Consulting
Services

TIBCO SonarQube BusinessWorks 6.x Plugin Guidelines

Project Name	SonarQube BusinessWorks 6.x Plugin
Release	1.0.0
Date	21/12/2015
Primary Author	Kapil Shivarkar/TIBCO
Document Owner	Kapil Shivarkar/TIBCO
Client	
Document Location	
Purpose	This document is the guide for SonarQube BusinessWorks 6.x Plugin.

TIBCO®
The Power of Now®

http://www.tibco.com

Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304
Tel: +1 650-846-1000
Toll Free: 1 800-420-8450
Fax: +1 650-846-1005

LICENSE INFORMATION

Revision History

Version	Date	Author	Comments
1.0.0	21/12/2015	Kapil Shivarkar	First release

Approvals

This document requires the following approvals. Signed approval forms are filed in the project files.

Name	Signature	Title	Company	Date of Issue	Version

Distribution

This document has been distributed to:

Name	Title	Company	Date of Issue	Version

Table of Contents

A Introduction 6

B Architecture 7

C Continuous Integration 8

D Definitions 9

D.1 Commons 9
D.1.1 Quality management 9
D.1.2 Quality assurance 9
D.2 SonarQube 9
D.2.1 Basis 9
D.2.2 Analysis modes 10
D.2.3 Stakeholders/components 10
D.2.4 Quality 11
D.2.5 Web interface 11
D.3 Analysis pipeline 12
D.4 Plugin extensions 12

E Features 14

E.1 Code analysis 14
E.1.1 Languages 14
E.1.2 Profiles 14
E.1.3 Sensors 14
E.2 UI extensions 14
E.2.1 BusinessWorks Metrics Widget 14

F Rules for BusinessWorks 6.x analysis 16

F.1 BusinessWorks 6.x 16
F.1.1 Deadlock Detection Check 16
F.1.2 Activities in Critical Section Check 16
F.1.3 Checkpoint Activity inside Critical Section Group Check 16
F.1.4 Multiple Transitions Check 17
F.1.5 Log Only in Subprocess Check 17
F.1.6 Checkpoint inside Transaction Group Check 17
F.1.7 Checkpoint after HTTP Activities Check 18
F.1.8 Checkpoint after REST Webservice Call Check 18
F.1.9 Checkpoint after JDBC Query Activity Check 18
F.1.10 Choice Condition with No Otherwise Check 19
F.1.11 Transition Labels Check 19
F.1.12 JDBC WildCard Check 19
F.1.13 JDBC HardCoded Check 19
F.1.14 JMS HardCoded Check 20
F.1.15 For-Each Group Check 20
F.1.16 For-Each Mapping Check 20
F.1.17 JMS Acknowledgement Mode Check 21
F.1.18 Data Availability to Inline SubProcess Check 21
F.1.19 Number of Activities Check 21

F.1.20 Number of Exposed Services Check ... 21
F.1.21 No Process Description Check ... 22

G Administration guide ... 23

G.1 Requirements ... 23
G.2 Installation ... 23
G.3 Configuration ... 23
G.3.1 General Settings ... 23
G.4 Customization ... 23
G.4.1 Quality Profiles ... 23
G.4.2 Rules ... 24
G.4.3 Quality Gates ... 26

H Developer Guide ... 27

H.1 Coding a new rule ... 27

I User guide ... 28

I.1 Execute an analysis ... 28
I.1.1 Introduction ... 28
I.1.2 Using Maven ... 28
I.1.2.1 Prerequisites ... 28
I.1.2.2 Configure your pom.xml ... 29
I.1.2.3 Run the analysis ... 30
I.1.3 Using SonarQube Runner ... 30
I.1.3.1 Prerequisites ... 30
I.1.3.2 Create project configuration file ... 30
I.1.3.3 Run the analysis ... 30
I.1.4 Using Jenkins ... 30
I.2 Customize a report ... 31
I.2.1 Introduction ... 31
I.2.2 Add the BusinessWorks Metrics Widget ... 31
I.3 BusinessWorks Process visualization ... 32
I.3.1 Issues Visualization ... 32

J Configuring SonarQube in TIBCO BusinessStudio ... 35

K Useful links ... 37

A Introduction

SonarQube is an open source platform for continuous inspection of code quality. SonarQube BusinessWorks 6 Plugin is a custom extension developed in order to manage TIBCO Business Works code quality and design best practices.

SonarQube covers the 7 axes of code quality:

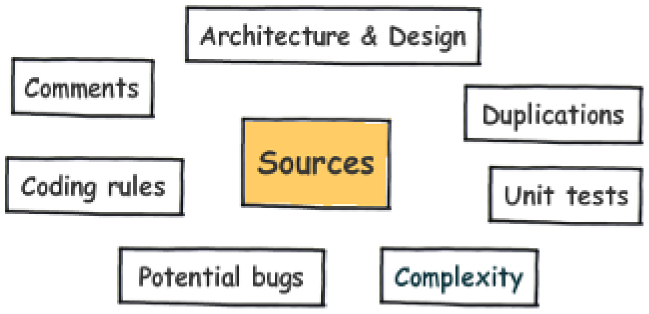


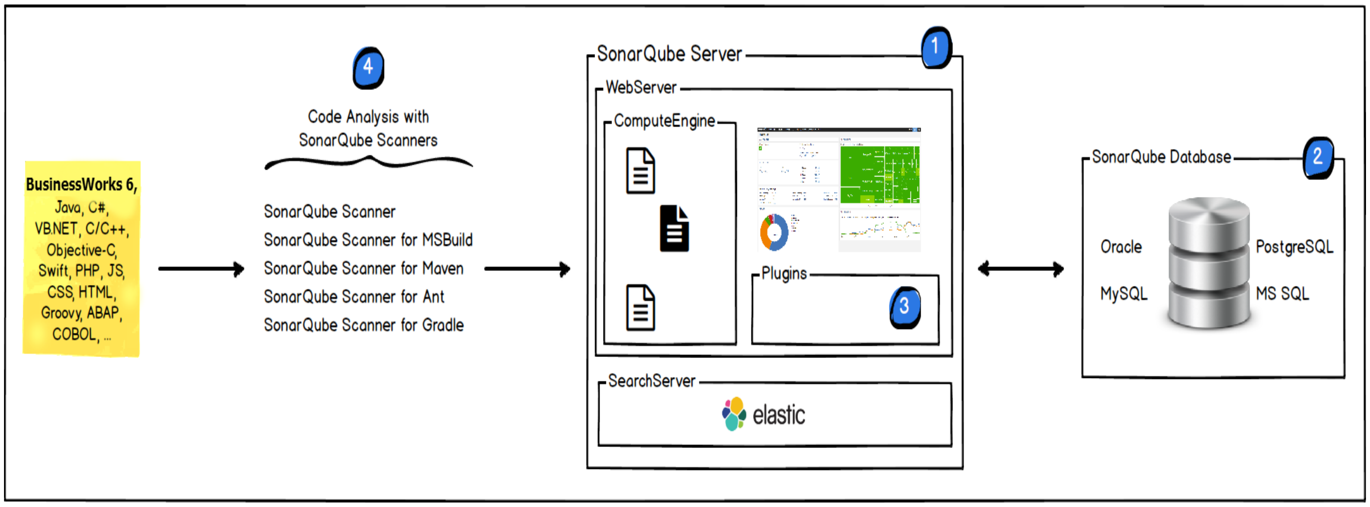
***Figure 1.** Seven axes of code quality in SonarQube*

SonarQube is a web-based application. Rules, alerts, thresholds, exclusions, settings… can be configured online. By leveraging its database, SonarQube not only allows to combine metrics altogether but also to mix them with historical measures.

B Architecture

The SonarQube Platform is made of 4 components:

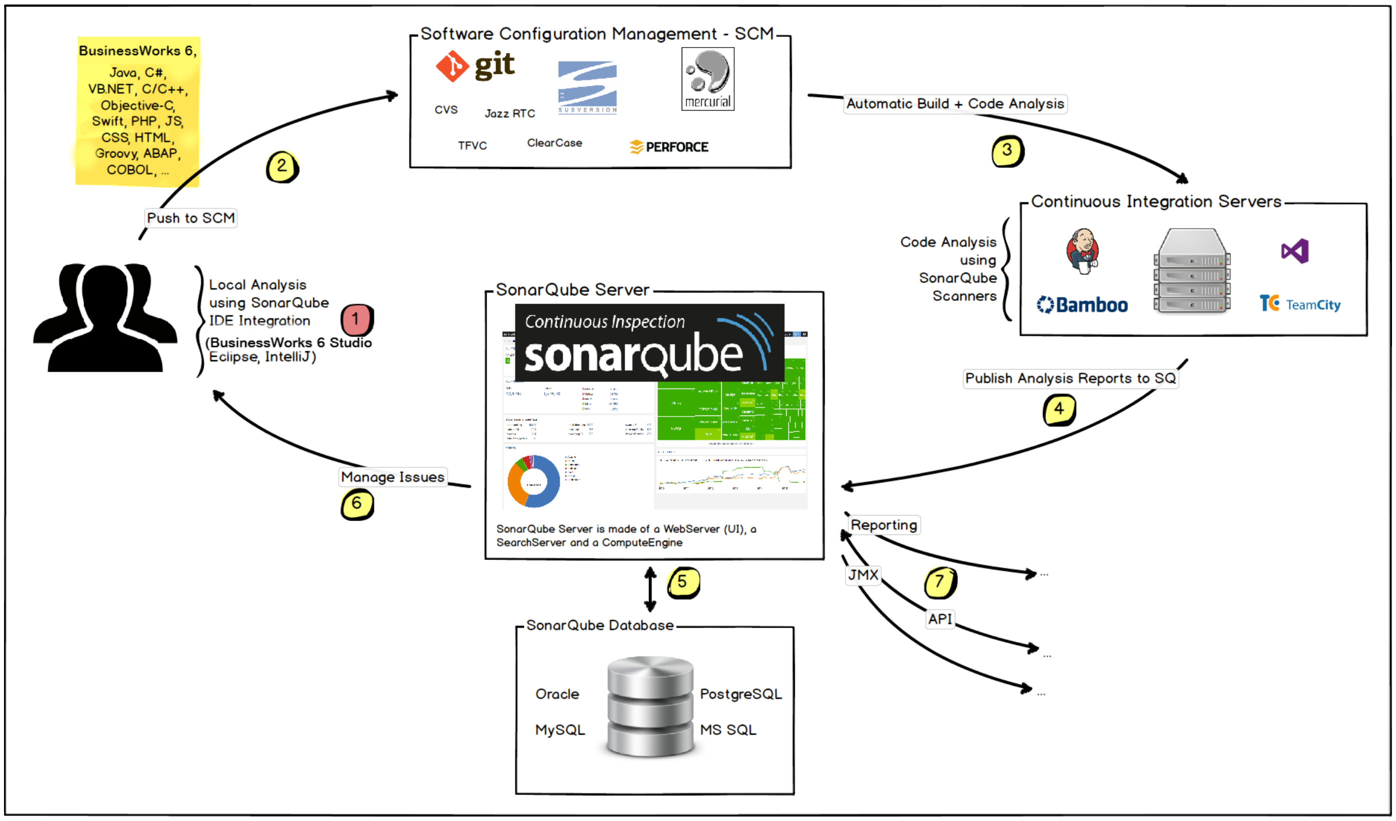
1. One **SonarQube Server** starting 2 main processes:
    a. **Web Server** for developers, managers to browse quality snapshots and configure the SonarQube instance
    b. **Search Server** based on Elasticsearch to back searches from the UI
2. One **SonarQube Database** to store:
    a. the configuration of the SonarQube instance (security, plugins settings, etc.)
    b. the quality snapshots of projects, views, etc.
3. Multiple **SonarQube Plugins** installed on the server. There are SonarQube plugins for languages (***BusinessWorks 6***, Java etc), SCM, integration, authentication, and governance plugins
4. One or more **SonarQube Scanners** running on your Build / Continuous Integration Servers to analyze projects


BusinessWorks 6, Java, C#, VB.NET, C/C++, Objective-C, Swift, PHP, JS, CSS, HTML, Groovy, ABAP, COBOL, ...
4
Code Analysis with SonarQube Scanners
SonarQube Scanner
SonarQube Scanner for MSBuild
SonarQube Scanner for Maven
SonarQube Scanner for Ant
SonarQube Scanner for Gradle
SonarQube Server
1
WebServer
ComputeEngine
Plugins
3
SearchServer
elastic
SonarQube Database
2
Oracle
PostgreSQL
MySQL
MS SQL

C Continuous Integration

The following schema shows how SonarQube integrates with other ALM tools and where the various components of SonarQube are used.

1. Developers code in their IDEs (**BusinessWorks 6 Studio**) and use SonarQube or SonarLint plugin to run local analysis.
2. Developers push their code into their favourite SCM : git, SVN, TFVC, ...
3. The Continuous Integration Server triggers an automatic build, and the execution of the SonarQube Scanner required to run the SonarQube analysis.
4. The analysis report is sent to the SonarQube Server for processing.
5. SonarQube Server processes and stores the analysis report results in the SonarQube Database, and displays the results in the UI.
6. Developers review, comment, challenge their Issues to manage and reduce their Technical Debt through the SonarQube UI.
7. Managers receive Reports from the analysis.
   Ops use APIs to automate configuration and extract data from SonarQube.
   Ops use JMX to monitor SonarQube Server.


BusinessWorks 6,
Java, C#, VB.NET, C/C++, Objective-C, Swift, PHP, JS, CSS, HTML, Groovy, ABAP, COBOL, ...
Software Configuration Management - SCM
git
CVS
Jazz RTC
SUBVERSION
mercurial
TFVC
ClearCase
PERFORCE
Automatic Build + Code Analysis
3
2
Push to SCM
Continuous Integration Servers
Code Analysis using SonarQube Scanners
Bamboo
TeamCity
Local Analysis using SonarQube IDE Integration
1
(BusinessWorks 6 Studio Eclipse, IntelliJ)
SonarQube Server
Continuous Inspection
sonarqube
Publish Analysis Reports to SQ
4
Manage Issues
6
SonarQube Server is made of a WebServer (UI), a SearchServer and a ComputeEngine
Reporting
JMX
7
API
...
5
SonarQube Database
Oracle
PostgreSQL
MySQL
MS SQL

D Definitions

D.1 Commons

D.1.1 Quality management

Quality management ensures that an organization, product or service is consistent. It has four main components: quality planning, quality control, quality assurance and quality improvement.

Quality management is focused not only on product and service quality, but also the means to achieve it. Quality management therefore uses quality assurance and control of processes as well as products to achieve more consistent quality.

D.1.2 Quality assurance

Quality Assurance is the planned or systematic actions necessary to provide enough confidence that a product or service will satisfy the given requirements.

D.2 SonarQube

D.2.1 Basis

Rules

In SonarQube, plugins contribute rules which are executed on source code and which generate issues. The Rules page is the entry point where you can discover all the existing rules or create new ones based on provided templates. Three types of rules exist in SonarQube:

- Standard Rules : basic rules that can be activated and define the issue severity
- Custom Rules : they are considered like any other rule but can be edited or deleted at any moment of time
- Rule Templates : they can only be used to create custom rules and cannot be activated because they are just empty templates with empty parameters

Issues

While running an analysis, SonarQube raises an issue every time a piece of code breaks a coding rule. The set of coding rules is defined through the quality profile associated with the project. Developers can also manually raise issues that cannot be detected by SonarQube (examples: the implementation of the method does not comply to the functional requirements, the javadoc of the method does not match its implementation, etc.).
Each issue has one of five severities:

- BLOCKER: Bug with a high probability to impact the behavior of the application in production: memory leak, unclosed JDBC connection, deadlocks etc.... The code MUST be immediately fixed.

- CRITICAL: Either a bug with a low probability to impact the behavior of the application in production or an issue which represents a security flaw: empty catch block, SQL injection, etc... The code MUST be immediately reviewed.

- MAJOR: Quality flaw which can highly impact the developer productivity: uncovered piece of code, duplicated blocks, unused parameters, etc...

- MINOR: Quality flaw which can slightly impact the developer productivity: lines should not be too long or "switch" statements should have at least 3 cases, etc...

- INFO: Neither a bug nor a quality flaw, just a finding.

D.2.2 Analysis modes

Concept	Definition
Analysis	Standard way to analyze the source code. The source code is analyzed and measures and issues are pushed to the SonarQube database. The results of the analysis can be browsed through the web interface.
Incremental	Same as Preview mode but only new or modified files (compared to the latest version available on the remote server) are analyzed. This is the default mode of the SonarQube Eclipse plugin and the SonarQube IntelliJ plugin.
Preview	The source code is analyzed but the measures and issues are not pushed to the SonarQube database. Therefore, they cannot be browsed through the web interface. This mode can be used with the Issues Report plugin, which generates an HTML issues report to local file.

D.2.3 Stakeholders/components

Concept	Definition
Analyzer	A client application that analyzes the source code to compute snapshots.
Database	Stores: • configuration

	• snapshots
Server	Web interface that is used to browse snapshot data and make configuration changes

D.2.4 Quality

Concept	Definition
Check	Check = Coding Rule.
Coding Rule	A good coding practice. Not complying with coding rules leads to quality flaws and creation of issues in SonarQube. Coding rules can check quality on files, unit tests or packages.
Component	A piece of software (project, module/package, file, resource, process, etc…) or a view or a developer.
Issue	When a component does not comply with a coding rule, an issue is logged (was violation prior to SonarQube 3.6) on the snapshot. An issue can be logged on a source file or a unit test file.
Measure	The value of a metric for a given component at a given time. Example: 125 processes in BusinessWorks project MyProject
Metric	A type of measurement. Metrics can have varying values, or measures, over time. Examples: number of lines of code, complexity, etc. A metric may be either: • Qualitative: gives a quality indication on the component (ex: density of duplicated lines, line coverage by unit tests, etc.) • Quantitative: does not give a quality indication on the component (ex: number of lines of code, complexity, etc.)
Quality Profile	A set of coding rules. Each snapshot is based on a single quality profile.
Snapshot	A set of measures and issues on a given component at a given time. A snapshot is generated for each analysis.

D.2.5 Web interface

Concept	Definition
Dashboard	Web page that provides a way to display any data stored in the database. A dashboard is a set of widgets.
Widget	It is a box that displays data on a dashboard. There are two types of widget: • Global widget - displays data from multiple projects • Project widget - displays data from a specific project
Drilldown	A file-specific presentation of measure data. Some metrics have specialized presentations.

D.3 Analysis pipeline

A SonarQube analysis follows the following lifecycle:

1. Bootstrapper (SonarQube Maven Plugin, SonarQube Runner, SonarQube Ant Task) collects a set of properties describing the project to analyze and starts the batch.

2. ProjectBuilder extensions are called to give a chance for plugins to change project structure (add/remove module, change any property). After this step project structure can't be modified.

3. For each module (bottom-up):

   - Initializer extensions are called to give a chance to customize module configuration (add/remove sources, change any property)

   - SonarQube FileSystem is constructed (ie list of files to analyze). All project files are indexed according to configuration (inclusions/exclusions). After this step the FileSystem can't be modified.

   - Sensor extensions are called. Usually to add measures/issues on files.

   - Decorator extensions are called bottom-up on each element of the resource tree (File -> Directory -> Module -> Project). Usually to aggregate measures or compute "level-2" issues (issues based on result of sensors).

   - All collected data (measures, issues, etc...) are persisted. No addition for this module is permitted after this step.

4. Results of analysis are sent to the server

5. PostJob extensions are called. A PostJob can access all results of the analysis but not change anything. Used for example to produce various reports (PDF, CSV).

D.4 Plugin extensions

A SonarQube plugin is a set of Java objects that implement extension points. These extension points are interfaces or abstract classes which model an aspect of the system and define contracts of what needs to be implemented. An extension point is a point in the application where plugin code can be invoked, such as webapp page or code analyzer. Extension points are generally interfaces that can be implemented by plugins. Implementations have to be declared in the method org.sonar.api.SonarPlugin#getExtensions() and are then injected in the IoC container.

The extension points are listed and documented in the Javadoc of SonarQube.

E Features

E.1 Code analysis

SonarQube extensions (based on standard API) have been implemented in order to manage code analysis for BusinessWorks 6 projects.

E.1.1 Languages

Similar to programming languages like Java, Groovy, C-Sharp, Android, PHP, JavaScript etc. There is a new language that has been defined in SonarQube for TIBCO BusinessWorks 6.

1. The BusinessWorks 6 language defined in SonarQube will scan through the BusinessWorks 6 projects/applications and perform analysis against defined set of rules.

E.1.2 Profiles

Quality Profile is a set of coding rules. The **BusinessWorks 6 Profile** has extensive set of rules defined for BusinessWorks 6 language. The coding rules are based on code and design best practices.

E.1.3 Sensors

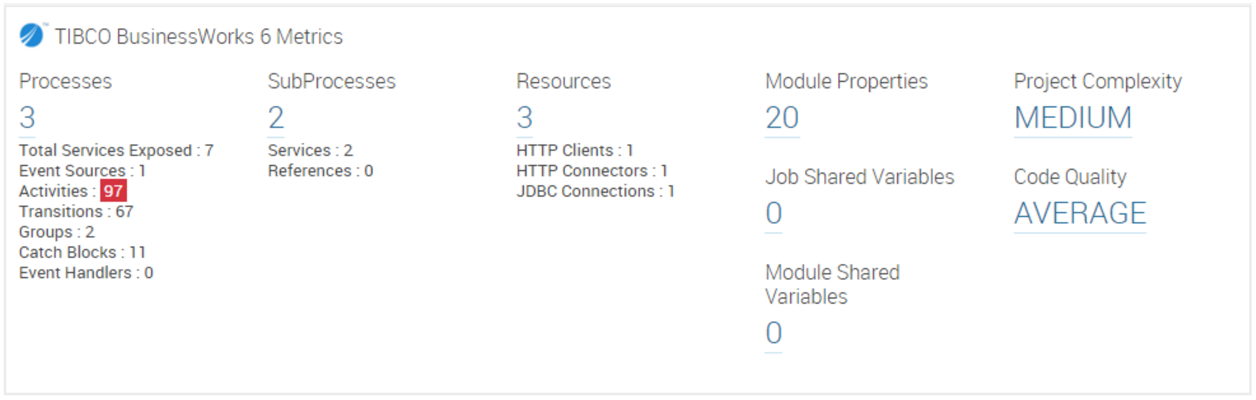
Two kinds of sensors are implemented in SonarQube BusinessWorks6 plugin:

A) Metrics sensors that count and calculate all the measures related to BusinessWorks 6 projects
B) Rules sensors executing coding rules, checking code quality and to raising issues.

E.2 UI extensions

E.2.1 BusinessWorks Metrics Widget

A new widget has been implemented in order to show BusinessWorks 6 project metrics:


TIBCO BusinessWorks 6 Metrics

Processes
3
Total Services Exposed : 7
Event Sources : 1
Activities : 97
Transitions : 67
Groups : 2
Catch Blocks : 11
Event Handlers : 0

SubProcesses
2
Services : 2
References : 0

Resources
3
HTTP Clients : 1
HTTP Connectors : 1
JDBC Connections : 1

Module Properties
20

Job Shared Variables
0

Module Shared Variables
0

Project Complexity
MEDIUM

Code Quality
AVERAGE

This widget is implemented in BusinessWorksMetricsWidget class of com.tibco.sonar.plugins.bw.widget package, based on a ruby (erb) template defined in the resource folder: /com/tibco/businessworks6/sonar/plugin/widget/BusinessWorksMetrics.html.erb

It gives a quick overview of BW project size, with trend on each measure.

F Rules for BusinessWorks 6.x analysis

F.1 BusinessWorks 6.x

F.1.1 Deadlock Detection Check

Description	There are many situations in which deadlocks can be created between communicating web services. This rule checks for deadlocks and infinite loops in BW6 process design.
Priority	BLOCKER
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	DeadLockCheck

F.1.2 Activities in Critical Section Check

Description	Critical section groups cause multiple concurrently running process instances to wait for one process instance to execute the activities in the group. As a result, there may be performance implications when using these groups. This rules checks that the Critical Section group does not include any activities that wait for incoming events or have long durations, such as Request/Reply activities, Wait For (Signal-In) activities, Sleep activity, or other activities that require a long time to execute.
Priority	CRITICAL
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	CriticalSectionCheck

F.1.3 Checkpoint Activity inside Critical Section Group Check

Description	This rule checks the placement of a Checkpoint activity within a process. It’s a bad design practice to place Checkpoint activity within a Critical Section Group.
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	CheckpointInTransation

F.1.4 Multiple Transitions Check

Description	EMPTY activity should be used if you want to join multiple transition flows. For example, there are multiple transitions out of an activity and each transition takes a different path in the process. In this scenario you can create a transition from the activity at the end of each path to an Empty activity to resume a single flow of execution in the process. This rule checks whether multiple transitions from an activity in a parallel flow merge into EMPTY activity
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	MultipleTransitionCheck

F.1.5 Log Only in Subprocess Check

Description	If there is logging or auditing required at multiple points in your project, its advised to write logging and auditing code in a SubProcess and invoke this process from any point where this functionality is required. This rule checks whether LOG activity is used in subprocess
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	LogOnlyInSubprocessCheck

F.1.6 Checkpoint inside Transaction Group Check

Description	This rule checks the placement of a Checkpoint activity within a process. Do not place checkpoint within or in parallel to a Transaction Group. Checkpoint activities should be placed at points that are guaranteed to be reached before or after the transaction group is reached.
Priority	CRITICAL
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	CheckpointInTransation

F.1.7 Checkpoint after HTTP Activities Check

Description	This rule checks the placement of a Checkpoint activity within a process. When placing your checkpoint in a process, be careful with certain types of process starters or incoming events, so that a recovered process instance does not attempt to access resources that no longer exist. For example, consider a process with an HTTP process starter that takes a checkpoint after receiving a request but before sending a response. In this case, when the engine restarts after a crash, the recovered process instance cannot respond to the request since the HTTP socket is already closed. As a best practice, do not place Checkpoint activity right after or in parallel path to HTTP activities.
Priority	CRITICAL
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	CheckpointAfterHttpCheck

F.1.8 Checkpoint after REST Webservice Call Check

Description	This rule checks the placement of a Checkpoint activity within a process. Do not place checkpoint after or in a parallel flow of REST webservice call.
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	CheckpointAfterHttpCheck

F.1.9 Checkpoint after JDBC Query Activity Check

Description	This rule checks the placement of a Checkpoint activity within a process. Do not place checkpoint after or in a parallel flow of Query activities or idempotent activities. Database operations such as Update, Insert and Delete are considered non-idempotent operations. You should always place a checkpoint immediately after any database insert or update activity to persist the response. However, for queries, there is no need to place checkpoints
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	CheckpointAfterJDBCÇheck

F.1.10 Choice Condition with No Otherwise Check

Description	This rule checks all activities input mapping for choice statement. As a coding best practice, the choice statement should always include the option otherwise.
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	ChoiceOtherwiseCheck

F.1.11 Transition Labels Check

Description	This rule checks whether the transitions with the type 'Success With Condition' (XPath) have a proper label. This will improve code readability
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	TransitionLabelCheck

F.1.12 JDBC WildCard Check

Description	This rule checks whether JDBC activities are using wildcards in the query. As a good coding practice, never use wildcards in JDBC queries.
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	JDBCWildCardCheck

F.1.13 JDBC HardCoded Check

Description	This rule checks JDBC activities for hardcoded values for fields Timeout and MaxRows. Use Process property or Module property.
Priority	MAJOR
Type	Custom Rule

Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	JDBCHardCodeCheck

F.1.14 JMS HardCoded Check

Description	This rule checks JMS activities for hardcoded values for fields Timeout, Destinaton, Reply to Destination, Message Selector, Polling Interval. Use Process property or Module property
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	JMSHardCodeCheck

F.1.15 For-Each Group Check

Description	This rule checks the ForEach group. It is recommended to use For-Each activity input mapping instead of using For-Each/Iteration Group wherever possible. Do not use iteration groups just for mapping repeating elements.
Priority	INFO
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	ForEachGroupCheck

F.1.16 For-Each Mapping Check

Description	This rule checks the Input mappings of activities. In activity Input mapping for performance reasons, it is recommended ato use Copy-Of instead of For-Each whenever possible.
Priority	INFO
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	ForEachMappingCheck

F.1.17 JMS Acknowledgement Mode Check

Description	This rule checks the acknowledgement mode used in JMS activities. Avoid using Auto Acknowledgement to minimize the risk of data loss.
Priority	INFO
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	JMSAcknowledgementModeCheck

F.1.18 Data Availability to Inline SubProcess Check

Description	This rule checks if there is large set of data being passed everytime to Inline SubProcess. Use of Job Shared Variable is recommended in this scenario to increase performance.
Priority	INFO
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	SubProcessInlineCheck

F.1.19 Number of Activities Check

Description	This rule checks the number of activities within a process, too many activities reduces the process readability.
Priority	MINOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	NumberofActivitiesCheck

F.1.20 Number of Exposed Services Check

Description	This rule checks the number of exposed services within a process. It is a good design practice to construct not more than 5 services in the same process.
Priority	MAJOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process

Class	NumberofServicesCheck

F.1.21 No Process Description Check

Description	This rule checks if there is description specified for a process
Priority	MINOR
Type	Custom Rule
Package	com.tibco.businessworks6.sonar.plugin.check.process
Class	NoDescriptionCheck

G Administration guide

G.1 Requirements

- SonarQube 4.5.6 or later
- JDK/JRE 7 or later

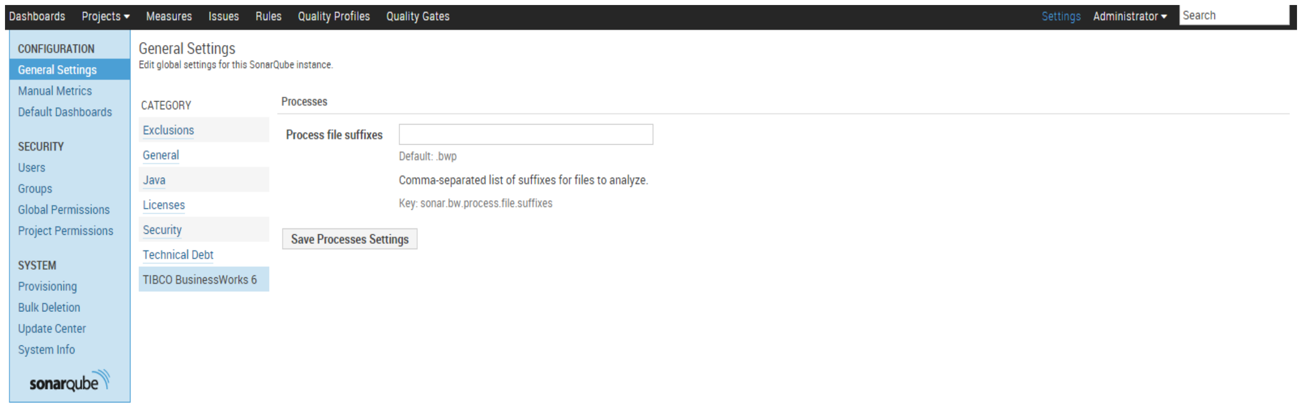
G.2 Installation

Copy the delivered jar into /extensions/plugins folder, relative to your Sonar installation folder.

G.3 Configuration

G.3.1 General Settings

You will be able to redefine files extensions for each language using the General Configuration Section in SonarQube Settings > Configuration.



General Settings

Edit global settings for this SonarQube instance.

Processes

Process file suffixes

Default: .bwp

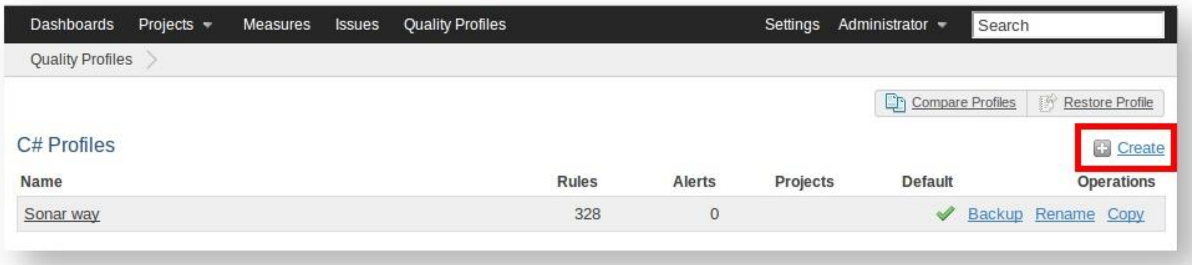
Comma-separated list of suffixes for files to analyze.

Key: sonar.bw.process.file.suffixes

G.4 Customization

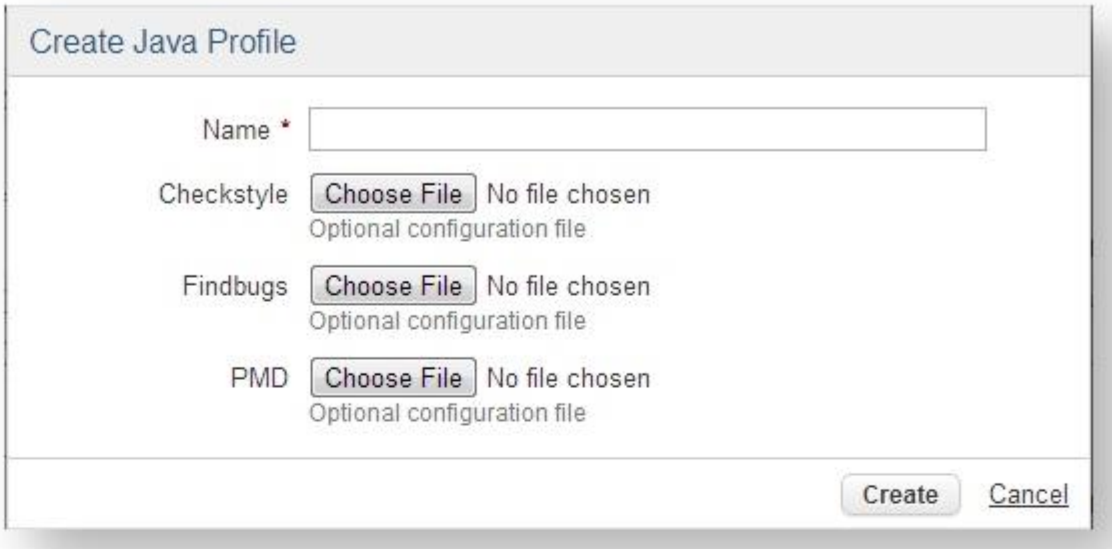
G.4.1 Quality Profiles

To create a quality profile, click on the Create button on the upper right of the Quality Profiles page.



Name	Rules	Alerts	Projects	Default	Operations
Sonar way	328	0		✓	Backup Rename Copy

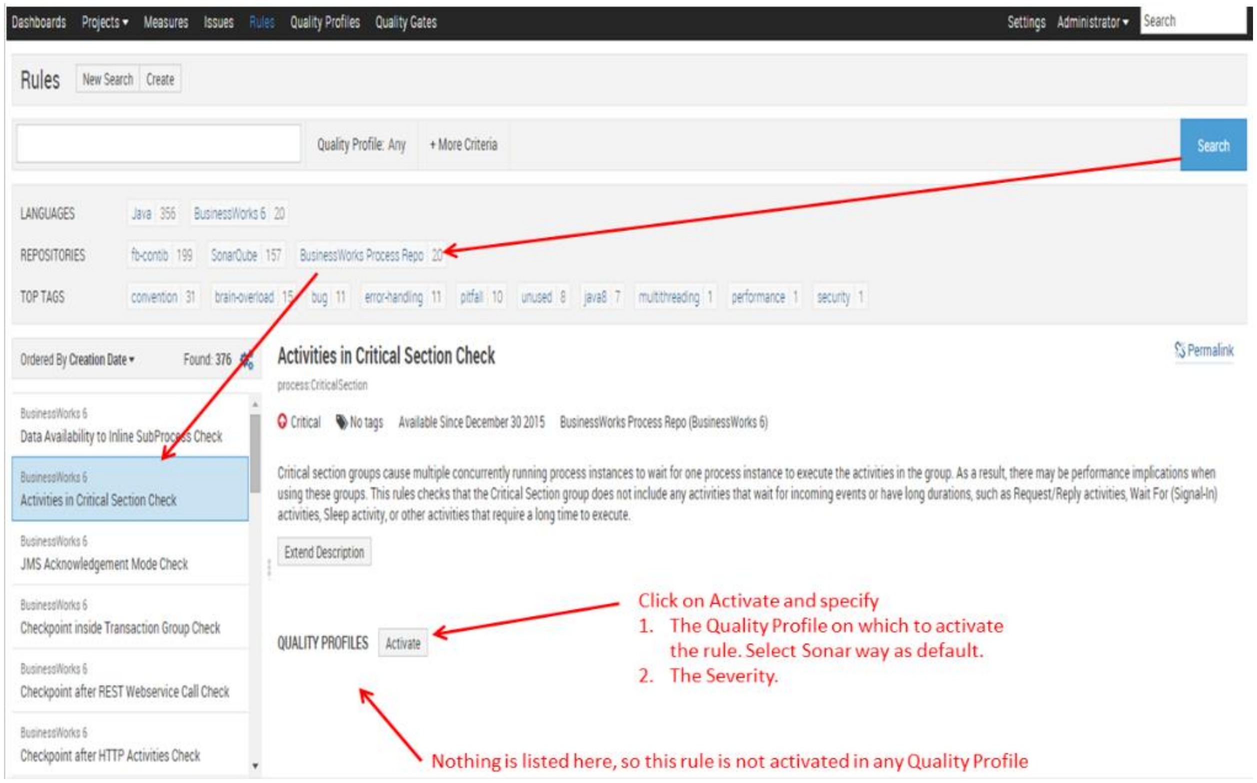
In the dialog that pops up enter the name of the quality profile. It must be unique among profile names for that language. For some languages, such as Java and PHP, you can optionally provide configuration files for the external tools used during analysis in order to pre-populate the new quality profile with some existing rules configurations. For Java you can provide files for Checkstyle, PMD and Findbugs.



G.4.2 Rules

Whatever search you're doing on the Rules page, you always have the opportunity to activate the rule you're looking at on a quality profile (assuming you're logged in and have the correct permissions).

For instance, let's say that you're browsing all the rules working on “BusinessWorks Process” language and you find that one is not activated in any of your profiles:

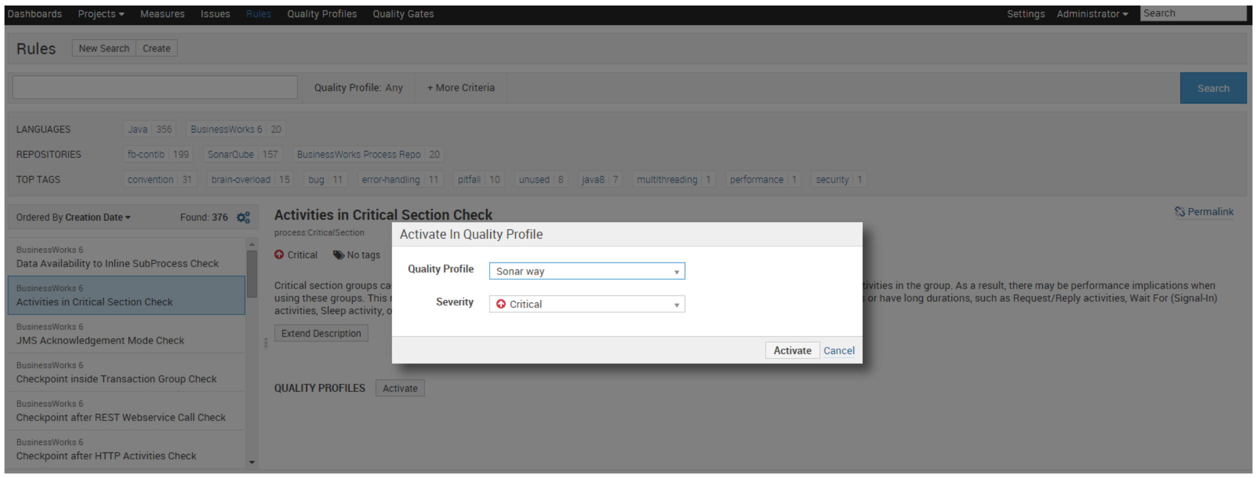

Dashboards
Projects
Measures
Issues
Rules
Quality Profiles
Quality Gates
Settings
Administrator
Search
Rules
New Search
Create
Quality Profile: Any
+ More Criteria
Search
LANGUAGES Java 356 BusinessWorks 6 20
REPOSITORIES fb-contrib 199 SonarQube 157 BusinessWorks Process Repo 20
TOP TAGS convention 31 brain-overload 15 bug 11 error-handling 11 pitfall 10 unused 8 java8 7 multithreading 1 performance 1 security 1
Ordered By Creation Date
Found: 376
BusinessWorks 6 Data Availability to Inline SubProcess Check
BusinessWorks 6 Activities in Critical Section Check
BusinessWorks 6 JMS Acknowledgement Mode Check
BusinessWorks 6 Checkpoint inside Transaction Group Check
BusinessWorks 6 Checkpoint after REST Webservice Call Check
BusinessWorks 6 Checkpoint after HTTP Activities Check
Permalink
Activities in Critical Section Check
process:CriticalSection
Critical
No tags
Available Since December 30 2015
BusinessWorks Process Repo (BusinessWorks 6)
Extend Description
QUALITY PROFILES
Activate

Critical section groups cause multiple concurrently running process instances to wait for one process instance to execute the activities in the group. As a result, there may be performance implications when using these groups. This rules checks that the Critical Section group does not include any activities that wait for incoming events or have long durations, such as Request/Reply activities, Wait For (Signal-In) activities, Sleep activity, or other activities that require a long time to execute.

Click on Activate and specify
1. The Quality Profile on which to activate the rule. Select Sonar way as default.
2. The Severity.

Nothing is listed here, so this rule is not activated in any Quality Profile

After clicking on the activate button, you will be able to configure your rule:



Quality Profile: Sonar way

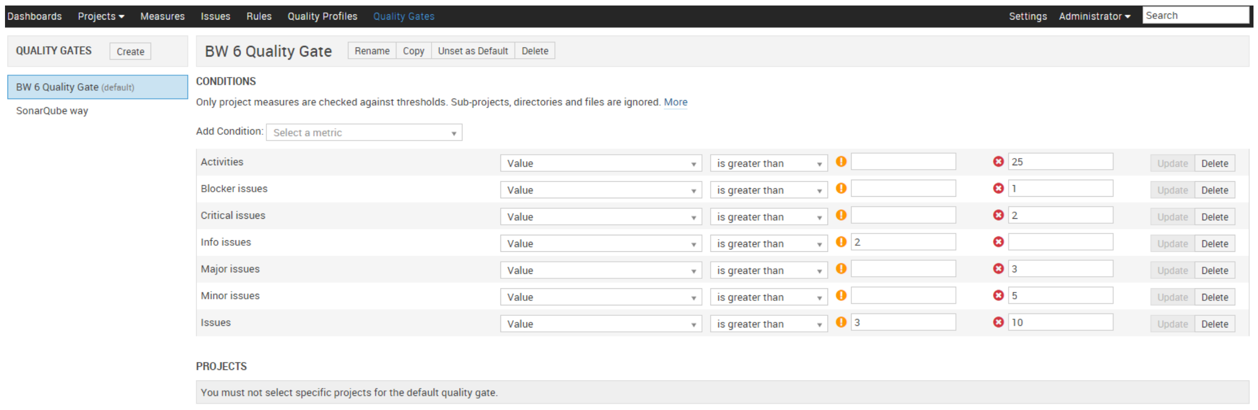
Severity: Critical

Once the rule is activated, it appears in the list of the "Quality Profiles" section, in BusinessWorks 6 Profiles.

Note that you do not necessarily need to do this activation rule by rule. You have the option to bulk activate/deactivate all the rules returned by your search for a single profile:

G.4.3 Quality Gates

To manage quality gates, go to Quality Gates (top bar):



BW 6 Quality Gate

CONDITIONS

Only project measures are checked against thresholds. Sub-projects, directories and files are ignored. More

Add Condition: Select a metric

Activities	Value	is greater than		25	Update Delete
Blocker issues	Value	is greater than		1	Update Delete
Critical issues	Value	is greater than		2	Update Delete
Info issues	Value	is greater than	2		Update Delete
Major issues	Value	is greater than		3	Update Delete
Minor issues	Value	is greater than		5	Update Delete
Issues	Value	is greater than	3	10	Update Delete

PROJECTS

You must not select specific projects for the default quality gate.

A quality gate is a set of conditions and a set of projects to be checked against these conditions. Conditions can be set on measures (i.e. No blocker issues) or on deltas (i.e. No new blocker issues since previous version). Two thresholds can be set for each condition: warning and error.

H Developer Guide

H.1 Coding a new rule

1) Create a new class (ex MyNewRuleCheck) in package com.tibco.businessworks6.sonar.plugin.check.process

2) Extend class AbstractProcessCheck and write your logic for the rule in the implemented method

   protected void validate(ProcessSource processSource)

3) Add default rule configuration annotations

```
@Rule(key = MyNewRuleCheck.RULE_KEY, name="Example Rule Check", priority = Priority.MAJOR,
description = "This is an example rule for BusinessWorks 6")
@BelongsToProfile(title = ProcessSonarWayProfile.defaultProfileName, priority = Priority.MAJOR)
```

4) In the implemented validate method

   a) When the conditions of the rule are met, create a violation (com.tibco.businessworks6.sonar.plugin.violation.Violation) such as:

```
Violation violation = new DefaultViolation(getRule(), 1, "MyNewRule conditions are fulfilled");
```

   a. Add violation to source code.

```
processSource.addViolation(violation);
```

5) In class ProcessRuleDefinition of package com.tibco.businessworks6.sonar.plugin.rulerepository add your rule in the constant

```
public static Class check[] = {
                com.tibco.businessworks6.sonar.plugin.check.process.NoDescriptionCheck.class,
                com.tibco.businessworks6.sonar.plugin.check.process.NumberofActivitiesCheck.class,
                com.tibco.businessworks6.sonar.plugin.check.process. MyNewRuleCheck.class
        };
```

6) In constructor of the class AbstractRuleSensor add the rule to the list

```
List<Class> allChecks = new ArrayList<Class>();
allChecks.add(NoDescriptionCheck.class);
allChecks.add(NumberofActivitiesCheck.class);
allChecks.add(MyNewRuleCheck.class);
```

7) Setup Sonar to add the rule in the current profile as explained in G.4.2

I User guide

I.1 Execute an analysis

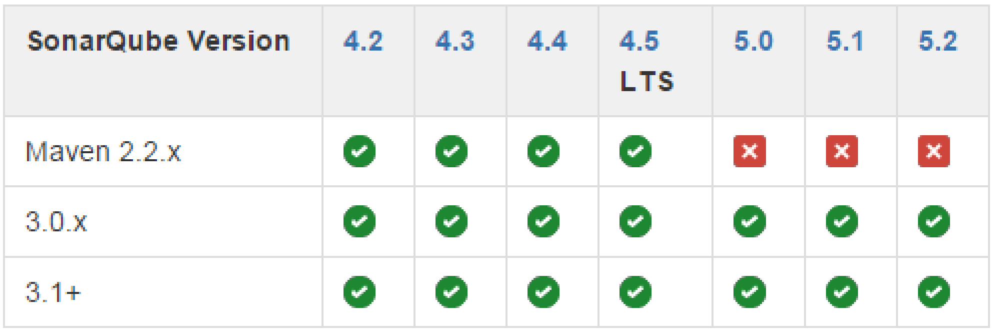
I.1.1 Introduction

First, you should install the plugin(s) for the language(s) of the project to be analyzed, either by a direct download or through the update center.

Then, you need to choose an analysis method. The following are available:
- Analyzing with SonarQube Runner (recommended analyzer)
- Analyzing with Maven
- Analyzing with SonarQube Ant Task
- Analyzing with Gradle
- CI Engines

Compatibility Matrix

This chart shows the backward compatibility of the current version of each analysis engine.



SonarQube Version	4.2	4.3	4.4	4.5 LTS	5.0	5.1	5.2
Maven 2.2.x	✓	✓	✓	✓	✗	✗	✗
3.0.x	✓	✓	✓	✓	✓	✓	✓
3.1+	✓	✓	✓	✓	✓	✓	✓

I.1.2 Using Maven

I.1.2.1 Prerequisites

Download and install Maven (see Compatibility Matrix).

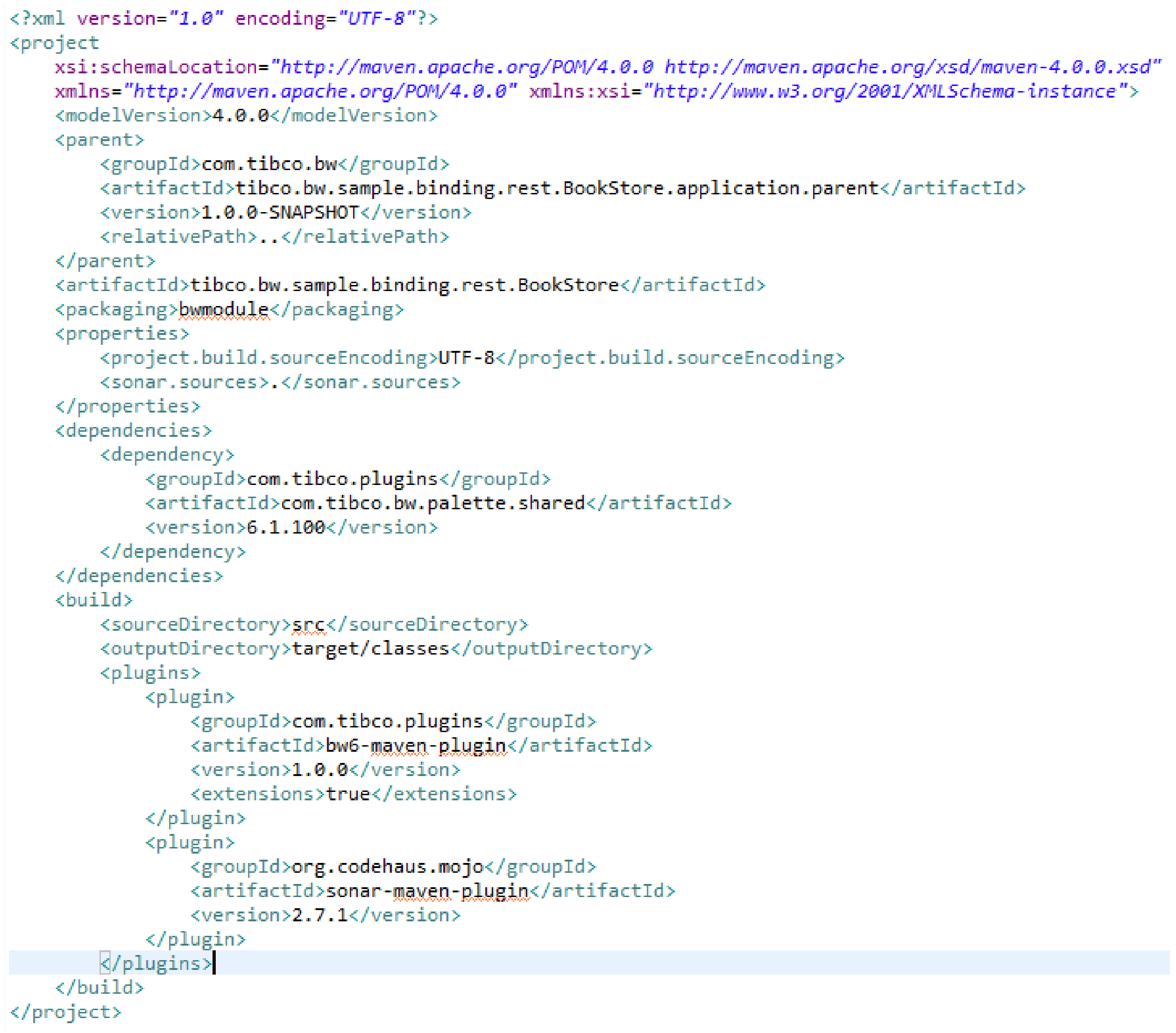
You must have previously installed and configured Maven for SonarQube (http://docs.sonarqube.org/display/SONAR/Installing+and+Configuring+SonarQube+Scanner+for+Maven).

This link explains how to set global settings in ***settings.xml*** for database parameters to be used as well as the SonarQube server URL.

I.1.2.2 Configure your pom.xml

Generate the pom.xml using the TIBCO BusinessWorks 6 Maven plugin. Thereafter, configure the pom of your project to tell to SonarQube where is included your source code.

Mainly, you should define a sonar.sources property pointing your project folder:



```
<?xml version="1.0" encoding="UTF-8"?>
<project
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
    xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.tibco.bw</groupId>
        <artifactId>tibco.bw.sample.binding.rest.BookStore.application.parent</artifactId>
        <version>1.0.0-SNAPSHOT</version>
        <relativePath>..</relativePath>
    </parent>
    <artifactId>tibco.bw.sample.binding.rest.BookStore</artifactId>
    <packaging>bwmodule</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <sonar.sources>.</sonar.sources>
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.tibco.plugins</groupId>
            <artifactId>com.tibco.bw.palette.shared</artifactId>
            <version>6.1.100</version>
        </dependency>
    </dependencies>
    <build>
        <sourceDirectory>src</sourceDirectory>
        <outputDirectory>target/classes</outputDirectory>
        <plugins>
            <plugin>
                <groupId>com.tibco.plugins</groupId>
                <artifactId>bw6-maven-plugin</artifactId>
                <version>1.0.0</version>
                <extensions>true</extensions>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>sonar-maven-plugin</artifactId>
                <version>2.7.1</version>
            </plugin>
        </plugins>
    </build>
</project>
```

I.1.2.3 Run the analysis

Then it’s very simple to run your analysis. You just have to execute the following goal on your project:

```
mvn clean install sonar:sonar
```

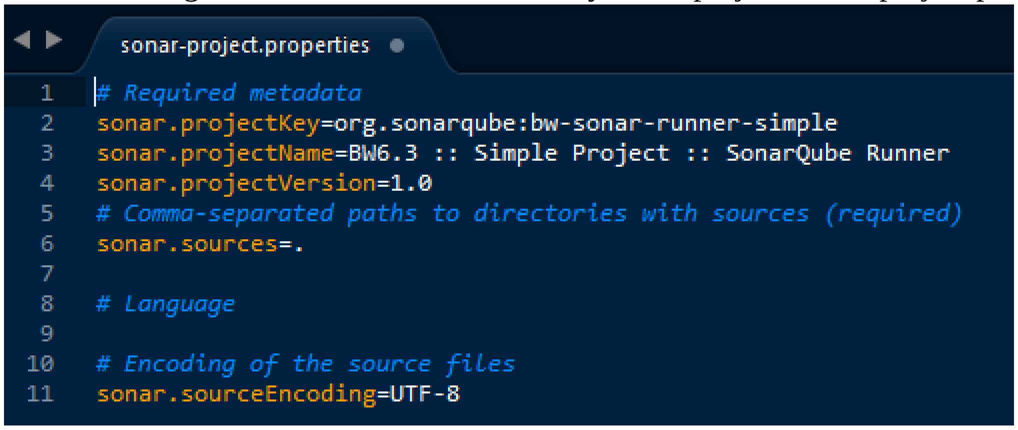
I.1.3 Using SonarQube Runner

I.1.3.1 Prerequisites

You must have previously installed the SonarQube Runner and read Analyzing Code Source. You must also have the BusinessWorks SonarQube plugin installed.

I.1.3.2 Create project configuration file

Create a configuration file in the root directory of the project: sonar-project.properties



```
# Required metadata
sonar.projectKey=org.sonarqube:bw-sonar-runner-simple
sonar.projectName=BW6.3 :: Simple Project :: SonarQube Runner
sonar.projectVersion=1.0
# Comma-separated paths to directories with sources (required)
sonar.sources=.

# Language

# Encoding of the source files
sonar.sourceEncoding=UTF-8
```

I.1.3.3 Run the analysis

Run the following command from the project base directory to launch the analysis:

```
sonar-runner
```

I.1.4 Using Jenkins

Please, check the official documentation about the Jenkins plugin for SonarQube:

http://docs.sonarqube.org/display/SONAR/Analyzing+with+SonarQube+Scanner+for+Jenkins

I.2 Customize a report

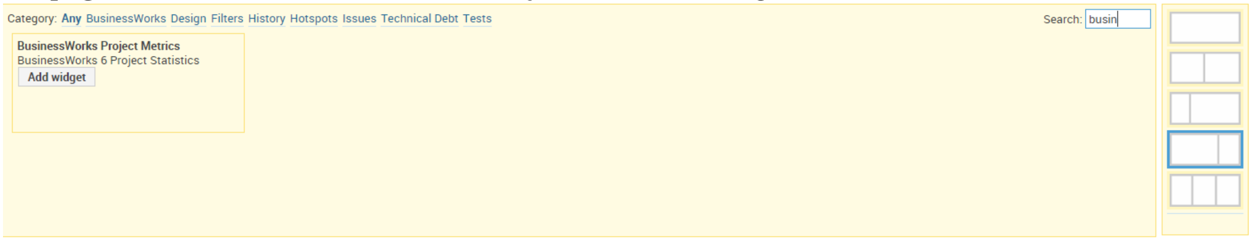
I.2.1 Introduction

Users should customize their SonarQube web interface to stay focused on what is of interest for them:

- Customizing Dashboards
- Favorite
- Notifications

I.2.2 Add the BusinessWorks Metrics Widget

To add a widget, click on Configure widgets. The list of available widgets is shown at the top of the page. Search for "*BusinessWorks Project Metrics*" widget.

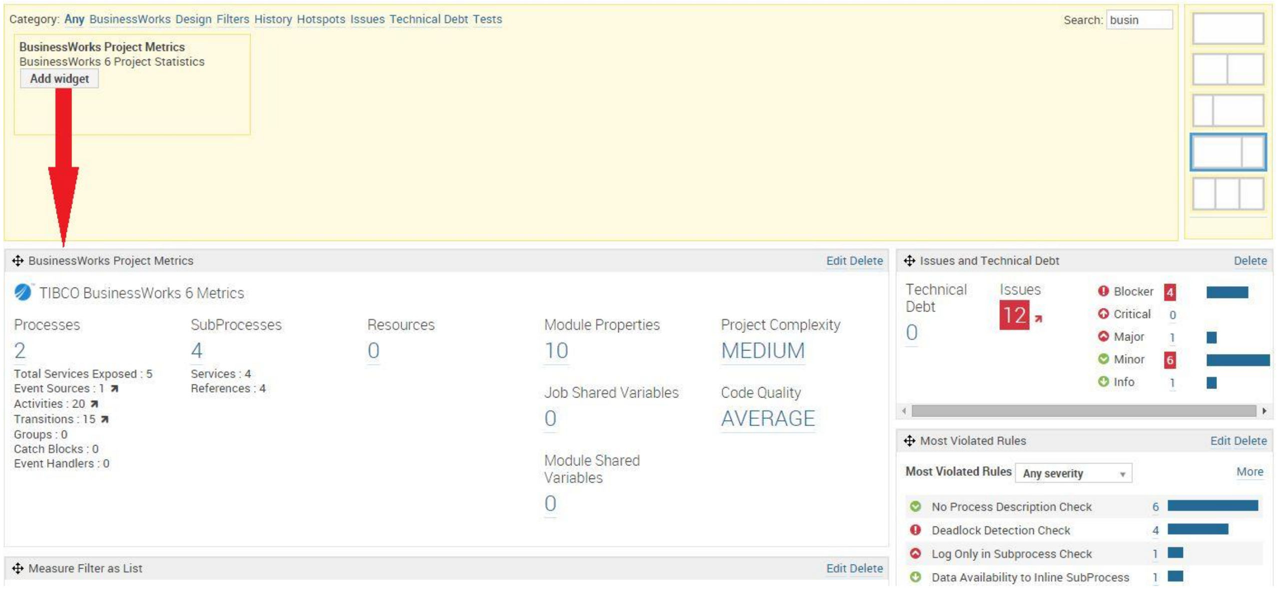


Category: Any BusinessWorks Design Filters History Hotspots Issues Technical Debt Tests

Search: busin

BusinessWorks Project Metrics
BusinessWorks 6 Project Statistics
Add widget

Then you can click on Add widget button to append the "*BusinessWorks Project Metrics*" widget into your dashboard:



Category: Any BusinessWorks Design Filters History Hotspots Issues Technical Debt Tests

Search: busin

BusinessWorks Project Metrics
BusinessWorks 6 Project Statistics
Add widget

BusinessWorks Project Metrics — Edit Delete

TIBCO BusinessWorks 6 Metrics

Processes: 2
Total Services Exposed : 5
Event Sources : 1
Activities : 20
Transitions : 15
Groups : 0
Catch Blocks : 0
Event Handlers : 0

SubProcesses: 4
Services : 4
References : 4

Resources: 0

Module Properties: 10

Job Shared Variables: 0

Module Shared Variables: 0

Project Complexity: MEDIUM

Code Quality: AVERAGE

Measure Filter as List — Edit Delete

Issues and Technical Debt — Delete

Technical Debt: 0

Issues: 12

Blocker	4
Critical	0
Major	1
Minor	6
Info	1

Most Violated Rules — Edit Delete

Most Violated Rules: Any severity — More

No Process Description Check	6
Deadlock Detection Check	4
Log Only in Subprocess Check	1
Data Availability to Inline SubProcess	1

Obviously, you can position this widget wherever you want in your dashboard, as any other widget.

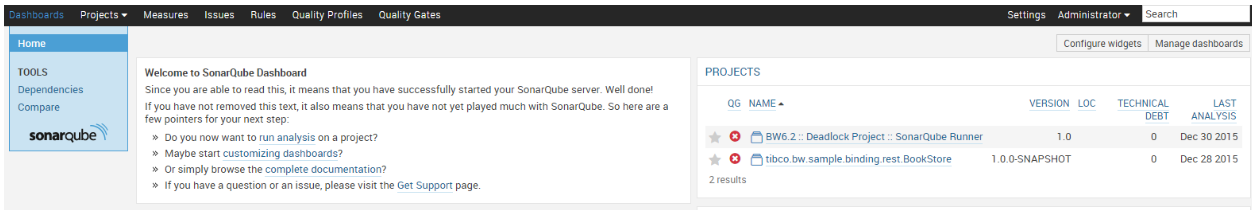
I.3 BusinessWorks Process visualization

I.3.1 Issues Visualization

This feature is usable in any SonarQube web page.

The following steps allow you visualizing your processes:

1) After running analysis on TIBCO BusinessWorks 6 project, on your browser navigate to *http://localhost:9000/*
   Click on the project you have run analysis on.



Welcome to SonarQube Dashboard

Since you are able to read this, it means that you have successfully started your SonarQube server. Well done!

If you have not removed this text, it also means that you have not yet played much with SonarQube. So here are a few pointers for your next step:

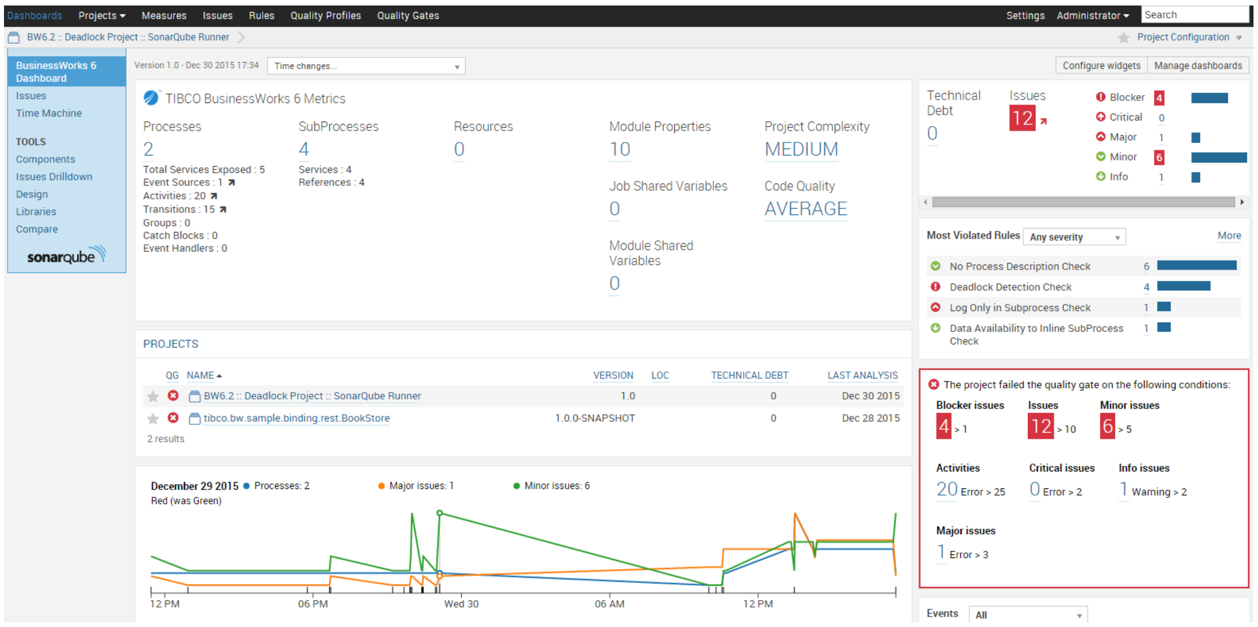
» Do you now want to run analysis on a project?
» Maybe start customizing dashboards?
» Or simply browse the complete documentation?
» If you have a question or an issue, please visit the Get Support page.

PROJECTS

QG	NAME	VERSION	LOC	TECHNICAL DEBT	LAST ANALYSIS
	BW6.2 :: Deadlock Project :: SonarQube Runner	1.0		0	Dec 30 2015
	tibco.bw.sample.binding.rest.BookStore	1.0.0-SNAPSHOT		0	Dec 28 2015

2 results

2) After clicking on the project you are redirected to Dashboard page where you can see various widgets highlighting BusinessWorks 6 project statistics, Issues etc.



Version 1.0 - Dec 30 2015 17:34

TIBCO BusinessWorks 6 Metrics

Processes 2
Total Services Exposed : 5
Event Sources : 1
Activities : 20
Transitions : 15
Groups : 0
Catch Blocks : 0
Event Handlers : 0

SubProcesses 4
Services : 4
References : 4

Resources 0

Module Properties 10

Job Shared Variables 0

Module Shared Variables 0

Project Complexity MEDIUM

Code Quality AVERAGE

PROJECTS

QG	NAME	VERSION	LOC	TECHNICAL DEBT	LAST ANALYSIS
	BW6.2 :: Deadlock Project :: SonarQube Runner	1.0		0	Dec 30 2015
	tibco.bw.sample.binding.rest.BookStore	1.0.0-SNAPSHOT		0	Dec 28 2015

2 results

December 29 2015 ● Processes: 2 ● Major issues: 1 ● Minor issues: 6
Red (was Green)

Technical Debt 0

Issues 12

Blocker	4
Critical	0
Major	1
Minor	6
Info	1

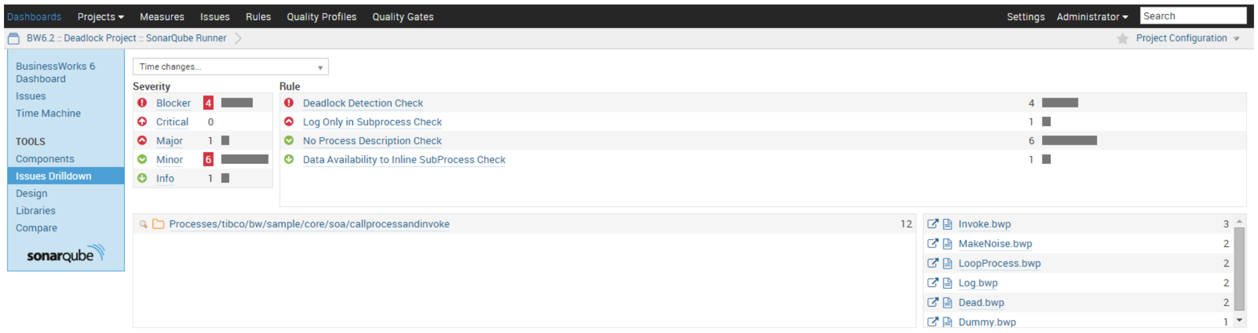
Most Violated Rules Any severity More

No Process Description Check	6
Deadlock Detection Check	4
Log Only in Subprocess Check	1
Data Availability to Inline SubProcess Check	1

The project failed the quality gate on the following conditions:

Blocker issues 4 > 1
Issues 12 > 10
Minor issues 6 > 5
Activities 20 Error > 25
Critical issues 0 Error > 2
Info issues 1 Warning > 2
Major issues 1 Error > 3

3) After clicking on Issues you will be redirected to a page where you will find which rules were violated and which processes the violations took place in.



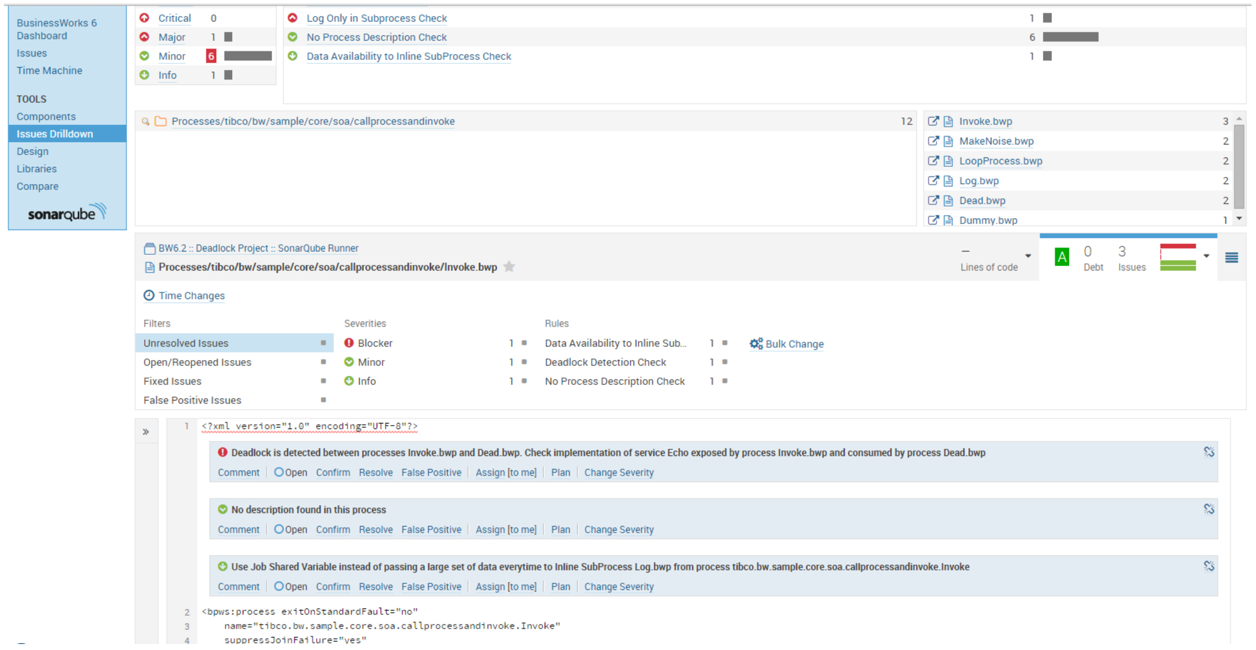
Severity	
Blocker	4
Critical	0
Major	1
Minor	6
Info	1

Rule	
Deadlock Detection Check	4
Log Only in Subprocess Check	1
No Process Description Check	6
Data Availability to Inline SubProcess Check	1

Processes/tibco/bw/sample/core/soa/callprocessandinvoke 12

Invoke.bwp	3
MakeNoise.bwp	2
LoopProcess.bwp	2
Log.bwp	2
Dead.bwp	2
Dummy.bwp	1

4) After clicking on the process you will see the violations in the process with detailed summary of the violation.



Critical	0
Major	1
Minor	6
Info	1

Log Only in Subprocess Check	1
No Process Description Check	6
Data Availability to Inline SubProcess Check	1

Processes/tibco/bw/sample/core/soa/callprocessandinvoke 12

Invoke.bwp	3
MakeNoise.bwp	2
LoopProcess.bwp	2
Log.bwp	2
Dead.bwp	2
Dummy.bwp	1

BW6.2 :: Deadlock Project :: SonarQube Runner

Processes/tibco/bw/sample/core/soa/callprocessandinvoke/Invoke.bwp

– Lines of code | A | 0 Debt | 3 Issues

Time Changes

Filters		Severities		Rules	
Unresolved Issues		Blocker	1	Data Availability to Inline Sub...	1
Open/Reopened Issues		Minor	1	Deadlock Detection Check	1
Fixed Issues		Info	1	No Process Description Check	1
False Positive Issues					

Bulk Change

```
<?xml version="1.0" encoding="UTF-8"?>
```

Deadlock is detected between processes Invoke.bwp and Dead.bwp. Check implementation of service Echo exposed by process Invoke.bwp and consumed by process Dead.bwp

Comment | Open Confirm Resolve False Positive | Assign [to me] | Plan | Change Severity

No description found in this process

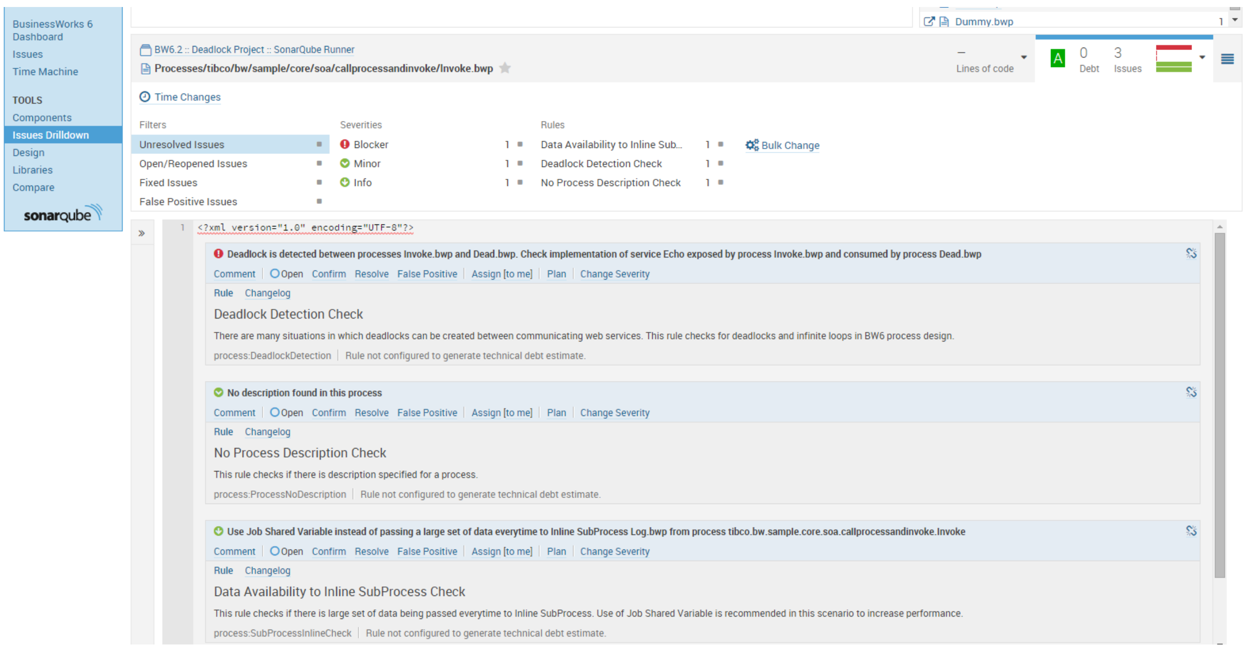
Comment | Open Confirm Resolve False Positive | Assign [to me] | Plan | Change Severity

Use Job Shared Variable instead of passing a large set of data everytime to Inline SubProcess Log.bwp from process tibco.bw.sample.core.soa.callprocessandinvoke.Invoke

Comment | Open Confirm Resolve False Positive | Assign [to me] | Plan | Change Severity

```
<bpws:process exitOnStandardFault="no"
    name="tibco.bw.sample.core.soa.callprocessandinvoke.Invoke"
    suppressJoinFailure="yes"
```

5) After clicking on violation you will see Rule detail.



BusinessWorks 6 Dashboard
Issues
Time Machine
TOOLS
Components
Issues Drilldown
Design
Libraries
Compare

BW6.2 :: Deadlock Project :: SonarQube Runner
Processes/tibco/bw/sample/core/soa/callprocessandinvoke/Invoke.bwp

– Lines of code | A | 0 Debt | 3 Issues

Time Changes

Filters		Severities		Rules	
Unresolved Issues		Blocker	1	Data Availability to Inline Sub...	1
Open/Reopened Issues		Minor	1	Deadlock Detection Check	1
Fixed Issues		Info	1	No Process Description Check	1
False Positive Issues					

Bulk Change

```
<?xml version="1.0" encoding="UTF-8"?>
```

Deadlock is detected between processes Invoke.bwp and Dead.bwp. Check implementation of service Echo exposed by process Invoke.bwp and consumed by process Dead.bwp

Comment | Open Confirm Resolve False Positive Assign [to me] | Plan | Change Severity

Rule Changelog

Deadlock Detection Check

There are many situations in which deadlocks can be created between communicating web services. This rule checks for deadlocks and infinite loops in BW6 process design.

process:DeadlockDetection | Rule not configured to generate technical debt estimate.

No description found in this process

Comment | Open Confirm Resolve False Positive Assign [to me] | Plan | Change Severity

Rule Changelog

No Process Description Check

This rule checks if there is description specified for a process.

process:ProcessNoDescription | Rule not configured to generate technical debt estimate.

Use Job Shared Variable instead of passing a large set of data everytime to Inline SubProcess Log.bwp from process tibco.bw.sample.core.soa.callprocessandinvoke.Invoke

Comment | Open Confirm Resolve False Positive Assign [to me] | Plan | Change Severity

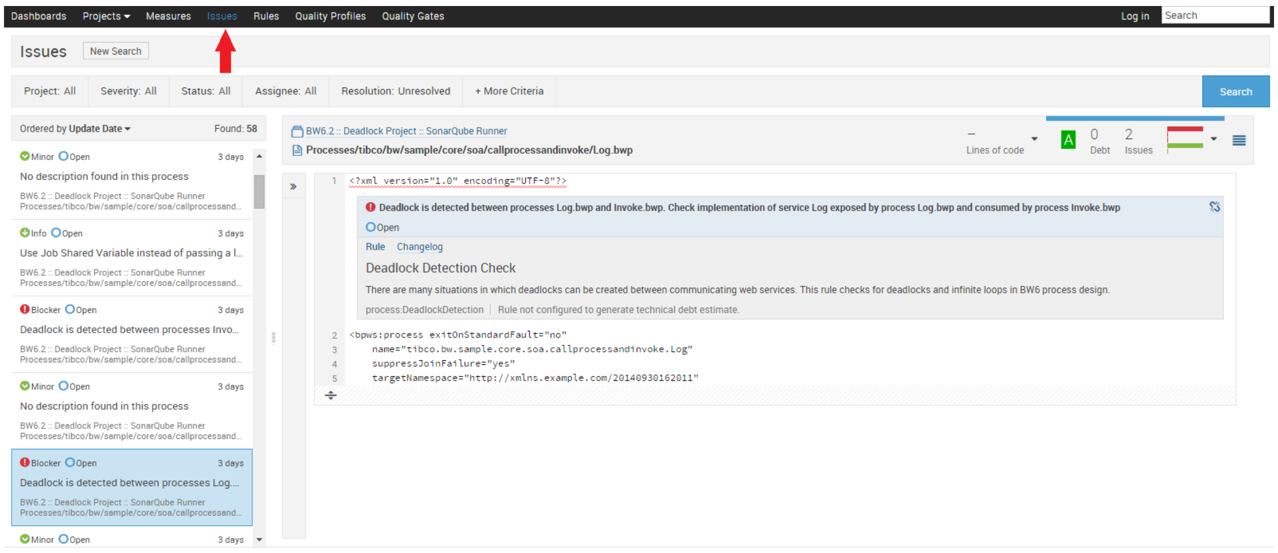
Rule Changelog

Data Availability to Inline SubProcess Check

This rule checks if there is large set of data being passed everytime to Inline SubProcess. Use of Job Shared Variable is recommended in this scenario to increase performance.

process:SubProcessInlineCheck | Rule not configured to generate technical debt estimate.

6) Alternatively, just click on Issues tab and you will be able to see all the issues listed in the project.



Issues New Search

Project: All | Severity: All | Status: All | Assignee: All | Resolution: Unresolved | + More Criteria | Search

Ordered by Update Date Found: 58

Minor Open 3 days
No description found in this process
BW6.2 :: Deadlock Project :: SonarQube Runner
Processes/tibco/bw/sample/core/soa/callprocessand...

Info Open 3 days
Use Job Shared Variable instead of passing a l...
BW6.2 :: Deadlock Project :: SonarQube Runner
Processes/tibco/bw/sample/core/soa/callprocessand...

Blocker Open 3 days
Deadlock is detected between processes Invo...
BW6.2 :: Deadlock Project :: SonarQube Runner
Processes/tibco/bw/sample/core/soa/callprocessand...

Minor Open 3 days
No description found in this process
BW6.2 :: Deadlock Project :: SonarQube Runner
Processes/tibco/bw/sample/core/soa/callprocessand...

Blocker Open 3 days
Deadlock is detected between processes Log....
BW6.2 :: Deadlock Project :: SonarQube Runner
Processes/tibco/bw/sample/core/soa/callprocessand...

Minor Open 3 days

BW6.2 :: Deadlock Project :: SonarQube Runner
Processes/tibco/bw/sample/core/soa/callprocessandinvoke/Log.bwp

– Lines of code | A | 0 Debt | 2 Issues

```
<?xml version="1.0" encoding="UTF-8"?>
```

Deadlock is detected between processes Log.bwp and Invoke.bwp. Check implementation of service Log exposed by process Log.bwp and consumed by process Invoke.bwp

Open

Rule Changelog

Deadlock Detection Check

There are many situations in which deadlocks can be created between communicating web services. This rule checks for deadlocks and infinite loops in BW6 process design.

process:DeadlockDetection | Rule not configured to generate technical debt estimate.

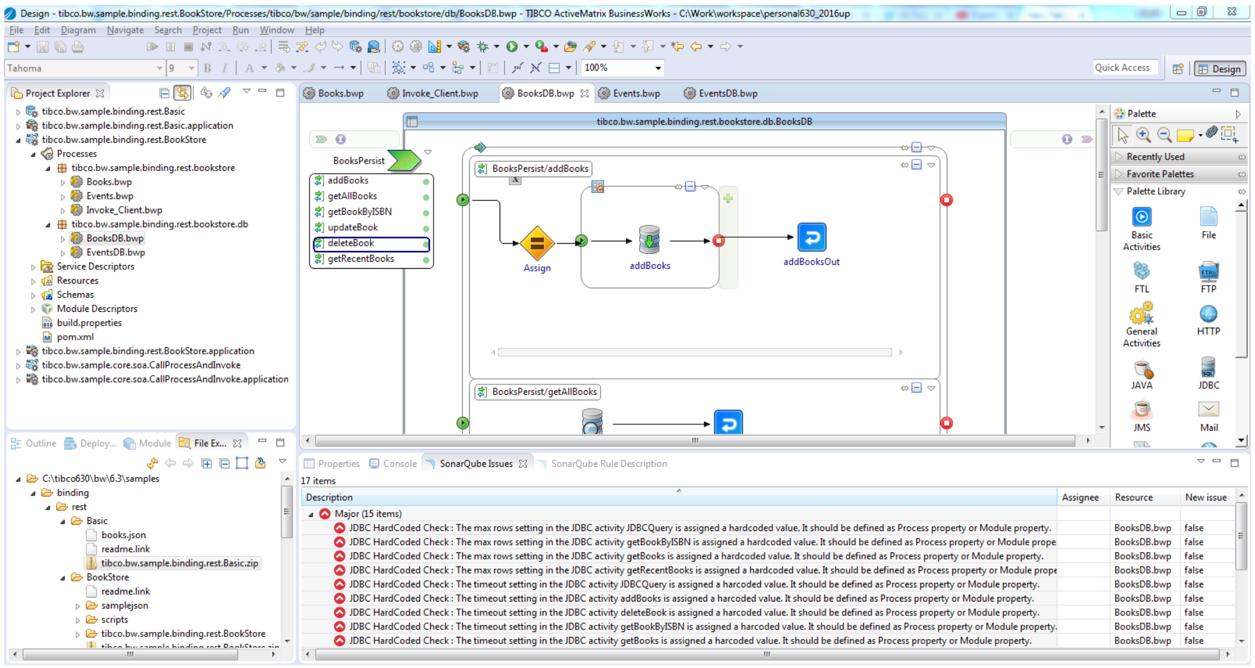
```
<bpws:process exitOnStandardFault="no"
    name="tibco.bw.sample.core.soa.callprocessandinvoke.Log"
    suppressJoinFailure="yes"
    targetNamespace="http://xmlns.example.com/20140930162011"
```

TIBCO® The Power of Now®

J Configuring SonarQube in TIBCO BusinessStudio

To setup SonarQube plugin for Eclipse, follow
http://docs.sonarqube.org/display/SONAR/Configuring+SonarQube+in+Eclipse

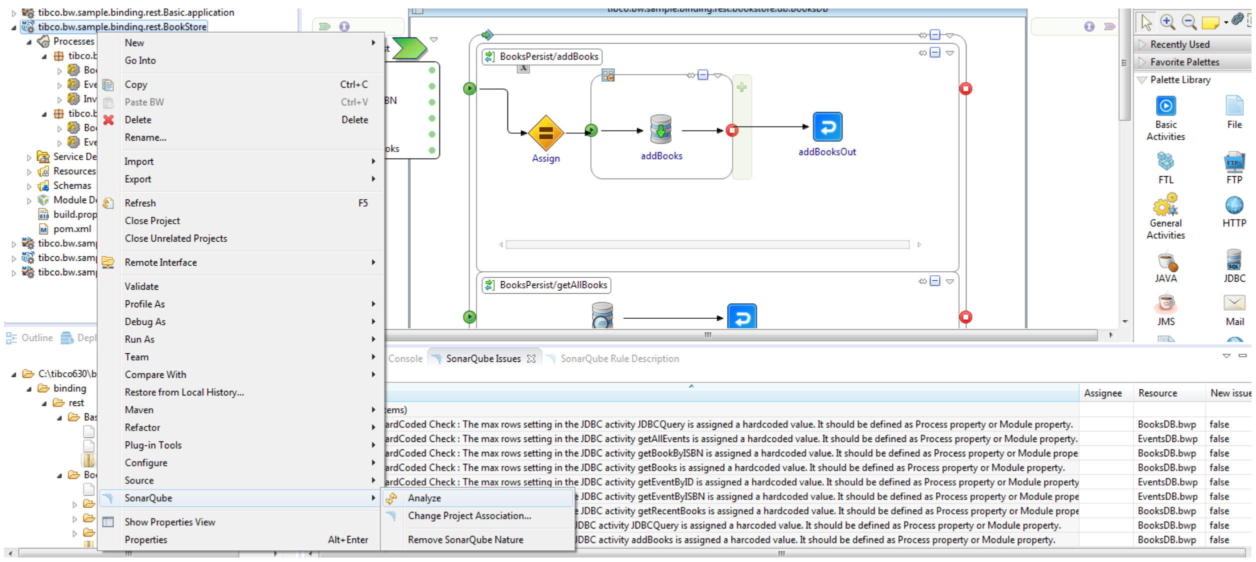
After running analysis you can view the issues in TIBCO BusinessStudio as well.



17 items

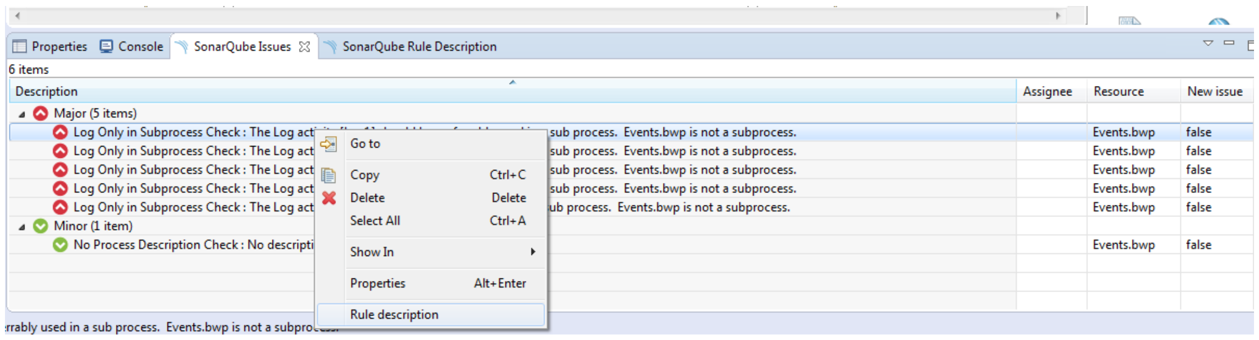
Description	Assignee	Resource	New issue
Major (15 items)			
JDBC HardCoded Check : The max rows setting in the JDBC activity JDBCQuery is assigned a hardcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
JDBC HardCoded Check : The max rows setting in the JDBC activity getBookByISBN is assigned a hardcoded value. It should be defined as Process property or Module prope		BooksDB.bwp	false
JDBC HardCoded Check : The max rows setting in the JDBC activity getBooks is assigned a hardcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
JDBC HardCoded Check : The max rows setting in the JDBC activity getRecentBooks is assigned a hardcoded value. It should be defined as Process property or Module prope		BooksDB.bwp	false
JDBC HardCoded Check : The timeout setting in the JDBC activity JDBCQuery is assigned a harcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
JDBC HardCoded Check : The timeout setting in the JDBC activity addBooks is assigned a harcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
JDBC HardCoded Check : The timeout setting in the JDBC activity deleteBook is assigned a harcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
JDBC HardCoded Check : The timeout setting in the JDBC activity getBookByISBN is assigned a harcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
JDBC HardCoded Check : The timeout setting in the JDBC activity getBooks is assigned a harcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false

Once running the analysis from Maven, you can run subsequent analysis incase of code change from TIBCO BusinessStudio itself by right clicking the project and choosing SonarQube>Analyze



Description	Assignee	Resource	New issue
tems)			
ardCoded Check : The max rows setting in the JDBC activity JDBCQuery is assigned a hardcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
ardCoded Check : The max rows setting in the JDBC activity getAllEvents is assigned a hardcoded value. It should be defined as Process property or Module property.		EventsDB.bwp	false
ardCoded Check : The max rows setting in the JDBC activity getBookByISBN is assigned a hardcoded value. It should be defined as Process property or Module prope		BooksDB.bwp	false
ardCoded Check : The max rows setting in the JDBC activity getBooks is assigned a hardcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
ardCoded Check : The max rows setting in the JDBC activity getEventByID is assigned a hardcoded value. It should be defined as Process property or Module property		EventsDB.bwp	false
e JDBC activity getEventByISBN is assigned a hardcoded value. It should be defined as Process property or Module prope		EventsDB.bwp	false
e JDBC activity getRecentBooks is assigned a hardcoded value. It should be defined as Process property or Module prope		BooksDB.bwp	false
JDBC activity JDBCQuery is assigned a harcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false
JDBC activity addBooks is assigned a harcoded value. It should be defined as Process property or Module property.		BooksDB.bwp	false

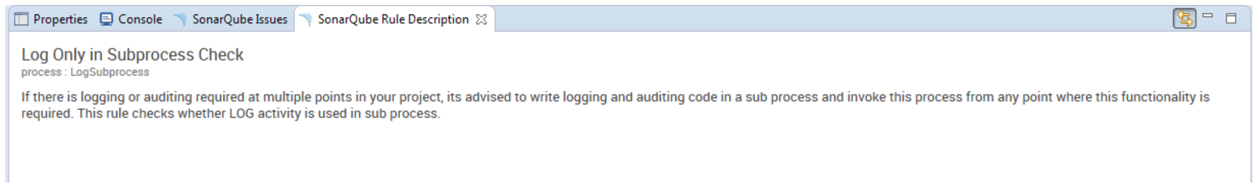
You can also view the Rules Description with associated violation by right clicking on the violation and choose Rule Description.



6 items

Description		Assignee	Resource	New issue
Major (5 items)				
Log Only in Subprocess Check : The Log act	sub process. Events.bwp is not a subprocess.		Events.bwp	false
Log Only in Subprocess Check : The Log act	sub process. Events.bwp is not a subprocess.		Events.bwp	false
Log Only in Subprocess Check : The Log act	sub process. Events.bwp is not a subprocess.		Events.bwp	false
Log Only in Subprocess Check : The Log act	sub process. Events.bwp is not a subprocess.		Events.bwp	false
Log Only in Subprocess Check : The Log act	ub process. Events.bwp is not a subprocess.		Events.bwp	false
Minor (1 item)				
No Process Description Check : No descripti			Events.bwp	false

:rably used in a sub process. Events.bwp is not a subpro



Log Only in Subprocess Check

process : LogSubprocess

If there is logging or auditing required at multiple points in your project, its advised to write logging and auditing code in a sub process and invoke this process from any point where this functionality is required. This rule checks whether LOG activity is used in sub process.

NOTE - The SonarQube Eclipse plugin does not work with SonarQube 5.2+. It is replaced by SonarLint for Eclipse. SonarLint presently doesn't support analysis on custom languages, but once it starts supporting it is advisable to use SonarLint instead of SonarQube Eclipse plugin.

K Useful links

http://docs.codehaus.org/display/SONAR/Documentation