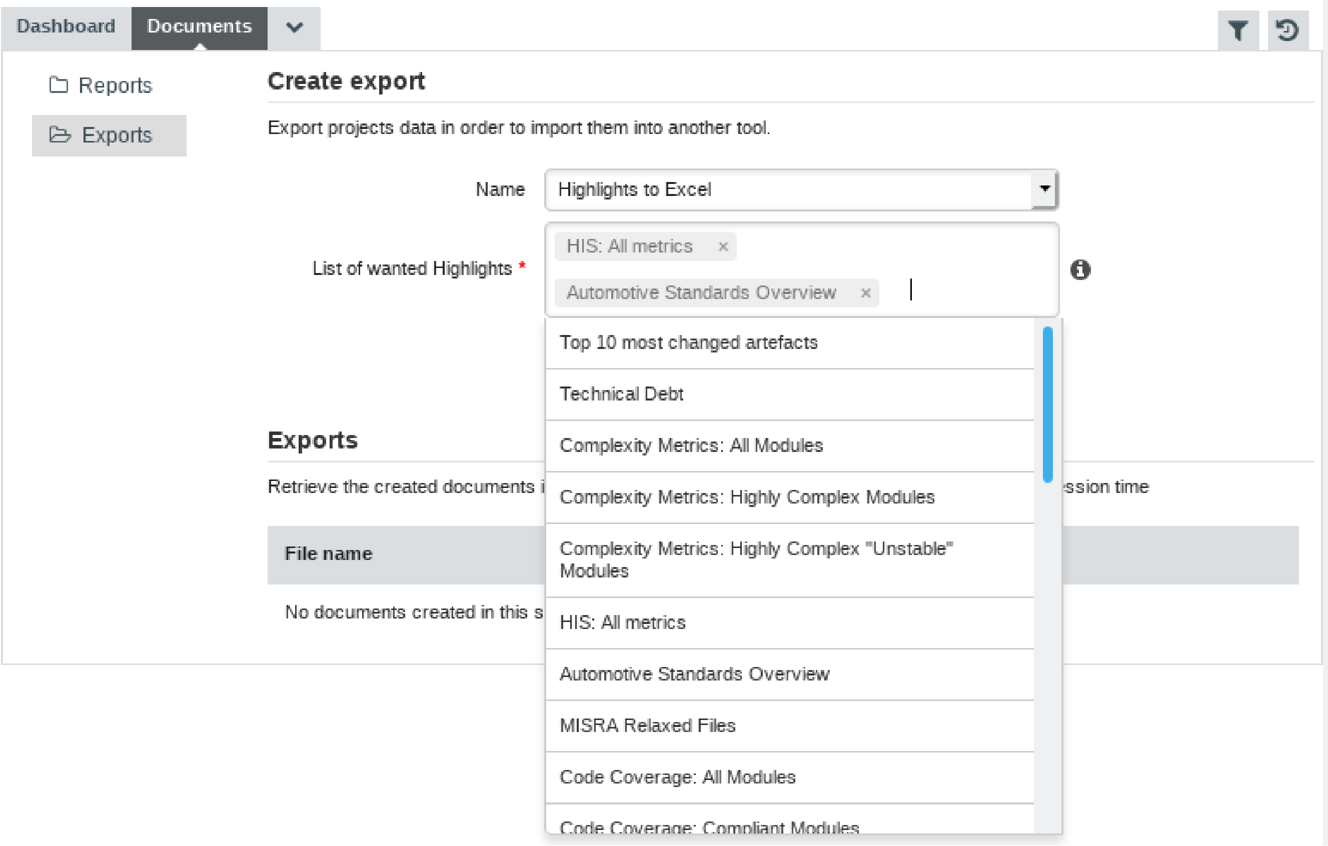
- `deleteTmpSrc` **(optional, default: false)** indicates whether or not the content of **sources** folder coming from this Repository Connector will be deleted upon Squore Server restart.

- `useCredentialsForSCA` **(optional, default: true)** allows specifying whether credentials dialog will be prompted in View Source Code or not.

> Consult SVN's *form.xml* in *<SQUORE_HOME>/configuration/repositoryConnectors/SVN* for a working example of a Repository Connector that uses all the phases described above.
>
> Please note, as data-provider, you can use the <exec-tool> parameter in order to call other elements while processing, like Data Provider or Scripts. For more informations about <exec-tool> parameter, please refer to Running your Data Provider.

Creating Export Definitions

The *form.xml* specification can also be used to create Export Definitions that allow users to export data based on one or more highlight categories from the web interface.



Create export

Export projects data in order to import them into another tool.

Name: Highlights to Excel

List of wanted Highlights *: HIS: All metrics ×, Automotive Standards Overview ×

- Top 10 most changed artefacts
- Technical Debt
- Complexity Metrics: All Modules
- Complexity Metrics: Highly Complex Modules
- Complexity Metrics: Highly Complex "Unstable" Modules
- HIS: All metrics
- Automotive Standards Overview
- MISRA Relaxed Files
- Code Coverage: All Modules
- Code Coverage: Compliant Modules

Exports

Retrieve the created documents i ... ssion time

File name
No documents created in this s

The ***Highlights to Excel*** *Export Definition*

The **Highlights to Excel** Export Definition uses the following *form.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>
<tags>
    <tag type="multipleChoice" displayType="multi-autocomplete" required="true" key=
"highlights">
        <values type="highlights" />
    </tag>

    <exec-phase id="export">
        <exec name="java">
```

```
            <arg value="${javaClasspath(poi,groovy,jackson)}"/>
            <arg value="groovy.lang.GroovyShell" />
            <arg value="${getConfigFile(to_excel.groovy)}"/>
            <arg value="${getSharedAddonsFile(GroovyScriptUtils.groovy)}"/>

            <arg value="-importScript"/>
            <arg value="${getSharedAddonsFile(exports_utils.groovy)}" />

            <arg value="-squoreApiUtils"/>
            <arg value="${getSharedAddonsFile(SquoreApiUtils.groovy)}" />

            <arg value="-excelUtilsScript"/>
            <arg value="${getSharedAddonsFile(ExcelUtils.groovy)}" />

            <arg value="-highlights"/>
            <arg tag="highlights" />

            <arg value="-outputDirectory" />
            <arg value="${outputDirectory}" />

            <arg value="-idArtefact" />
            <arg value="${idArtefact}"/>

            <arg value="-idVersion"/>
            <arg value="${idVersion}"/>

            <arg value="-idModel"/>
            <arg value="${idModel}"/>

            <arg value="-group"/>
            <arg value="${group}"/>

            <arg value="-serverUrl"/>
            <arg value="${localUrl}"/>

            <arg value="-token"/>
            <arg value="${token}"/>

            <arg value="-template"/>
            <arg value="${getConfigFile(template.xlsx)}" />
        </exec>
    </exec-phase>
</tags>
```

Data is exported from the server as a JSON file, which your Export Definition can modify as needed before sending it to the end-user who launched the export. You can consult the format of the JSON file in the **Data Exchange Formats** appendix for more information.

In order to create an Export Definition, the syntax described in Defining Data Provider Parameters and Running your Data Provider is augmented to include the extra additional capabilities:

1. A **multi-autocomplete**`displayType` for multipleChoice `tag` elements.

   The `tag` element accepts a `values` sub-element with a mandatory `type` attributes. When set to **highlights**, the widget automatically displays all the available highlight definitions for the currently selected artefact.

2. A mandatory `exec-phase` with `id="export"` that contains one or more `exec`s.

   This `exec-phase` is expected to return a data.properties file with the following keys:

   ```
   filename=/path/to/export/file
   ```

3. Variables that can be used in the `exec-phase` to pass the context of the currently selected artefact to the Export Definition:
   - **${idUser}** is the ID of the user generating the export
   - **${token}** is the auto-generated token for on the fly authentification to the API REST
   - **${idArtefact}** is the ID of the currently selected artefact
   - **${idVersion}** is the ID of the version of the project that is currently selected
   - **${idApplication}** is the ID of the project that currently selected
   - **${idModel}** is the ID of the analysis model used for the project that is currently selected
   - **${group}** is the path of the current selected group portfolio
   - **${serverUrl}** is the Squore Server URL, as defined in **Administration > System**
   - **${localUrl}** is the Squore Local URL

You can add your own Export Definition by following these steps:

1. Create a folder in *configuration/exports* called *my_export_definition*.
2. Create a *form.xml* and *form_en.properties* in *my_export_definition*
3. Define the `exec-phase` that your Export Definition will run
4. Add your Export Definition to your model's Export Bundle for the desired project role and artefact type, using the folder name (*my_export_definition*) as the `ExportDef`'s `name` attribute:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <Bundle>
   <Role name="DEFAULT">
       <Export type="...">
           <ExportDef name="my_export_definition" />
           ...
       </Export>
       ...
   </Role>
   </Bundle>
   ```

5. Reload the Squore configuration and your Export Definition should appear in the Documents tab of the Explorer.

For more examples of custom Export Definitions, consult the *configuration/exports* and *addons/exports* folders of the default Squore configuration.

Please note, as data-provider, you can use the <exec-tool> parameter in order to call other elements while processing, like Data Provider or Scripts. For more informations about <exec-tool> parameter, please refer to Running your Data Provider.

Appendix A: Data Provider Frameworks

Current Frameworks

The following Data Provider frameworks support importing all kinds of data into Squore. Whether you choose one or the other depends on the ability of your script or executable to produce CSV or XML data. Note that these frameworks are recommended over the legacy frameworks described in Legacy Frameworks, which are deprecated as of Squore 18.0.

csv_import Reference

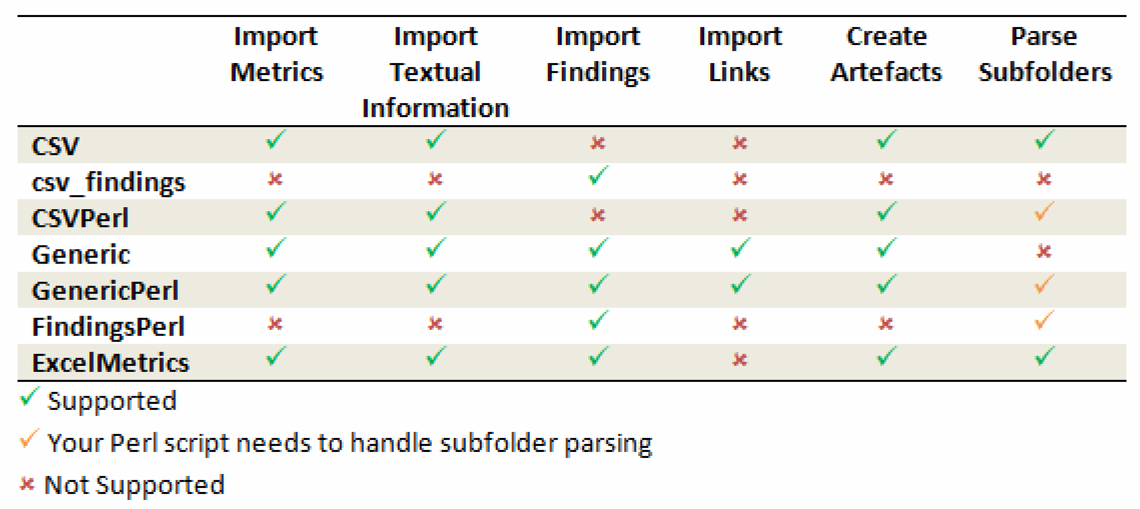
```
==============
= csv_import =
==============

The csv_import framework allows you to create Data Providers that produce CSV files
that the framework will translate into XML files that can be imported in your analysts
results. This framework is useful if writing XML files directly from your script is
not practical.

Using csv_import, you can import metrics, findings (including relaxed findings),
textual information, and links between artefacts (including to and from source code
artefacts).
This framework replaces all the legacy frameworks that wrote CSV files in previous
versions.

Note that this framework can be called by your Data Provider simply by creating an
exec-tool phase that calls the part of the framework located in the configuration
folder:
<exec-tool name="csv_import">
    <param key="csv" value="${getOutputFile(output.csv)}" />
    <param key="separator" value=";" />
    <param key="delimiter" value="&quot;" />
</exec-tool>

For a full description of all the parameters that can be used, consult the section
called "CSV Import" in the "Data Providers" chapter of this manual.


============================================
= CSV format expected by the data provider =
============================================

- Line to define an artefact (like a parent artefact for instance):
Artefact

- Line to add n metrics to an artefact:
Artefact;(MetricId;Value)*
```

```
- Line to add n infos to an artefact:
Artefact;(InfoId;Value)*

- Line to add a key to an artefact:
Artefact;Value

- Line to add a finding to an artefact:
Artefact;RuleId;Message;Location

- Line to add a relaxed finding to an artefact:
Artefact;RuleId;Message;Location;RelaxStatus;RelaxMessage

- Line to add a link between artefacts:
Artefact;LinkId;Artefact

where:
- MetricId is the id of the metric as declared in the Analysis Model
- InfoId is the id of the information to import
- Value is the value of the metric or the information or the key to import (a key is a
UUID used to reference an artefact)
- RuleId is the id of the rule violated as declared in the Analysis Model
- Message is the message of the finding, which is displayed after the rule description
- Location is the location of the finding (a line number for findings attached source
code artefacts, a url for findings attached to any other kind of artefact)
- RelaxStatus is one of DEROGATION, FALSE_POSITIVE or LEGACY and defines the
relaxation stat of the imported finding
- RelaxMessage is the justification message for the relaxation state of the finding
- LinkId is the id of the link to create between artefacts, as declared in the
Analysis Model

==========================
= Manipulating Artefacts =
==========================

The following functions are available to locate and manipulate source code artefacts
in the project:
- ${artefact(type,path)} ==> Identify an artefact by its type and full path
- ${artefact(type,path,uid)} ==> Identify an artefact by its type and full path and
assign it the unique identifier uid
- ${uid(value)} ==> Identify an artefact by its unique identifier (value)
- ${file(path)} ==> Tries to find a source code file matching the "path" in the
project
- ${function(fpath,line)} ==> Tries to find a source code function at line "line" in
file matching the "fpath" in the project
- ${function(fpath,name)} ==> Tries to find a source code function whose name matches
"name" in the file matching the "fpath" in the project
- ${class(fpath,line)} ==> Tries to find a source code class at line "line" in the
file matching the "fpath" in the project
- ${class(fpath,name)} ==> Tries to find a source code class whose name matches "name"
in the file matching the "fpath" in the project
```

```
===============
= Input Files =
===============

The data provider accepts the following files:
Metrics file accepts:
    Artefact definition line
    Metrics line

Findings file accepts:
    Artefact definition line
    Findings line

Keys file accepts:
    Artefact definition line
    Keys line

Information file accepts:
    Artefact definition line
    Information line

Links file accepts:
    Artefact definition line
    Links line

It is also possible to mix every kind of line in a single csv file, as long as each
line is prefixed with the kind of data it contains.
In this case, the first column must contain one of:
DEFINE (or D): when the line is used to define an artefact
METRIC (or M): to add a metric
INFO (or I): to add an information
KEY (or K): to add a key
FINDING (or F): to add a finding, relaxed or not
LINK (or L): to add link between artefacts

The following is an example of a csv file containing mixed lines:
D;${artefact(CR_FOLDER,/CRsCl)}
M;${artefact(CR,/CRsCl/cr2727,2727)};NB;2
M;${artefact(CR,/CRsCl/cr1010,1010)};NB;4
I;${uid(1010)};NBI;Bad weather
K;${artefact(CR,/CRsCl/cr2727,2727)};#CR2727
I;${artefact(CR,/CRsCl/cr2727,2727)};NBI;Nice Weather
F;${artefact(CR,/CRsCl/cr2727,2727)};BAD;Malformed
M;${uid(2727)};NB_EXT;3
I;${uid(2727)};NBI_EXT;Another Info
F;${uid(2727)};BAD_EXT;Badlyformed
F;${uid(2727)};BAD_EXT1;Badlyformed1;;FALSE_POSITIVE;Everything is in the title]]>
F;${function(machine.c,41)};R_GOTO;"No goto; neither togo;";41
F;${function(machine.c,42)};R_GOTO;No Goto;42;LEGACY;Was done a long time ago
L;${uid(1010)};CR2CR;${uid(2727)}
L;${uid(2727)};CR2CR;${uid(1010)}
```

```
=======
= xml =
=======

The xml framework is an implementation of a data provider that allows to import an xml
file, potentially after an xsl transformation. The transformed XML file is expected to
follow the syntax expected by other data providers (see input-data.xml specification).

This framework can be extended like the other frameworks, by creating a folder for
your data provider in your configuration/tools folder and creating a form.xml.
Following are three examples of the possible uses of this framework.

Example 1 - User enters an xml path and an xsl path, the xml is transformed using the
xsl and then imported
=========
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="xml">
  <tag type="text" key="xml" />
    <tag type="text" key="xslt" />

    <exec-phase id="add-data">
      <exec name="java" failOnError="true" failOnStdErr="true">
        <arg value="${javaClasspath(groovy,xml-resolver-1.2.jar)}"/>
        <arg value="groovy.lang.GroovyShell" />
        <arg value="xml.groovy" />
        <arg value="${outputDirectory}" />
        <arg tag="xml"/>
        <arg tag="xsl" />
      </exec>
    </exec-phase>
</tags>

Example 2 - The user enter an xml path, the xsl file is predefined (input-data.xsl)
and present in the same directory as form.xml
=========
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="xml">
  <tag type="text" key="xml" />

  <exec-phase id="add-data">
    <exec name="java" failOnError="true" failOnStdErr="true">
      <arg value="${javaClasspath(groovy,xml-resolver-1.2.jar)}"/>
      <arg value="groovy.lang.GroovyShell" />
      <arg value="xml.groovy" />
      <arg value="${outputDirectory}" />
      <arg tag="xml" />
      <arg value="${getToolConfigDir(input-data.xsl)}" />
    </exec>
  </exec-phase>
</tags>

Example 3 - The user enter an xml path of a file already in the expected format
=========
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="xml">
  <tag type="text" key="xml" />

  <exec-phase id="add-data">
    <exec name="java" failOnError="true" failOnStdErr="true">
      <arg value="${javaClasspath(groovy,xml-resolver-1.2.jar)}"/>
      <arg value="groovy.lang.GroovyShell" />
      <arg value="xml.groovy" />
      <arg value="${outputDirectory}" />
      <arg tag="xml" />
    </exec>
  </exec-phase>
</tags>
```

Legacy Frameworks



	Import Metrics	Import Textual Information	Import Findings	Import Links	Create Artefacts	Parse Subfolders
CSV	✓	✓	✗	✗	✓	✓
csv_findings	✗	✗	✓	✗	✗	✗
CSVPerl	✓	✓	✗	✗	✓	✔
Generic	✓	✓	✓	✓	✓	✗
GenericPerl	✓	✓	✓	✓	✓	✔
FindingsPerl	✗	✗	✓	✗	✗	✔
ExcelMetrics	✓	✓	✓	✗	✓	✓

✓ Supported

✔ Your Perl script needs to handle subfolder parsing

✗ Not Supported

Legacy Data Provider frameworks and their capabilities

1. **Csv**

   The Csv framework is used to import metrics or textual information and attach them to artefacts of type Application or File. While parsing one or more input CSV files, if it finds the same metric for the same artefact several times, it will only use the last occurrence of the metric and ignore the previous ones. Note that the type of artefacts you can attach metrics to is limited to Application and File artefacts. If you are working with File artefacts, you can let the Data Provider create the artefacts by itself if they do not exist already. Refer to the full Csv Reference for more information.

2. **csv_findings**

   The csv_findings framework is used to import findings in a project and attach them to artefacts of type Application, File or Function. It takes a single CSV file as input and is the only framework that allows you to import relaxed findings directly. Refer to the full csv_findings

Reference for more information.

3. **CsvPerl**

   The CsvPerl framework offers the same functionality as Csv, but instead of dealing with the raw input files directly, it allows you to run a perl script to modify them and produce a CSV file with the expected input format for the Csv framework. Refer to the full CsvPerl Reference for more information.

4. **FindingsPerl**

   The FindingsPerl framework is used to import findings and attach them to existing artefacts. Optionally, if an artefact cannot be found in your project, the finding can be attached to the root node of the project instead. When launching a Data Provider based on the FindingsPerl framework, a perl script is run first. This perl script is used to generate a CSV file with the expected format which will then be parsed by the framework. Refer to the full FindingsPerl Reference for more information.

5. **Generic**

   The Generic framework is the most flexible Data Provider framework, since it allows attaching metrics, findings, textual information and links to artefacts. If the artefacts do not exist in your project, they will be created automatically. It takes one or more CSV files as input (one per type of information you want to import) and works with any type of artefact. Refer to the full Generic Reference for more information.

6. **GenericPerl**

   The GenericPerl framework is an extension of the Generic framework that starts by running a perl script in order to generate the metrics, findings, information and links files. It is useful if you have an input file whose format needs to be converted to match the one expected by the Generic framework, or if you need to retrieve and modify information exported from a web service on your network. Refer to the full GenericPerl Reference for more information.

7. **ExcelMetrics**

   The ExcelMetrics framework is used to extract information from one or more Microsoft Excel files (.xls or .xslx). A detailed configuration file allows defining how the Excel document should be read and what information should be extracted. This framework allows importing metrics, findings and textual information to existing artefacts or artefacts that will be created by the Data Provider. Refer to the full ExcelMetrics Reference for more information.

After you choose the framework to extend, you should follow these steps to make your custom Data Provider known to Squore:

1. Create a new configuration *tools* folder to save your work in your custom configuration folder: *MyConfiguration/configuration/tools*.
2. Create a new folder for your data provider inside the new *tools* folder: **CustomDP**. This folder needs to contain the following files:
   - **form.xml** defines the input parameters for the Data Provider, and the base framework to use, as described in Defining Data Provider Parameters
   - **form_en.properties** contains the strings displayed in the web interface for this Data Provider, as described in Localising your Data Provider
   - **config.tcl** contains the parameters for your custom Data Provider that are specific to the selected framework
   - **CustomDP.pl** is the perl script that is executed automatically if your custom Data Provider uses one of the *Perl frameworks.
3. Edit Squore Server's configuration file to register your new configuration path, as described in the Installation and Administration Guide.

4. Log into the web interface as a Squore administrator and reload the configuration.

Your new Data Provider is now known to Squore and can be triggered in analyses. Note that you may have to modify your Squore configuration to make your wizard aware of the new Data Provider and your model aware of the new metrics it provides. Refer to the relevant sections of the Configuration Guide for more information.

Csv Reference

```
=======
= Csv =
=======

The Csv framework is used to import metrics or textual information and attach them to
artefacts of type Application, File or Function. While parsing one or more input CSV
files, if it finds the same metric for the same artefact several times, it will only
use the last occurrence of the metric and ignore the previous ones. Note that the type
of artefacts you can attach metrics to is limited to Application, File and Function
artefacts. If you are working with File artefacts, you can let the Data Provider
create the artefacts by itself if they do not exist already.

============
= form.xml =
============

You can customise form.xml to either:
- specify the path to a single CSV file to import
- specify a pattern to import all csv files matching this pattern in a directory

In order to import a single CSV file:
=====================================
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="Csv" needSources="true">
    <tag type="text" key="csv" defaultValue="/path/to/mydata.csv" />
</tags>

Notes:
- The csv key is mandatory.
- Since Csv-based data providers commonly rely on artefacts created by Squan Sources,
you can set the needSources attribute to force users to specify at least one
repository connector when creating a project.

In order to import all files matching a pattern in a folder:
===========================================================
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="Csv" needSources="true">
    <!-- Root directory containing Csv files to import-->
    <tag type="text" key="dir" defaultValue="/path/to/mydata" />
    <!-- Pattern that needs to be matched by a file name in order to import it-->
    <tag type="text" key="ext" defaultValue="*.csv" />
    <!-- search for files in sub-folders -->
    <tag type="booleanChoice" defaultValue="true" key="sub" />
</tags>

Notes:
- The dir and ext keys are mandatory
- The sub key is optional (and its value set to false if not specified)



==============
= config.tcl =
==============

Sample config.tcl file:
=======================
# The separator used in the input CSV file
# Usually \t or ;
set Separator "\t"

# The delimiter used in the input CSV file
# This is normally left empty, except when you know that some of the values in the CSV
file
# contain the separator itself, for example:
# "A text containing ; the separator";no problem;end
# In this case, you need to set the delimiter to \" in order for the data provider to
find 3 values instead of 4.
# To include the delimiter itself in a value, you need to escape it by duplicating it,
for example:
# "A text containing "" the delimiter";no problemo;end
# Default: none
set Delimiter \"

# ArtefactLevel is one of:
#      Application: to import data at application level
#      File: to import data at file level. In this case ArtefactKey has to be set
#             to the value of the header (key) of the column containing the file path
#             in the input CSV file.
#      Function : to import data at function level, in this case:
#                 ArtefactKey has to be set to the value of the header (key) of the
column containing the path of the file
#                 FunctionKey has to be set to the value of the header (key) of the
column containing the name and signature of the function
# Note that the values are case-sensitive.
set ArtefactLevel File
set ArtefactKey File

# Should the File paths be case-insensitive?
# true or false (default)
# This is used when searching for a matching artefact in already-existing artefacts.
set PathsAreCaseInsensitive "false"
```

```
# Should file artefacts declared in the input CSV file be created automatically?
# true (default) or false
set CreateMissingFile "true"

# FileOrganisation defines the layout of the input CSV file and is one of:
#     header::column: values are referenced from the column header
#     header::line: NOT AVAILABLE
#     alternate::line: lines are a sequence of {Key Value}
#     alternate::column: columns are a sequence of {Key Value}
# There are more examples of possible CSV layouts later in this document
set FileOrganisation header::column

# Metric2Key contains a case-sensitive list of paired metric IDs:
#     {MeasureID KeyName [Format]}
# where:
#   - MeasureID is the id of the measure as defined in your analysis model
#   - KeyName, depending on the FileOrganisation, is either the name of the column or
the name
#      in the cell preceding the value to import as found in the input CSV file
#   - Format is the optional format of the data, the only accepted format
#      is "text" to attach textual information to an artefact, for normal metrics omit
this field
set Metric2Key {
    {BRANCHES Branchs}
    {VERSIONS Versions}
    {CREATED Created}
    {IDENTICAL Identical}
    {ADDED Added}
    {REMOV Removed}
    {MODIF Modified}
    {COMMENT Comment text}
}


==========================
= Sample CSV Input Files =
==========================

Example 1:
==========
 FileOrganisation : header::column
 ArtefactLevel : File
 ArtefactKey   : Path

Path    Branchs Versions
./foo.c 15      105
./bar.c 12      58

Example 2:
==========
 FileOrganisation : alternate::line
 ArtefactLevel : File
 ArtefactKey   : Path

Path    ./foo.c Branchs 15  Versions    105
Path    ./bar.c Branchs 12  Versions    58

Example 3:
==========
 FileOrganisation : header::column
 ArtefactLevel : Application

ChangeRequest   Corrected   Open
27              15          11

Example 4:
==========
 FileOrganisation : alternate::column
 ArtefactLevel : Application

ChangeRequest   15
Corrected       11

Example 5:
==========
 FileOrganisation : alternate::column
 ArtefactLevel : File
 ArtefactKey   : Path

Path    ./foo.c
Branchs 15
Versions    105
Path    ./bar.c
Branchs 12
Versions    58

Example 6:
==========
 FileOrganisation : header::column
 ArtefactLevel : Function
 ArtefactKey   : Path
 FunctionKey   : Name

Path    Name    Decisions Tested
./foo.c end_game(int*,int*) 15      3
./bar.c bar(char)   12      6

Working With Paths:
===================

- Path seperators are unified: you do not need to worry about handling differences
between Windows and Linux
```

```
- With the option PathsAreCaseInsensitive, case is ignored when searching for files in
the Squore internal data
- Paths known by Squore are relative paths starting at the root of what was specified
in the repository connector durign the analysis. This relative path is the one used to
match with a path in a csv file.

Here is a valid example of file matching:
  1. You provide C:\A\B\C\D as the root folder in a repository connector
  2. C:\A\B\C\D contains E\e.c then Squore will know E/e.c as a file

  3. You provide a csv file produced on linux and containing
    /tmp/X/Y/E/e.c as path, then Squore will be able to match it with the known file.

Squore uses the longest possible match.
In case of conflict, no file is found and a message is sent to the log.
```

csv_findings Reference

```
================
= csv_findings =
================

The csv_findings data provider is used to import findings (rule violations) and attach
them to artefacts of type Application, File or Function.
The format of the csv file given as parameter has to be:

FILE;FUNCTION;RULE_ID;MESSAGE;LINE;COL;STATUS;STATUS_MESSAGE;TOOL

where:
=====
FILE : is the full path of the file where the finding is located
FUNCTION : is the name of the function where the finding is located
RULE_ID : is the Squore ID of the rule which is violated
MESSAGE : is the specific message of the violation
LINE: is the line number where the violation occurs
COL: (optional, leave empty if not provided) is the column number where the violation
occurs
STATUS: (optional, leave empty if not provided) is the staus of the relaxation if the
violation has to be relaxed (DEROGATION, FALSE_POSITIVE, LEGACY)
STATUS_MSG: (optional, leave empty if not provided) is the message for the relaxation
when relaxed
TOOL: is the tool providing the violation

The header line is read and ignored (it has to be there)
The separator (semicolon by default) can be changed in the config.tcl file (see below)
The delimiter (no delimiter by default) can be changed in the config.tcl (see below)

==============
= config.tcl =
==============

Sample config.tcl file:
=======================
# The separator used in the input CSV file
# Usually ; or \t
set Separator \;

# The delimiter used in the CSV input file
# This is normally left empty, except when you know that some of the values in the CSV
file
# contain the separator itself, for example:
# "A text containing ; the separator";no problem;end
# In this case, you need to set the delimiter to \" in order for the data provider to
find 3 values instead of 4.
# To include the delimiter itself in a value, you need to escape it by duplicating it,
for example:
# "A text containing "" the delimiter";no problemo;end
# Default: none
set Delimiter \"

# You can add some patterns to avoid new findings when some strings in the finding
message changes
# i.e. Unreachable code Default switch clause is unreachable. switch-expression at
line 608 (column 12).
# In this case we do not want the line number to be part of the signagture of the
finding,
# to achieve this user will add a pattern as shown below (patterns are TCL regex
patterns):
lappend InconstantFindingsPatterns {at line [0-9]+}
```

CsvPerl Reference

```
===========
= CsvPerl =
===========

The CsvPerl framework offers the same functionality as Csv, but instead of dealing
with the raw input files directly, it allows you to run a perl script to modify them
and produce a CSV file with the expected input format for the Csv framework.



============
= form.xml =
============

In your form.xml, specify the input parameters you need for your Data Provider.
```

```
Our example will use two parameters: a path to a CSV file and another text parameter:

<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="CsvPerl" needSources="true">
    <tag type="text" key="csv" defaultValue="/path/to/csv" />
    <tag type="text" key="param" defaultValue="MyValue" />
</tags>

- Since Csv-based data providers commonly rely on artefacts created by Squan Sources,
you can set the needSources attribute to force users to specify at least one
repository connector when creating a project.



==============
= config.tcl =
==============

Refer to the description of config.tcl for the Csv framework.

For CsvPerl one more option is possible:

# The variable NeedSources is used to request the perl script to be executed once for
each
# repository node of the project. In that case an additional parameter is sent to the
# perl script (see below for its position)
#set ::NeedSources 1



==========================
= Sample CSV Input Files =
==========================

Refer to the examples for the Csv framework.



===============
= Perl Script =
===============

The perl scipt will receive as arguments:
 - all parameters defined in form.xml (as -${key} $value)
 - the input directory to process (only if ::NeedSources is set to 1 in the config.tcl
file)
 - the location of the output directory where temporary files can be generated
 - the full path of the csv file to be generated

For the form.xml we created earlier in this document, the command line will be:
 perl <configuration_folder>/tools/CustomDP/CustomDP.pl -csv /path/to/csv -param
MyValue <output_folder> <output_folder>/CustomDP.csv

Example of perl script:
```

```
======================
#!/usr/bin/perl
use strict;
use warnings;
$|=1 ;

($csvKey, $csvValue, $paramKey, $paramValue, $output_folder, $output_csv) = @ARGV;

    # Parse input CSV file
    # ...

    # Write results to CSV
    open(CSVFILE, ">" . ${output_csv}) || die "perl: can not write: $!\n";
    binmode(CSVFILE, ":utf8");
    print CSVFILE  "ChangeRequest;15";
    close CSVFILE;

exit 0;
```

Generic Reference

```
===========
= Generic =
===========

The Generic framework is the most flexible Data Provider framework, since it allows
attaching metrics, findings, textual information and links to artefacts. If the
artefacts do not exist in your project, they will be created automatically. It takes
one or more CSV files as input (one per type of information you want to import) and
works with any type of artefact.


============
= form.xml =
============

In form.xml, allow users to specify the path to a CSV file for each type of data you
want to import.
You can set needSources to true or false, depending on whether or not you want to
require the use of a repository connector when your custom Data Provider is used.

Example of form.xml file:
=========================
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="Generic" needSources="false">
    <!-- Path to CSV file containing Metrics data -->
    <tag type="text" key="csv" defaultValue="mydata.csv" />
    <!-- Path to CSV file containing Findings data: -->
    <tag type="text" key="fdg" defaultValue="mydata_fdg.csv" />
    <!-- Path to CSV file containing Information data: -->
    <tag type="text" key="inf" defaultValue="mydata_inf.csv" />
    <!-- Path to CSV file containing Links data: -->
    <tag type="text" key="lnk" defaultValue="mydata_lnk.csv" />
</tags>

Note: All tags are optional. You only need to specify the tag element for the type of
data you want to import with your custom Data Provider.



==============
= config.tcl =
==============

Sample config.tcl file:
=======================
# The separator used in the input csv files
# Usually \t or ; or ,
# In our example below, a space is used.
set Separator " "

# The delimiter used in the input CSV file
# This is normally left empty, except when you know that some of the values in the CSV
file
# contain the separator itself, for example:
# "A text containing ; the separator";no problem;end
# In this case, you need to set the delimiter to \" in order for the data provider to
find 3 values instead of 4.
# To include the delimiter itself in a value, you need to escape it by duplicating it,
for example:
# "A text containing "" the delimiter";no problemo;end
# Default: none
set Delimiter \"

# The path separator in an artefact's path
# in the input CSV file.
# Note that artefact is spellt with an "i"
# and not an "e" in this option.
set ArtifactPathSeparator "/"

# If the data provider needs to specify a different toolName (optional)
set SpecifyToolName 1

# Metric2Key contains a case-sensitive list of paired metric IDs:
#     {MeasureID KeyName [Format]}
# where:
#   - MeasureID is the id of the measure as defined in your analysis model
#   - KeyName is the name in the cell preceding the value to import as found in the
input CSV file
#   - Format is the optional format of the data, the only accepted format
#      is "text" to attach textual information to an artefact. Note that the same
result can also
#       be achieved with Info2Key (see below). For normal metrics omit this field.
set Metric2Key {
    {CHANGES Changed}
}

# Finding2Key contains a case-sensitive list of paired rule IDs:
#     {FindingID KeyName}
# where:
#   - FindingID is the id of the rule as defined in your analysis model
#   - KeyName is the name in the finding name in the input CSV file
set Finding2Key {
    {R_NOTLINKED NotLinked}
}

# Info2Key contains a case-sensitive list of paired info IDs:
#     {InfoID KeyName}
# where:
#   - InfoID is the id of the textual information as defiend in your analysis model
#   - KeyName is the name of the information name in the input CSV file
set Info2Key
    {SPECIAL_LABEL Label}
}

# Ignore findings for artefacts that are not part of the project (orphan findings)
# When set to 1, the findings are ignored
# When set to 0, the findings are imported and attached to the APPLICATION node
# (default: 1)
set IgnoreIfArtefactNotFound 1

# If data in csv concerns source code artefacts (File, Class or Function), the way to
# match file paths can be case-insensitive
# true or false (default)
# This is used when searching for a matching artefact in already-existing artefacts.
set PathsAreCaseInsensitive "false"

# For findings of a type that is not in your ruleset, set a default rule ID.
# The value for this parameter must be a valid rule ID from your analysys model.
# (default: empty)
set UnknownRuleId UNKNOWN_RULE

# Save the total count of orphan findings as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanArteCountId NB_ORPHANS

# Save the total count of unknown rules as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesCountId NB_UNKNOWN_RULES

# Save the list of unknown rule IDs as textual information at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesListId UNKNOWN_RULES_INFO



====================
=  CSV File Format =
====================

All the examples listed below assume the use of the following config.tcl:

set Separator ","
set ArtifactPathSeparator "/"
set Metric2Key {
    {CHANGES Changed}
}
set Finding2Key {
    {R_NOTLINKED NotLinked}
}
set Info2Key
    {SPECIAL_LABEL Label}
}

How to reference an artefact:
============================
 ==> artefact_type artefact_path
 Example:
 REQ_MODULES,Requirements
 REQ_MODULE,Requirements/Module
 REQUIREMENT,Requirements/Module/My_Req

 References the following artefact
 Application
      Requirements (type: REQ_MODULES)
            Module (type: REQ_MODULE)
                My_Req  (type: REQUIREMENT)

Note: For source code artefacts there are 3 special artefact kinds:
 ==> FILE file_path
 ==> CLASS file_path (Name|Line)
 ==> FUNCTION file_path (Name|Line)

 Examples:
 FUNCTION src/file.c 23
 references the function which contains line 23 in the source file src/file.c, if no
 function found the line whole line of the csv file is ignored.
 FUNCTION src/file.c foo()
 references a function named foo in source file src/file.c. If more than one function
foo
 is defined in this file, then the signature of the function (which is optional) is
used
 to find the best match.

Layout for Metrics File:
========================
 ==> artefact_type artefact_path (Key Value)*

 When the parent artefact type is not given it defaults to <artefact_type>_FOLDER.
 Example:
 REQ_MODULE,Requirements/Module
 REQUIREMENT,Requirements/Module/My_Req,Changed,1

 will produce the following artefact tree:
 Application
      Requirements (type: REQ_MODULE_FOLDER)
          Module (type: REQ_MODULE)
              My_Req : (type: REQUIREMENT) with 1 metric CHANGES = 1

Note: the key "Changed" is mapped to the metric "CHANGES", as specified by the
Metric2Key parameter, so that it matches what is expected by the model.


Layout for Findings File:
=========================
 ==> artefact_type artefact_path key message

 When the parent artefact type is not given it defaults to <artefact_type>_FOLDER.
 Example:
 REQ_MODULE,Requirements/Module
 REQUIREMENT,Requirements/Module/My_Req,NotLinked,A Requirement should always been
linked

 will produce the following artefact tree:
 Application
      Requirements (type: REQ_MODULE_FOLDER)
          Module (type: REQ_MODULE)
              My_Req (type: REQUIREMENT) with 1 finding R_NOTLINKED whose description
is "A Requirement should always been linked"

Note: the key "NotLinked" is mapped to the finding "R_NOTLINKED", as specified by the
Finding2Key parameter, so that it matches what is expected by the model.

Layout for Textual Information File:
====================================
 ==> artefact_type artefact_path label value
 When the parent artefact type is not given it defaults to <artefact_type>_FOLDER.
 Example:
 REQ_MODULE,Requirements/Module
 REQUIREMENT,Requirements/Module/My_Req,Label,This is the label of the req

 will produce the following artefact tree:
 Application
      Requirements (type: REQ_MODULE_FOLDER)
          Module (type: REQ_MODULE)
              My_Req (type: REQUIREMENT) with 1 information of type SPECIAL_LABEL
whose content is "This is the label of the req"

Note: the label "Label" is mapped to the finding "SPECIAL_LABEL", as specified by the
Info2Key parameter, so that it matches what is expected by the model.

Layout for Links File:
======================
 ==> artefact_type artefact_path dest_artefact_type dest_artefact_path link_type

 When the parent artefact type is not given it defaults to <artefact_type>_FOLDER
 Example:
 REQ_MODULE Requirements/Module
 TEST_MODULE Tests/Module
 REQUIREMENT Requirements/Module/My_Req TEST Tests/Module/My_test TESTED_BY

 will produce the following artefact tree:
 Application
    Requirements (type: REQ_MODULE_FOLDER)
        Module (type: REQ_MODULE)
            My_Req (type: REQUIREMENT) ------>
    Tests (type: TEST_MODULE_FOLDER)           |
        Module (type: TEST_MODULE)             |
            My_Test (type: TEST) <------------+ link (type: TESTED_BY)

The TESTED_BY relationship is created with My_Req as source of the link and My_test as
the destination

CSV file organisation when SpecifyToolName is set to 1
======================================================
When the variable SpecifyToolName is set to 1 (or true) a column has to be added
at the beginning of each line in each csv file. This column can be empty or filled
with a different toolName.

 Example:
 ,REQ_MODULE,Requirements/Module
 MyReqChecker,REQUIREMENT,Requirements/Module/My_Req Label,This is the label of the
req

The finding of type Label will be set as reported by the tool "MyReqChecker".
```

GenericPerl Reference

```
===============
= GenericPerl =
===============

The GenericPerl framework is an extension of the Generic framework that starts by
running a perl script in order to generate the metrics, findings, information and
links files. It is useful if you have an input file whose format needs to be converted
to match the one expected by the Generic framework, or if you need to retrieve and
modify information exported from a web service on your network.



============
= form.xml =
============

In your form.xml, specify the input parameters you need for your Data Provider.
Our example will use two parameters: a path to a CSV file and another text parameter:

<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="CsvPerl" needSources="false">
    <tag type="text" key="csv" defaultValue="/path/to/csv" />
    <tag type="text" key="param" defaultValue="MyValue" />
</tags>



==============
= config.tcl =
==============

Refer to the description of config.tcl for the Generic framework for the basic
options.
Additionally, the following options are available for the GenericPerl framework, in
order to know which type of information your custom Data Provider should try to
import.

# If the data provider needs to specify a different toolName (optional)
#set SpecifyToolName 1

# Set to 1 to import metrics csv file, 0 otherwise



# ImportMetrics
# When set to 1, your custom Data Provider (CustomDP) will try to import
# metrics from a file called CustomDP.mtr.csv that your perl script
# should generate according to the expected format described in the
# documentation of the Generic framework.
set ImportMetrics 1
```

```
# ImportInfos
# When set to 1, your custom Data Provider (CustomDP) will try to import
# textual information from a file called CustomDP.inf.csv that your perl script
# should generate according to the expected format described in the
# documentation of the Generic framework.
set ImportInfos 0

# ImportFindings
# When set to 1, your custom Data Provider (CustomDP) will try to import
# findings from a file called CustomDP.fdg.csv that your perl script
# should generate according to the expected format described in the
# documentation of the Generic framework.
set ImportFindings 1

# ImportLinks
# When set to 1, your custom Data Provider (CustomDP) will try to import
# artefact links from a file called CustomDP.lnk.csv that your perl script
# should generate according to the expected format described in the
# documentation of the Generic framework.
set ImportLinks 0

# Ignore findings for artefacts that are not part of the project (orphan findings)
# When set to 1, the findings are ignored
# When set to 0, the findings are imported and attached to the APPLICATION node
# (default: 1)
set IgnoreIfArtefactNotFound 1

# For findings of a type that is not in your ruleset, set a default rule ID.
# The value for this parameter must be a valid rule ID from your analysys model.
# (default: empty)
set UnknownRuleId UNKNOWN_RULE

# Save the total count of orphan findings as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanArteCountId NB_ORPHANS

# Save the total count of unknown rules as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesCountId NB_UNKNOWN_RULES

# Save the list of unknown rule IDs as textual information at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesListId UNKNOWN_RULES_INFO
```

```
====================
=  CSV File Format =
====================

Refer to the examples in the Generic framework.



===============
= Perl Script =
===============

The perl scipt will receive as arguments:
- all parameters defined in form.xml (as -${key} $value)
- the location of the output directory where temporary files can be generated
- the full path of the metric csv file to be generated (if ImportMetrics is set to 1
in config.tcl)
- the full path of the findings csv file to be generated (if ImportFindings is set to
1 in config.tcl)
- the full path of the textual information csv file to be generated (if ImportInfos is
set to 1 in config.tcl)
- the full path of the links csv file to be generated (if ImportLinks is set to 1 in
config.tcl)
- the full path to the output directory used by this data provider in the previous
analysis

For the form.xml and config.tcl we created earlier in this document, the command line
will be:
 perl <configuration_folder>/tools/CustomDP/CustomDP.pl -csv /path/to/csv -param
MyValue <output_folder> <output_folder>/CustomDP.mtr.csv
<output_folder>/CustomDP.fdg.csv <previous_output_folder>

The following perl functions are made available in the perl environment so you can use
them in your script:
- get_tag_value(key) (returns the value for $key parameter from your form.xml)
- get_output_metric()
- get_output_finding()
- get_output_info()
- get_output_link()
- get_output_dir()
- get_input_dir() (returns the folder containing sources if needSources is set to 1)
- get_previous_dir()


Example of perl script:
======================
#!/usr/bin/perl
use strict;
use warnings;
$|=1 ;

    # Parse input CSV file
    my $csvFile = get_tag_value("csv");
    my $param = get_tag_value("param");
    # ...

    # Write metrics to CSV
    open(METRICS_FILE, ">" . get_output_metric()) || die "perl: can not write: $!\n";
    binmode(METRICS_FILE, ":utf8");
    print METRICS_FILE  "REQUIREMENTS;Requirements/All_Requirements;NB_REQ;15";
    close METRICS_FILE;

    # Write findings to CSV
    open(FINDINGS_FILE, ">" . get_output_findings()) || die "perl: can not write:
$!\n";
    binmode(FINDINGS_FILE, ":utf8");
    print FINDINGS_FILE  "REQUIREMENTS;Requirements/All_Requirements;R_LOW_REQS;\"The
minimum number of requirement should be at least 25.\"";
    close FINDINGS_FILE;

exit 0;
```

FindingsPerl Reference

```
================
= FindingsPerl =
================

The FindingsPerl framework is used to import findings and attach them to existing
artefacts. Optionally, if an artefact cannot be found in your project, the finding can
be attached to the root node of the project instead. When launching a Data Provider
based on the FindingsPerl framework, a perl script is run first. This perl script is
used to generate a CSV file with the expected format which will then be parsed by the
framework.



============
= form.xml =
============

In your form.xml, specify the input parameters you need for your Data Provider.
Our example will use two parameters: a path to a CSV file and another text parameter:

<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="CsvPerl" needSources="true">
    <tag type="text" key="csv" defaultValue="/path/to/csv" />
    <tag type="text" key="param" defaultValue="MyValue" />
</tags>

- Since FindingsPerl-based data providers commonly rely on artefacts created by Squan
```

```
Sources, you can set the needSources attribute to force users to specify at least one
repository connector when creating a project.



==============
= config.tcl =
==============



Sample config.tcl file:
=======================
# The separator to be used in the generated CSV file
# Usually \t or ;
set Separator ";"

# The delimiter used in the input CSV file
# This is normally left empty, except when you know that some of the values in the CSV
file
# contain the separator itself, for example:
# "A text containing ; the separator";no problem;end
# In this case, you need to set the delimiter to \" in order for the data provider to
find 3 values instead of 4.
# To include the delimiter itself in a value, you need to escape it by duplicating it,
for example:
# "A text containing "" the delimiter";no problemo;end
# Default: none
set Delimiter \"

# Should the perl script execcuted once for each repository node of the project ?
# 1 or 0 (default)
# If true an additional parameter is sent to the
# perl script (see below for its position)
set ::NeedSources 0

# Should the violated rules definitions be generated?
# true or false (default)
# This creates a ruleset file with rules that are not already
# part of your analysis model so you can review it and add
# the rules manually if needed.
set generateRulesDefinitions false

# Should the File paths be case-insensitive?
# true or false (default)
# This is used when searching for a matching artefact in already-existing artefacts.
set PathsAreCaseInsensitive false

# Should file artefacts declared in the input CSV file be created automatically?
# true (default) or false
set CreateMissingFile true

# Ignore findings for artefacts that are not part of the project (orphan findings)
# When set to 0, the findings are imported and attached to the APPLICATION node
instead of the real artefact
# When set to 1, the findings are not imported at all
# (default: 0)
set IgnoreIfArtefactNotFound 0

# For findings of a type that is not in your ruleset, set a default rule ID.
# The value for this parameter must be a valid rule ID from your analysis model.
# (default: empty)
set UnknownRuleId UNKNOWN_RULE

# Save the total count of orphan findings as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanArteCountId NB_ORPHANS

# Save the total count of unknown rules as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesCountId NB_UNKNOWN_RULES

# Save the list of unknown rule IDs as textual information at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesListId UNKNOWN_RULES_INFO

# The tool version to specify in the generated rules definitions
# The default value is ""
# Note that the toolName is the name of the folder you created
# for your custom Data Provider
set ToolVersion ""

# FileOrganisation defines the layout of the CSV file that is produced by your perl
script:
#     header::column: values are referenced from the column header
#     header::line: NOT AVAILABLE
#     alternate::line: NOT AVAILABLE
#     alternate::column: NOT AVAILABLE
set FileOrganisation header::column

# In order to attach a finding to an artefact of type FILE:
#   - Tool (optional) if present it overrides the name of the tool providing the
finding
#   - Path has to be the path of the file
#   - Type has to be set to FILE
#   - Line can be either empty or the line in the file where the finding is located
#   Rule is the rule identifier, can be used as is or translated using Rule2Key
#   Descr is the description message, which can be empty
```

```
#
# In order to attach a finding to an artefact of type FUNCTION:
#   - Tool (optional) if present it overrides the name of the tool providing the
finding
#   - Path has to be the path of the file containing the function
#   - Type has to be FUNCTION
#   - If line is an integer, the system will try to find an artefact function
#       at the given line of the file
#   - If no Line or Line is not an integer, Name is used to find an artefact in
#       the given file having name and signature as found in this column.
# (Line and Name are optional columns)

# Rule2Key contains a case-sensitive list of paired rule IDs:
#     {RuleID KeyName}
# where:
#   - RuleID is the id of the rule as defined in your analysis model
#   - KeyName is the rule ID as written by your perl script in the produced CSV file
# Note: Rules that are not mapped keep their original name. The list of unmapped rules
is in the log file generated by your Data Provider.
set Rule2Key {
    {   ExtractedRuleID_1   MappedRuleId_1   }
    {   ExtractedRuleID_2   MappedRuleId_2   }
}


====================
=  CSV File Format =
====================

According to the options defined earlier in config.tcl, a valid csv file would be:

Path;Type;Line;Name;Rule;Descr
/src/project/module1/f1.c;FILE;12;;R1;Rule R1 is violated because variable v1
/src/project/module1/f1.c;FUNCTION;202;;R4;Rule R4 is violated because function f1
/src/project/module2/f2.c;FUNCTION;42;;R1;Rule R1 is violated because variable v2
/src/project/module2/f2.c;FUNCTION;;skip_line(int);R1;Rule R1 is violated because
variable v2

Working With Paths:
===================

- Path seperators are unified: you do not need to worry about handling differences
between Windows and Linux
- With the option PathsAreCaseInsensitive, case is ignored when searching for files in
the Squore internal data
- Paths known by Squore are relative paths starting at the root of what was specified
in the repository connector durign the analysis. This relative path is the one used to
match with a path in a csv file.

Here is a valid example of file matching:
  1. You provide C:\A\B\C\D as the root folder in a repository connector
  2. C:\A\B\C\D contains E\e.c then Squore will know E/e.c as a file

  3. You provide a csv file produced on linux and containing
    /tmp/X/Y/E/e.c as path, then Squore will be able to match it with the known file.

Squore uses the longest possible match.
In case of conflict, no file is found and a message is sent to the log.



===============
= Perl Script =
===============

The perl scipt will receive as arguments:
    - all parameters defined in form.xml (as -${key} $value)
    - the input directory to process (only if ::NeedSources is set to 1)
    - the location of the output directory where temporary files can be generated
    - the full path of the findings csv file to be generated

For the form.xml and config.tcl we created earlier in this document, the command line
will be:
 perl <configuration_folder>/tools/CustomDP/CustomDP.pl -csv /path/to/csv -param
MyValue <output_folder> <output_folder>/CustomDP.fdg.csv
<output_folder>/CustomDP.fdg.csv

Example of perl script:
======================
#!/usr/bin/perl
use strict;
use warnings;
$|=1 ;

($csvKey, $csvValue, $paramKey, $paramValue, $output_folder, $output_csv) = @ARGV;

    # Parse input CSV file
    # ...

    # Write results to CSV
    open(CSVFILE, ">" . ${output_csv}) || die "perl: can not write: $!\n";
    binmode(CSVFILE, ":utf8");
    print CSVFILE "Path;Type;Line;Name;Rule;Descr";
    print CSVFILE "/src/project/module1/f1.c;FILE;12;;R1;Rule R1 is violated because
variable v1";
    close CSVFILE;

exit 0;
```

ExcelMetrics Reference

```
================
= ExcelMetrics =
================

The ExcelMetrics framework is used to extract information from one or more Microsoft
Excel files (.xls or .xslx). A detailed configuration file allows defining how the
Excel document should be read and what information should be extracted. This framework
allows importing metrics, findings and textual information to existing artefacts or
artefacts that will be created by the Data Provider.


============
= form.xml =
============

You can customise form.xml to either:
- specify the path to a single Excel file to import
- specify a pattern to import all Excel files matching this pattern in a directory

In order to import a single Excel file:
=====================================
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="ExcelMetrics" needSources="false">
    <tag type="text" key="excel" defaultValue="/path/to/mydata.xslx" />
</tags>

Notes:
- The excel key is mandatory.

In order to import all files matching a patter in a folder:
===========================================================
<?xml version="1.0" encoding="UTF-8"?>
<tags baseName="ExcelMetrics" needSources="false">
    <!-- Root directory containing Excel files to import-->
    <tag type="text" key="dir" defaultValue="/path/to/mydata" />
    <!-- Pattern that needs to be matched by a file name in order to import it-->
    <tag type="text" key="ext" defaultValue="*.xlsx" />
    <!-- search for files in sub-folders -->
    <tag type="booleanChoice" defaultValue="true" key="sub" />
</tags>


Notes:
- The dir and ext keys are mandatory
- The sub key is optional (and its value set to false if not specified)


==============
= config.tcl =
==============
Sample config.tcl file:
=======================
# The separator to be used in the generated csv file
# Usually \t or ; or ,
set Separator  ";"

# The delimiter used in the input CSV file
# This is normally left empty, except when you know that some of the values in the CSV
file
# contain the separator itself, for example:
# "A text containing ; the separator";no problem;end
# In this case, you need to set the delimiter to \" in order for the data provider to
find 3 values instead of 4.
# To include the delimiter itself in a value, you need to escape it by duplicating it,
for example:
# "A text containing "" the delimiter";no problemo;end
# Default: none
set Delimiter \"

# The path separator in an artefact's path
# in the generated CSV file.
set ArtefactPathSeparator "/"

# Ignore findings for artefacts that are not part of the project (orphan findings)
# When set to 1, the findings are ignored
# When set to 0, the findings are imported and attached to the APPLICATION node
# (default: 1)
set IgnoreIfArtefactNotFound 1

# For findings of a type that is not in your ruleset, set a default rule ID.
# The value for this parameter must be a valid rule ID from your analysys model.
# (default: empty)
set UnknownRuleId UNKNOWN_RULE

# Save the total count of orphan findings as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanArteCountId NB_ORPHANS

# Save the total count of unknown rules as a metric at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesCountId NB_UNKNOWN_RULES

# Save the list of unknown rule IDs as textual information at application level
# Specify the ID of the metric to use in your analysys model
# to store the information
# (default: empty)
set OrphanRulesListId UNKNOWN_RULES_INFO
```

```
# The list of the Excel sheets to read, each sheet has the number of the first line to
read
# A Perl regexp pattern can be used instead of the name of the sheet (the first sheet
matching
# the pattern will be considered)
set Sheets {{Baselines 5} {ChangeNotes 5}}

# ######################
# # COMMON DEFINITIONS #
# ######################
#
# - <value> is a list of column specifications whose values will be concatened. When
no column name is present, the
#         text is taken as it appears. Optional sheet name can be added (with ! char
to separate from the column name)
#       Examples:
#            - {C:} the value will be the value in column C on the current row
#            - {C: B:} the value will be the concatenation of values found in column C
and B of the current row
#            - {Deliveries} the value will be Deliveries
#            - {BJ: " - " BL:} the value will be the concatenation of value found in
column BJ,
#                string " - " and the value found in column BL fo the current row
#            - {OtherSheet!C:} the value will be the value in column C from the sheet
OtherSheet on the current row
#
#  - <condition> is a list of conditions. An empty condition is always true. A
condition is a column name followed by colon,
#              optionally followed by a perl regexp. Optional sheet name can be added
(with ! char to separate from the column name)
#       Examples:
#       - {B:} the value in column B must be empty on the current row
#       - {B:.+} the value in column B can not be empty on the current row
#       - {B:R_.+} the value in column B is a word starting by R_ on the current row
#       - {A: B:.+ C:R_.+} the value in column A must be empty and the value in column
B must contain something and
#           the column C contains a word starting with R_  on the current row
#       - {OtherSheet!B:.+} the value in column B from sheet OtherSheet on the current
row can not be empty.

# #############
# # ARTEFACTS #
# #############
# The variable is a list of artefact hierarchy specification:
# {ArtefactHierarchySpec1 ArtefactHierarchySpec2 ... ArtefactHierarchySpecN}
# where each ArtefactHierarchySpecx is a list of ArtefactSpec
#
# An ArtefactSpec is a list of items, each item being:
# {<(sheetName!)?artefactType> <conditions> <name> <parentType>? <parentName>?}
# where:
```

```
#    - <(sheetName!)?artefactType>: allows specifying the type. Optional sheetName can
be added (with ! char to separate from the type) to limit
#                                  the artefact search in one specific sheet. When
Sheets are given with regexp, the same regexp has to be used
#                                  for the sheetName.
#                                  If the type is followed by a question mark (?),
this level of artefact is optional.
#                                  If the type is followed by a plus char (+), this
level is repeatable on the next row
#    - <condition>: see COMMON DEFINITIONS
#    - <value>: the name of the artefact to build, see COMMON DEFINITIONS
#
#   - <parentType>: This element is optional. When present, it means that the current
element will be attached to a parent having this type
#   - <parentValue>: This is a list like <value> to build the name of the artefact of
type <parentType>. If such artefact is not found,
#                  the current artefact does not match
#
# Note: to add metrics at application level, specify an APPLICATION artefact which
will match only one line:
#       e.g. {APPLICATION {A:.+} {}} will recognize as application the line having
column A not empty.
set ArtefactsSpecs {
    {
        {DELIVERY {} {Deliveries}}
        {RELEASE {E:.+} {E:}}
        {SPRINT {O:SW_Software} {Q:}}
    }
    {
        {DELIVERY {} {Deliveries}}
        {RELEASE {O:SY_System} {Q:}}
    }
    {
        {WP {BL:.+ AF:.+} {BJ: " - " BL:} SPRINT {AF:}}
        {ChangeNotes!TASK {D:(added|changed|unchanged) T:imes} {W: AD:}}
    }
    {
        {WP {} {{Unplanned imes}} SPRINT {AF:}}
        {TASK {BL: D:(added|changed|unchanged) T:imes W:.+} {W: AD:}}
    }
}

# ###########
# # METRICS #
# ###########
# Specification of metrics to be retreived
# This is a list where each element is:
#  {<artefactTypeList> <metricId> <condition> <value> <format>}
# Where:
#     - <artefactTypeList>: the list of artefact types for which the metric has to be
used
```

```
#                       each element of the list is (sheetName!)?artefactType where
sheetName is used
#                       to restrict search to only one sheet. sheetName is optional.
#     - <metricId>: the name of the MeasureId to be injected into Squore, as defined
in your analysis model
#     - <confition>: see COMMON DEFINITIONS above. This is the condition for the
metric to be generated.
#     - <value> : see COMMON DEFINITIONS above. This is the value for the metric (can
be built from multi column)
#     - <format> : optional, defaults to NUMBER
#                Possible format are:
#                           * DATE_FR, DATE_EN for date stored as string
#                           * DATE for cell formatted as date
#                           * NUMBER_FR, NUMBER_EN for number stored as string
#                           * NUMBER for cell formatted as number
#                           * LINES for counting the number of text lines in a cell
#     - <formatPattern> : optional
#                Only used by the LINES format.
#                This is a pattern (can contain perl regexp) used to filter lines to
count
set MetricsSpecs {
    {{RELEASE SPRINT} TIMESTAMP {} {A:} DATE_EN}
    {{RELEASE SPRINT} DATE_ACTUAL_RELEASE {} {S:} DATE_EN}
    {{RELEASE SPRINT} DATE_FINISH {} {T:} DATE_EN}
    {{RELEASE SPRINT} DELIVERY_STATUS {} {U:}}
    {{WP} WP_STATUS {} {BO:}}
    {{ChangeNotes!TASK} IS_UNPLAN {} {BL:}}
    {{TASK WP} DATE_LABEL {} {BP:} DATE_EN}
    {{TASK WP} DATE_INTEG_PLAN {} {BD:} DATE_EN}
    {{TASK} TASK_STATUS {} {AE:}}
    {{TASK} TASK_TYPE {} {AB:}}
}

# ############
# # FINDINGS #
# ############
# This is a list where each element is:
#  {<artefactTypeList> <findingId> <condition> <value> <localisation>}
# Where:
#     - <artefactTypeList>: the list of artefact type for which the metric has to be
used
#                       each element of the list is (sheetName!)?artefactType where
sheetName is used
#                       to restrict search to only one sheet. sheetName is optional.
#     - <findingId>: the name of the FindingId to be injected into Squore, as defined
in your analysis model
#     - <confition>: see COMMON DEFINITIONS above. This is the condition for the
finding to be triggered.
#     - <value>: see COMMON DEFINITIONS above. This is the value for the message of
the finding (can be built from multi column)
#     - <localisation>: this a <value> representing the localisation of the finding
```

```
(free text)
set FindingsSpecs {
    {{WP} {BAD_WP} {BL:.+ AF:.+} {{This WP is not in a correct state } AF:.+} {A:}}
}

# ########################
# # TEXTUAL INFORMATION #
# ########################
# This is a list where each element is:
#  {<artefactTypeList> <infoId> <condition> <value>}
# Where:
#     - <artefactTypeList> the list of artefact types for which the info has to be used
#                        each element of the list is (sheetName!)?artefactType where sheetName is used
#                        to restrict search to only one sheet. sheetName is optional.
#     - <infoId> : is the name of the Information to be attached to the artefact, as defined in your analysis model
#     - <confition> : see COMMON DEFINITIONS above. This is the condition for the info to be generated.
#     - <value> : see COMMON DEFINITIONS above. This is the value for the info (can be built from multi column)
set InfosSpecs {
    {{TASK} ASSIGN_TO {} {XB:}}
}

# #########################
# # LABEL TRANSFORMATION #
# #########################
# This is a list value specification for MeasureId or InfoId:
#    <MeasureId|InfoId> { {<LABEL1> <value1>} ... {<LABELn> <valuen>}}
# Where:
#    - <MeasureId|InfoId> : is either a MeasureId, an InfoId, or * if it is available for every measureid/infoid
#    - <LABELx> : is the label to macth (can contain perl regexp)
#    - <valuex> : is the value to replace the label by, it has to match the correct format for the metrics (no format for infoid)
#
# Note: only metrics which are labels in the excel file or information which need to be rewriten, need to be described here.
set Label2ValueSpec {
    {
        STATUS {
            {OPENED 0}
            {ANALYZED 1}
            {CLOSED 2}
            {.* -1}
        }
    }
    {
        * {
            {FATAL 0}
            {ERROR 1}
            {WARNING 2}
            {{LEVEL:\s*0} 1}
            {{LEVEL:\s*1} 2}
            {{LEVEL:\s*[2-9]+} 3}
        }
    }
}

Note that a sample Excel file with its associated config.tcl is available in
$SQUORE_HOME/addons/tools/ExcelMetrics in order to further explain available
configuration options.
```

Appendix B: Squore XML Schemas

input-data-2.xsd

Download input-data-2.xsd

form.xsd

Download form.xsd

properties-1.2.xsd

Download properties-1.2.xsd

config-1.3.xsd

Download config-1.3.xsd

analysis.xsd

Download analysis.xsd

decision.xsd

Download decision.xsd

description.xsd

Download description.xsd

exports.xsd

Download exports.xsd

highlights.xsd

Download highlights.xsd

properties.xsd

Download properties.xsd

tutorials.xsd

Download tutorials.xsd

wizards.xsd

Download wizards.xsd

Appendix C: Milestones Tutorial

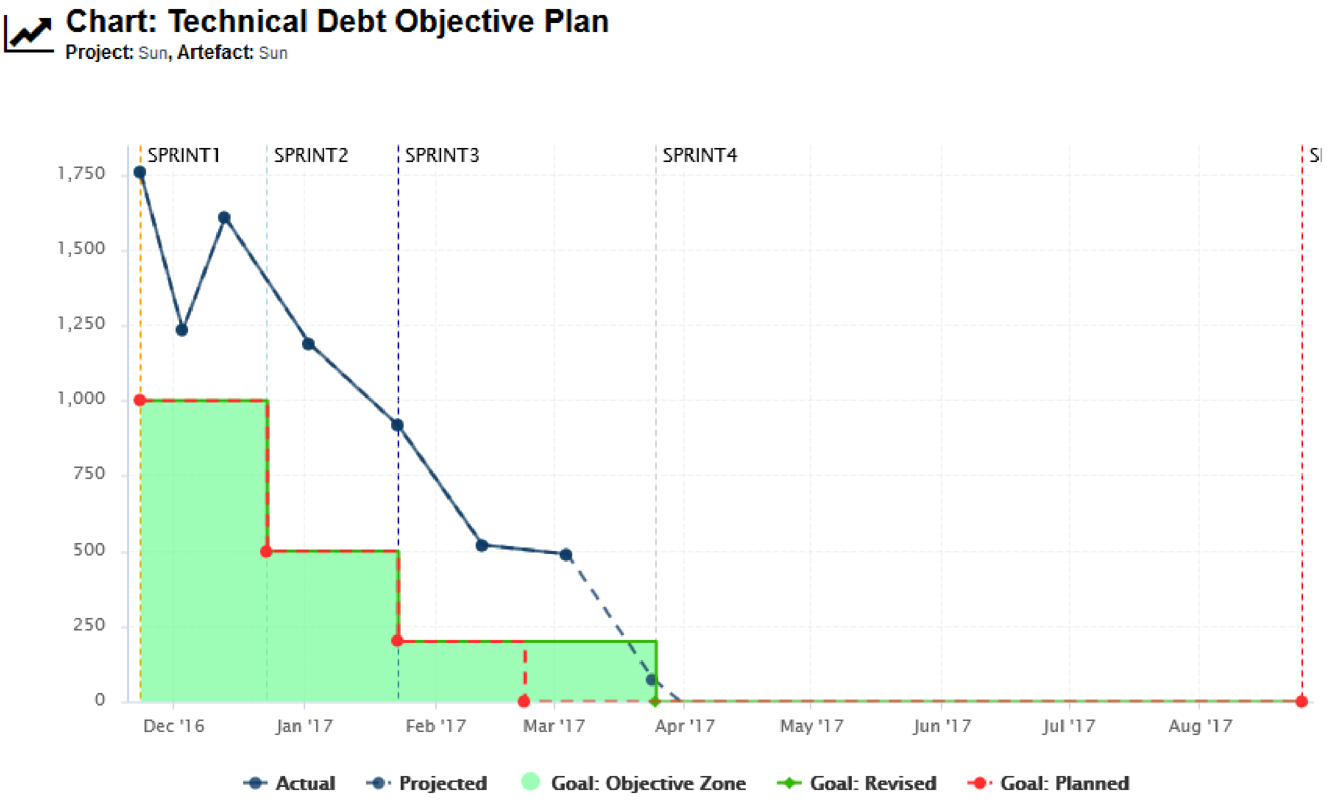
Concept

With the introduction milestones in your project, Squore offers new ways to measure your objectives and detect deviations from your goals early. Milestones are a series of goals for specific metrics at certain dates in the life of your project and add the following to your process management:

- You are alerted early if your current performance shows that you will not meet your goals and can react before it is too late
- You keep track of your various goals and communicate any change to the rest of your team
- You can reflect on a project's history and learn from it

This example focuses on a project that is slipping, and shows how the team reacts along the course of the development process. Our team is tracking several objectives around issue management, technical debt and self-descriptiveness over the lifetime of the project, which includes milestones for 5 sprints labelled SPRINT1 to SPRINT5.

Here is where they stand in the fourth sprint and try to assess whether they will meet their Technical Debt objective for the release date at the end of Sprint 5:



The chart shows the following information:

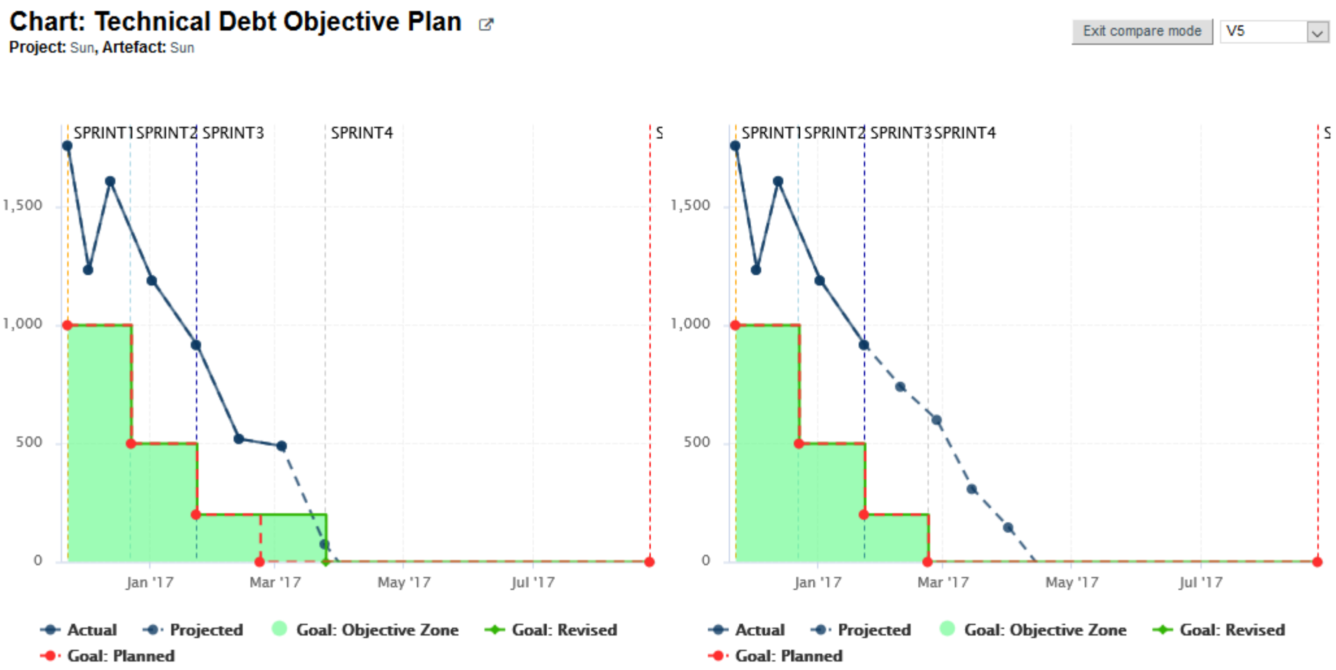
- Vertical dotted lines (markers) on the x-axis for each milestone in the project at the predefined date
- A solid dark-blue line showing the technical debt value for each version of the project so far
- A dotted dark-blue line showing estimations for technical debt for future versions absed on the progress so far
- A dotted red line showing the goals set at the beginning of the project for each sprint for the technical debt metric
- A solid green line showing the goals as they were revised as time went on (the date for Sprint

4 was moved back).

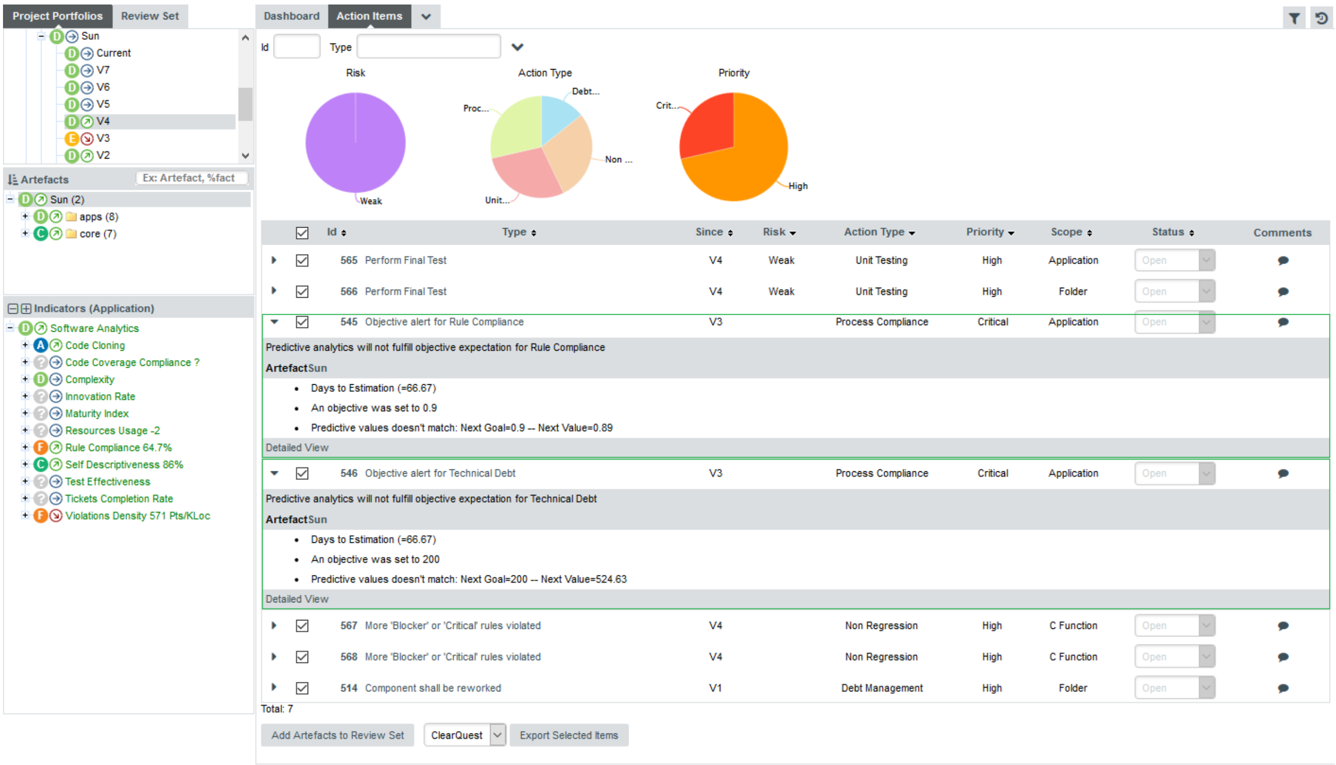
- A turquoise area highlighting the acceptable range for the tehnical debt for each sprint, making it clear that the technical debt has never been under control so far, but that projections show that the goal should be met by the end of Sprint 3

In order to understand why changes were made to the goals, let's go back to V4 and look at the Technical Debt Objective Plan again. The end of Sprint4 still has its original date, and projections aready show that technical debt will not be under control by the end of the sprint.

Our chart is configured to show the projected value for the next 5 analyses (based on the rate of previous analyses), and the firth projection meeting the expectations for SPRINT4 appear well after the original date for SPRINT4.



The team knew this at the time: a **Objective alert for Technical Debt** action item was opened on as early as V3 to inform them that the current performance could cause problems for their objective set 50 days later.



☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
☑	565	Perform Final Test	V4	Weak	Unit Testing	High	Application	Open	
☑	566	Perform Final Test	V4	Weak	Unit Testing	High	Folder	Open	
☑	545	Objective alert for Rule Compliance	V3		Process Compliance	Critical	Application	Open	

Predictive analytics will not fulfill objective expectation for Rule Compliance

Artefact Sun

- Days to Estimation (=66.67)
- An objective was set to 0.9
- Predictive values doesn't match: Next Goal=0.9 -- Next Value=0.89

Detailed View

☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
☑	546	Objective alert for Technical Debt	V3		Process Compliance	Critical	Application	Open	

Predictive analytics will not fulfill objective expectation for Technical Debt

Artefact Sun

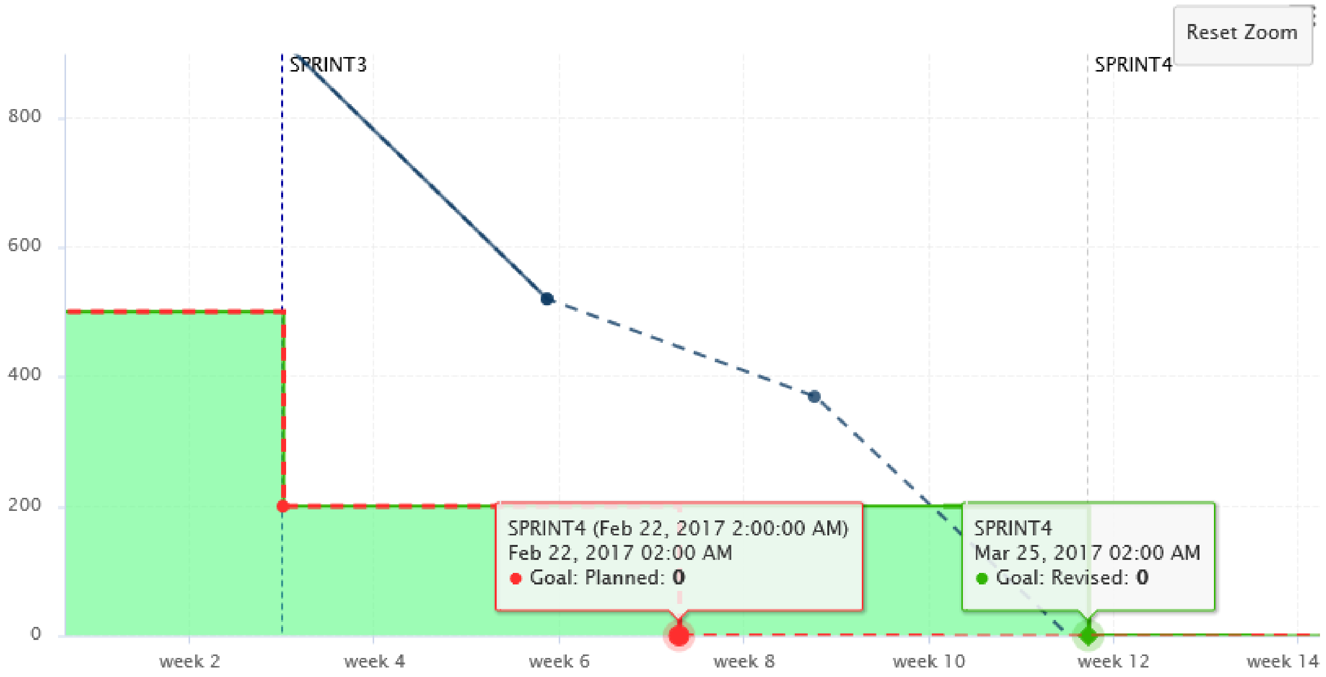
- Days to Estimation (=66.67)
- An objective was set to 200
- Predictive values doesn't match: Next Goal=200 -- Next Value=524.63

Detailed View

☑	Id	Type	Since	Risk	Action Type	Priority	Scope	Status	Comments
☑	567	More 'Blocker' or 'Critical' rules violated	V4		Non Regression	High	C Function	Open	
☑	568	More 'Blocker' or 'Critical' rules violated	V4		Non Regression	High	C Function	Open	
☑	514	Component shall be reworked	V1		Debt Management	High	Folder	Open	

Total: 7

After a team meeting, it is decided that the best course of action is to keep the goal for the SPRINT4 milestone, but move its date back by one month. The next analysis confirms this on the Technical Debt Objective plan chart, where you see the first deviation between the planned goal (red) and the actual goal (green). The progress objective will be now met:



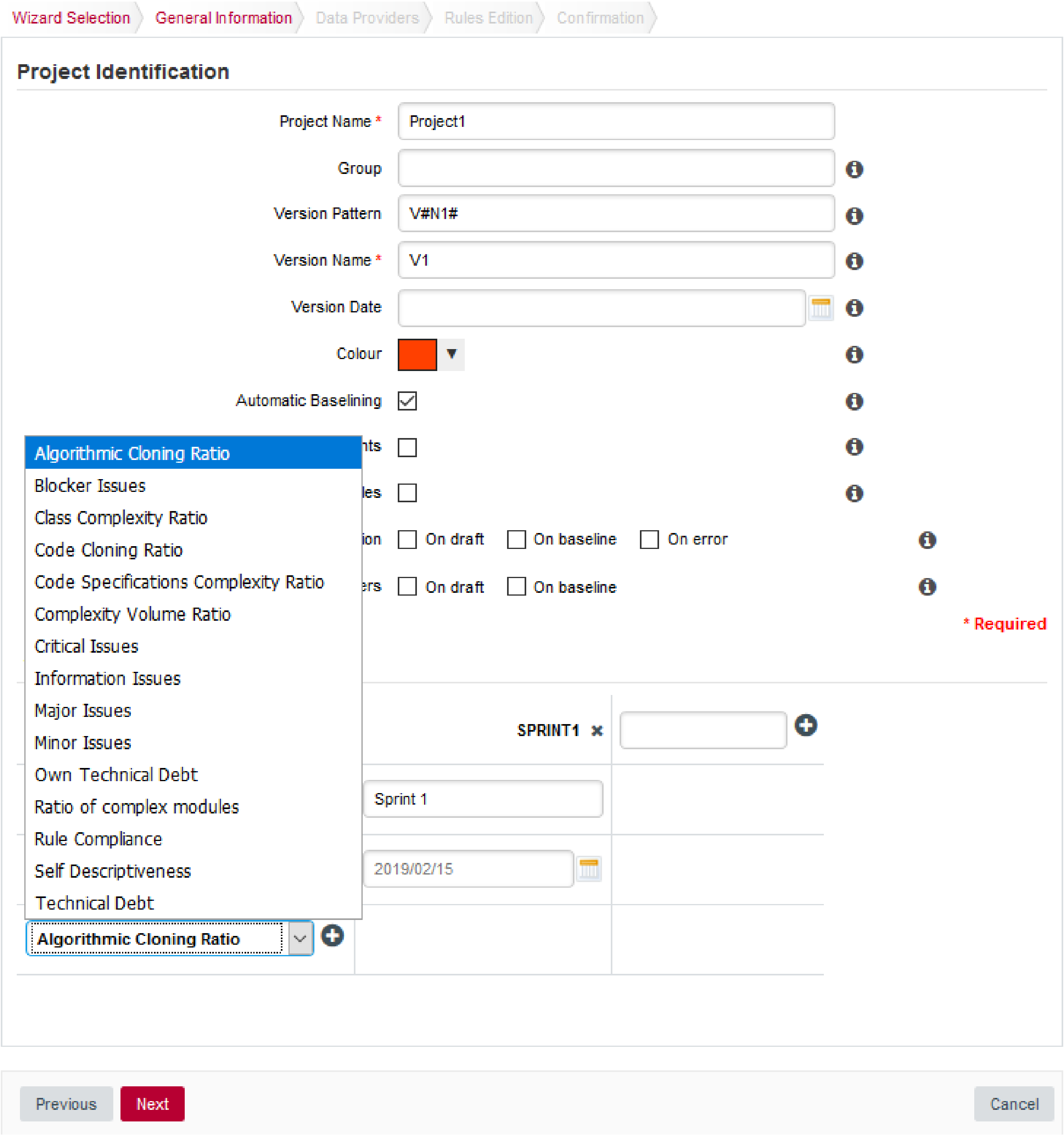
How it works

In order to add support for milestones to your model, configure your wizard to allow users to create milestones and goals:

```
<Bundle xmlns:xi="http://www.w3.org/2001/XInclude">
    <wizard wizardId="ANALYTICS" versionPattern="v#N1#" img=
"../../Shared/Wizards/squore_logo.png" hideRulesEdition="FALSE">
        <milestones canCreateMilestone="TRUE" canCreateGoal="TRUE">
            <goals displayableFamilies="ANALYTICS_GOALS" />
        </milestones>
    </wizard>
</Bundle>
```

The **`milestones`** element allows users to create milestones in the project wizard (`canCreateMilestone="TRUE"`) and also set goals (`canCreateGoal="TRUE"`). The goals can be set for metrics of the GOALS family only in this example (`displayableFamilies="ANALYTICS_GOALS"`).

The result in the web UI is the following:



Project Identification

Project Name *	Project1
Group	
Version Pattern	V#N1#
Version Name *	V1
Version Date	
Colour	🟥 ▼
Automatic Baselining	☑
…nts	□
…les	□
…ion	□ On draft □ On baseline □ On error
…ers	□ On draft □ On baseline

* Required

SPRINT1 ×

Sprint 1

2019/02/15

Algorithmic Cloning Ratio

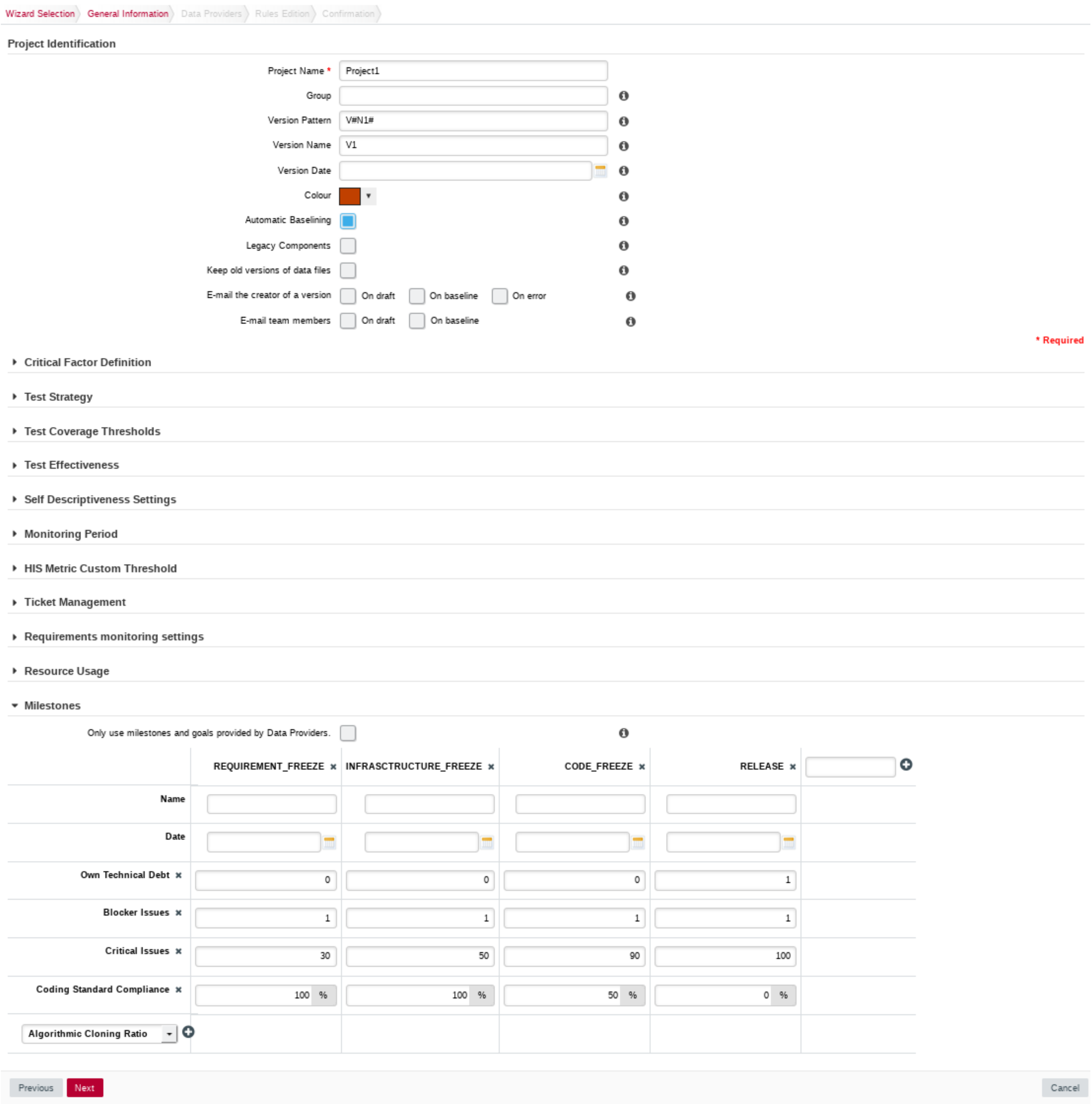
A wizard allowing users to create milestones freely during an analysis

When creating a new project, a user decides to create a **Sprint 1** milestone with one objective of **500** for the **Technical Debt** indicator. Other goals can be set, for the other metrics in the project that belong to the **ANALYTICS_GOALS** family listed in the dropdown list at the bottom of the table.

If you have company-wide milestones and objectives that need to be set for every project created with the wizard, you can specify the goals directly. Milestones can also be marked as mandatory or optional:

```
<Bundle xmlns:xi="http://www.w3.org/2001/XInclude">
    <wizard wizardId="ANALYTICS_WITH_MILESTONES" versionPattern="v#N1#" img=
"../../Shared/Wizards/squore_logo.png" hideRulesEdition="FALSE">
        <milestones canCreateMilestone="TRUE" canCreateGoal="TRUE">
            <goals displayableFamilies="GOALS">
                <goal measureId="TECH_DEBT" mandatory="TRUE" highestIsBest="FALSE" />
                <goal measureId="ISSUE_BLOCKER" mandatory="TRUE" highestIsBest="TRUE"
/>
                <goal measureId="ISSUE_CRITICAL" mandatory="TRUE" highestIsBest="TRUE"
/>
                <goal measureId="ROKR_SUBSET" mandatory="TRUE" highestIsBest="FALSE"
/>
            </goals>
            <milestone id="REQUIREMENT_FREEZE" mandatory="TRUE">
                <defaultGoal measureId="TECH_DEBT" value="0" />
                <defaultGoal measureId="ISSUE_BLOCKER" value="1" />
                <defaultGoal measureId="ISSUE_CRITICAL" value="30" />
                <defaultGoal measureId="ROKR_SUBSET" value="1" />
            </milestone>
            <milestone id="INFRASTRUCTURE_FREEZE" mandatory="TRUE">
                <defaultGoal measureId="TECH_DEBT" value="0" />
                <defaultGoal measureId="ISSUE_BLOCKER" value="1" />
                <defaultGoal measureId="ISSUE_CRITICAL" value="50" />
                <defaultGoal measureId="ROKR_SUBSET" value="1" />
            </milestone>
            <milestone id="CODE_FREEZE" mandatory="TRUE">
                <defaultGoal measureId="TECH_DEBT" value="0" />
                <defaultGoal measureId="ISSUE_BLOCKER" value="1" />
                <defaultGoal measureId="ISSUE_CRITICAL" value="90" />
                <defaultGoal measureId="ROKR_SUBSET" value="0.5" />
            </milestone>
            <milestone id="BETA_RELEASE" mandatory="FALSE">
                <defaultGoal measureId="TECH_DEBT" value="1" />
                <defaultGoal measureId="ISSUE_BLOCKER" value="1" />
                <defaultGoal measureId="ISSUE_CRITICAL" value="95" />
                <defaultGoal measureId="ROKR_SUBSET" value="0.3" />
            </milestone>
            <milestone id="RELEASE" mandatory="TRUE">
                <defaultGoal measureId="TECH_DEBT" value="1" />
                <defaultGoal measureId="ISSUE_BLOCKER" value="1" />
                <defaultGoal measureId="ISSUE_CRITICAL" value="100" />
                <defaultGoal measureId="ROKR_SUBSET" value="0" />
            </milestone>
        </milestones>
    </wizard>
</Bundle>
```

When creating a new project, the predefined goals are filled in in the web interface, and you can still add a **Beta Release** milestone (using the default values specified in the wizard bundle) if needed by using the **\+** icon:



Wizard Selection › General Information › Data Providers › Rules Edition › Confirmation

Project Identification

Project Name *	Project1
Group	
Version Pattern	V#N1#
Version Name	V1
Version Date	
Colour	▼
Automatic Baselining	☑
Legacy Components	□
Keep old versions of data files	□
E-mail the creator of a version	□ On draft □ On baseline □ On error
E-mail team members	□ On draft □ On baseline

* Required

- Critical Factor Definition
- Test Strategy
- Test Coverage Thresholds
- Test Effectiveness
- Self Descriptiveness Settings
- Monitoring Period
- HIS Metric Custom Threshold
- Ticket Management
- Requirements monitoring settings
- Resource Usage
- Milestones

Only use milestones and goals provided by Data Providers. □

	REQUIREMENT_FREEZE ✖	INFRASCTRUCTURE_FREEZE ✖	CODE_FREEZE ✖	RELEASE ✖
Name				
Date				
Own Technical Debt ✖	0	0	0	1
Blocker Issues ✖	1	1	1	1
Critical Issues ✖	30	50	90	100
Coding Standard Compliance ✖	100 %	100 %	50 %	0 %
Algorithmic Cloning Ratio ▾				

A project wizard with preconfigured milestones and goals

Command Line

If you create projects using the command line interface, you can specify settings for your milestones with the -M parameter:

```
-M "id=BETA_RELEASE,date=2017/05/31,ISSUE_CRITICAL=95"
```

or with a project config file:

```
<SquoreProjectSettings>
    <Wizard>
        <Milestones>
            <Milestone id="BETA_RELEASE" date="2017-05-31">
                <Goal id="ISSUE_CRITICAL" value="95" />
            </Milestone>
        </Milestones>
    </Wizard>
</SquoreProjectSettings>
```

Milestone Function

In your analysis model, new functions are available to work with milestones and projections:

- **HAS_MILESTONE([milestoneId or keyword] [, date])**

  This function can be used in charts that support computations

  Checks if a milestone with the specified milestoneId exists in the project. The function returns 0 if no milestone is found, 1 if a milestone is found.

  Find if we are at the last milestone of the project:

  ```
  IS_LAST_MILESTONE=IF(HAS_MILESTONE(),0,1)
  ```

- **DATE_MILESTONE([milestoneId or keyword] [, date])**

  This function can be used in charts that support computations

  Returns the date associated to a milestone.

  Find if the date for the milestone BETA_RELEASE has been modified between June 2015 and now:

  ```
  DATE_HAS_SLIPPED=(DATE_MILESTONE(BETA_RELEASE)-DATE_MILESTONE(BETA_RELEASE,
  DATE(2015,06,01))) != 0
  ```

  Compute the date difference between the previous and next milestones:

  ```
  MILESTONE_DURATION=DATE_MILESTONE(NEXT) - DATE_MILESTONE(PREVIOUS)
  ```

  Find the date slip for the next milestone between now and the previous anlaysis:

  ```
  DATE_SLIP=DATE_MILESTONE(NEXT) - DATE_MILESTONE(NEXT, VERSION_DATE(PREVIOUS))
  ```

  Find the amount of time left until the next milestone:

  ```
  DEADLINE=DATE_MILESTONE(NEXT) - VERSION_DATE()
  ```

- **GOAL(measureId [, milestoneId or keyword] [, date])**

  This function can be used in charts that support computations

Returns the goal for a metric at the specified milestone.

Find the goal for requirement stability set for the milestone PROTOTYPE as of June 2016:

```
REQ_STABILITY_GOAL=GOAL(REQ_STABILITY, PROTOTYPE, DATE(2016,06,01))
```

Find the delta between the goal for TEST between the previous and next milestones:

```
DELTA=GOAL(TEST) - GOAL(TEST, PREVIOUS)
```

Find the delta between the goal for TEST for the next milestone set for the previous analysis and now:

```
DELTA=GOAL(TEST) - GOAL(TEST, NEXT, VERSION_DATE(PREVIOUS))
```

Find the delta between the current value of TEST and the goal for TEST at the next milestone:

```
DELTA=GOAL(TEST) - TEST
```

> You can use keywords instead of using a milestone ID. You can retrieve information about the next, previous, first or last milestones in the project by using:
>
> - `NEXT`
> - `NEXT+STEP` where STEP is a number indicating how many milestones to jump ahead
> - `PREVIOUS`
> - `PREVIOUS-STEP` where STEP is a number indicating how many milestones to jump backward
> - `FIRST`
> - `LAST`

Consut the Configuration Guide for more details.

Milestones in Charts

On your charts, you are now able to:

- Display the goals defined for each milestone in your project
- Display the changes made to the goals defined for each milestone
- Display the date changes for your milestones
- Show markers for milestone dates and goals

You can also compute metrics with functions like **LEAST_SQUARE_FIT()**, which lets you calculate projections. This is how the Task Completion chart used in this example was created. You can find its full definition below:

```
<chart id="OBJECTIVE_TECH_DEBT" type="TE" byTime="true" dateFormat="MM/yy" yMin="0"
displayOnlyIf="HAD_GOAL_TECH_DEBT">
    <dataset renderer="LINE">
        <measure dataBounds="[0;[" color="#0B3861" stroke="SOLID" shape="CIRCLE"
alpha="200" label="Actual">TECH_DEBT</measure>
        <measure color="#0B3861" stroke="DOTTED" shape="CIRCLE" alpha="200" label=
"Projected">TECH_DEBT
            <forecast>
                <estimatedVersion timeValue="CUR_BUILD_DATE+1*DELTA_MEAN"/>
                <estimatedVersion timeValue="CUR_BUILD_DATE+2*DELTA_MEAN"/>
                <estimatedVersion timeValue="CUR_BUILD_DATE+3*DELTA_MEAN"/>
                <estimatedVersion timeValue="CUR_BUILD_DATE+4*DELTA_MEAN"/>
                <estimatedVersion timeValue="CUR_BUILD_DATE+5*DELTA_MEAN"/>
            </forecast>
        </measure>
    </dataset>

    <dataset renderer="AREA_STEP">
        <goal dataBounds="[0;[" color="88,250,130" stroke="SOLID" shape="DIAMOND"
alpha="150" label="Objective Zone">TECH_DEBT</goal>
    </dataset>

    <dataset renderer="STEP">
        <goal dataBounds="[0;[" color="#31B404" stroke="SOLID" shape="DIAMOND" alpha=
"255" label="Revised">TECH_DEBT</goal>
        <goal dataBounds="[0;[" color="#FE2E2E" stroke="DOTTED" shape="CIRCLE" alpha=
"255" label="Planned" versionDate="FIRST_BUILD_DATE">TECH_DEBT</goal>
    </dataset>

    <markers>
        <marker alpha="150" color="189,189,189" isVertical="false" endValue="0"/>
        <marker fromMilestones="true" alpha="150" isVertical="true" stroke="DOTTED" />
    </markers>
</chart>
```

The action items monitoring the project's progress also make use of the new **GOAL()** function and were defined as follows:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Bundle>
    <DecisionCriteria>
        (...)
        <DecisionCriterion dcId="AI_OBJECTIVE_IN_FUTURE_TECH_DEBT" categories=
"SCALE_PRIORITY.CRITICAL;SCALE_AI_TYPE.PROCESS_IMPROVEMENT" targetArtefactTypes=
"APPLICATION">
            <Triggers>
                <Trigger>
                    <Test expr="GOAL_ESTIMATED_TECH_DEBT-ESTIMATED_TECH_DEBT" bounds=
"];0[" descrId="GOAL_WILL_NOT_BE_REACHED" p0="#{MEASURE.GOAL_ESTIMATED_TECH_DEBT}" p1
="#{MEASURE.ESTIMATED_TECH_DEBT}" />
                    <Test expr="GOAL_TECH_DEBT" bounds="[1;1]" descrId=
"GOAL_IS_ACTIVATED" p0="#{MEASURE.GOAL_ESTIMATED_TECH_DEBT}" />
                    <Test expr="DAY_TO_ESTIMATION" bounds="];[" descrId=
"DAY_TO_ESTIMATION" p0="#{MEASURE.DAY_TO_ESTIMATION}" />
                </Trigger>
            </Triggers>
        </DecisionCriterion>
    </DecisionCriteria>
</Bundle>
```

Find out More

Check out the Getting Started Guide and the Configuration Guide to learn everything about milestones in Squore.

Index