Methods & Tools
Software Development Magazine - Project Management, Programming, Software Testing

Agile Maturity Assessment
Real-Ti
Organiz
and Ins
Needec

Software Testing Magazine

The Scrum Expert

Subscribe

Sonar - Open Source Project and Code Quality Monitoring

Olivier Gaudin, Freddy Mallet, SonarSource, http://www.sonarsource.co

What is Sonar ?

Sonar is an open source platform used by development teams to manage s code quality. Sonar has been developed with a main objective in mind: make c uality management accessible to everyone with minimal effort.

As such, Sonar provides code analyzers, reporting tools, defects hunting modules and TimeMachine as core functionality. But it also embarks a plugin mechanism enabling the community to extend the functionality (more than 35 plugins available), making Sonar the one-stop-shop for source code quality by addressing not only developers but also managers needs.

In terms of languages, Sonar support analysis of Java in the core, but also of Flex (ActionScript 3), PHP, PL/SQL and Cobol through plugins (Open Source or commercial) as the reporting engine is language agnostic.

Since version 2.0, Sonar enables to cover quality on 7 axes and so to report on:

- Duplicated code
- Coding standards
- Unit tests
- Complex code
- Potential bugs
- Comments
- Design and architecture

Sonar can be used for one-off audits, but has been designed to support global continuous improvement strategy on code quality in a company and therefore can be used as a shared central repository for quality management.

Web site: http://sonar.codehaus.org

Product Version: Sonar 2.0

Licence: LGPL V3

Support: http://sonar.codehaus.org/support/

Plugins: http://sonar-plugins.codehaus.org

Commercial plugins: http://www.sonarsource.com/plugins/

Why should you manage source code quality?

A well-written program is a program where the cost of implementing a feature is constant throughout the program's lifetime -- Itay Maman

As a quick intro, this is the best definition of source code quality I could find. It gets even stronger when put the other way around: *a badly written program is a program where the cost of implementing a feature grows throughout time.*

That sounds bad, doesn't it?

We have all seen situations where a new project starts whose objective is to develop from scratch an application in a leading-edge technology. Everything goes very fast; first, second, third release and then all of a sudden, the team's velocity starts to decrease. Fourth release is postponed for the third time, fixing something breaks something else…

What is happening here? Given the symptoms, we can make an assumption that the team is suffering from technical debt amongst other things and that stakeholders are not aware of it and can therefore not deal with it.

But this will most probably get resolved as the project is new, has visibility ar efore somebody is going to take care of it (at least we can hope so).

But this example was only a starter, as we, IT people, do not work most of e on applications whose development started less than 6 months ago! Our job is mai de of upgrades to existing applications. That is where most of the money is spent i here there is less visibility, where often there is a big yearly envelop to do as much can, where there are people who are key because they are the only ones able to und d the code, where we have no idea how long a change is going to take, where regr s are frequent and people are scared to make changes. And there is basically n ntion whatsoever from the business on that, just do it!

Managing source code quality is all about optimizing ROI as it is going to give y bility and therefore more control on:

1. how hard maintenance is going to be, what can we expect
2. the fact that things are not getting worse
3. the fact that some attention should be given to critical part of the system, to increase for example coverage by unit tests, suppress cycles, remove duplications

Further more it gives a backup for developers to raise their hand when they believe some refactoring is required that would add a bit to a change but would have good ROI.

How to manage source code quality?

There are seven technical axes that should be looked at when doing source code analysis of a project and Sonar is able to support the management of all seven. In the Sonar team, we like to call them the 7 deadly sins of the developer:

- non respect of coding standards and best practices
- lacking comments in the source code, especially in public APIs
- having duplicated lines of code
- having complex component or/and a bad distribution of complexity amongst components
- having no or low code coverage by unit tests, especially in complex part of the program
- leaving potential bugs
- having a spaghetti design (package cycles…)

The first step when doing source code quality management is really to define which of those axes are important to you and to what extend. Then based on the current situation, a plan should be established to reach the target level (that might be simply to keep a high level of quality). Very important is to start small and go bigger when it gets fully adopted by the whole development team.

Now, let's have a look at how to use Sonar in this approach.

Quality profiles

Sonar enables to manage multiple quality profiles in order to adapt the required level to the type of project (only support, new project, critical application, technical lib…). Managing a profile consists of:

activate / deactivate / weight coding rules

define thresholds on metrics for automatic alerting

define project / profile association

Dashboards

Sonar contains 2 dashboards that give the big picture to get hints where there might be issues and to compare projects:

1. a consolidated view that shows all projects
2. a project dashboard is also available at modules and packages level

SHARES

To confirm that what seems to be an issue is really an issue, Sonar offers a hunting toolset that enables to go from overview to smallest details:

- drill down on every measure displayed to see what is behind
- classes clouds to find less covered classes by unit tests
- hotspots to have on a page the most and the least files
- and a multi-entry (duplication, coverage, violations, tests success…) source r to confirm the findings made with the hunting tools

TimeMachine

No doubt that knowing where an application stands is very important. But more important is to know and understand its evolution. Indeed, what is it worth tc that there is 20% of code coverage by unit tests? Is it good or bad? Is the answer diff f two months ago it was 15% or 25%? TimeMachine enables to watch the evolution an y the past, especially as it records versions of the project

How does Sonar work?

Sonar is made of a fairly simple and flexible architecture that consists of three co nts:

- A set of source code analyzers that are grouped in a Maven plugin and are triggered on demand. The analyzers use configuration stored in the database. Although Sonar relies on Maven to run analysis, it is capable to analyze Maven and non-Maven projects.
- A database to not only persist the results of the analysis, the projects and global configuration but also to keep historical analysis for TimeMachine. 5 database engines are currently supported : Oracle, MySQL, Derby (demo only), PostgreSQL and MS SQLServer
- A web reporting tool to display code quality dashboards on projects, hunt for defects, check TimeMachine and to configure analysis.

As part of its analyzers, Sonar core embarks best of breed tools to find coding rules violations (PMD, Checkstyle), detect potential bugs (Findbugs) and measure coverage by unit tests (Cobertura, Clover). But what makes Sonar truly unique is Squid, its own code analyzer that not only parses source code but also byte code and mixes the results.

Since analysis is run through a Maven plugin, Sonar can be launched easily in "Continuous Integration" environments.

Use case on Apache commons-collection project

A pre-requisite to run Sonar is to have Java and Maven installed on the box. Once this is the case, you can run Sonar in 5 simple steps:

1. Download the distribution from http://sonar.codehaus.org/downloads/ and unzip it

2. Open a console and start the server:

> $SONAR_HOME\bin\windows-x86-32\StartSonar.bat on windows

> $SONAR_HOME/bin/[OS]/sonar.sh on other platforms

3. Open a console where you want to checkout the source and run:

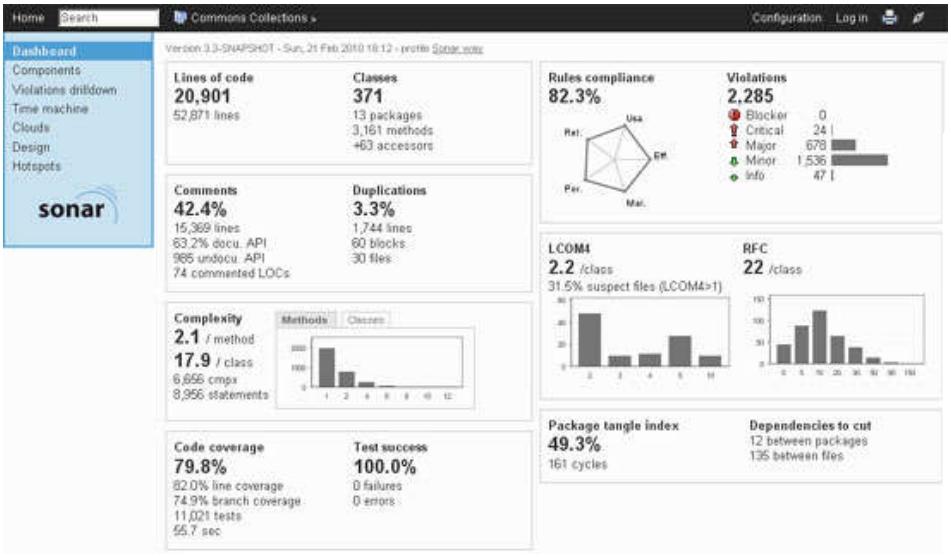
svn co http://svn.apache.org/viewvc/commons/proper/collections/trunk/.

4. Run mvn install sonar:sonar in the same directory

5. Browse http://localhost:9000

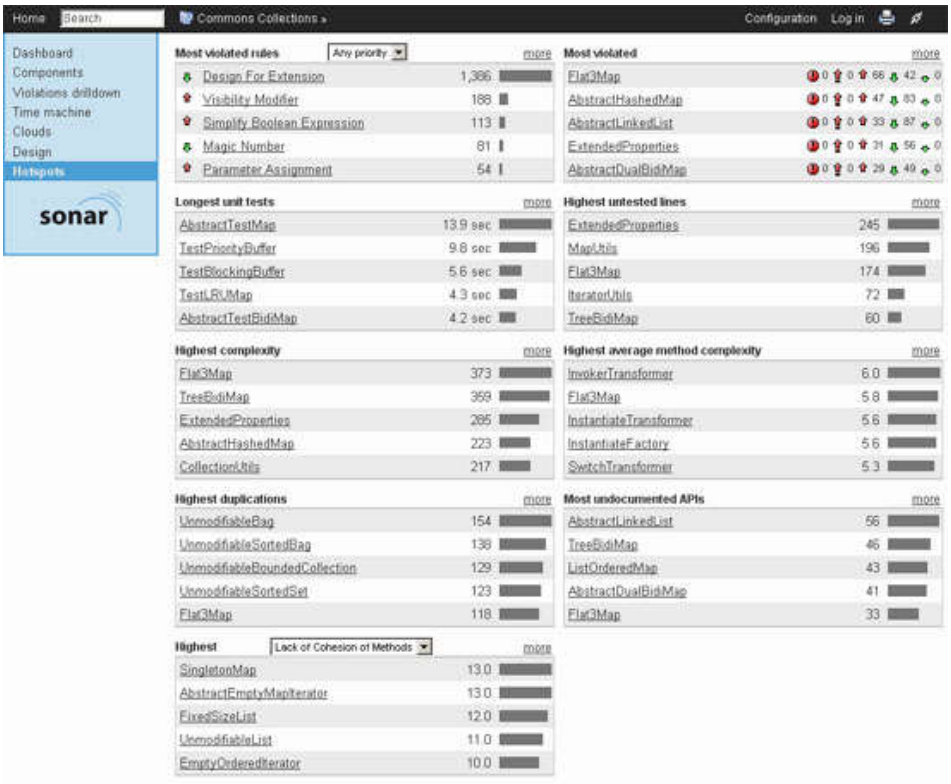
The home page of the application shows the list of projects under quality control with a few configurable metrics.

SHARES

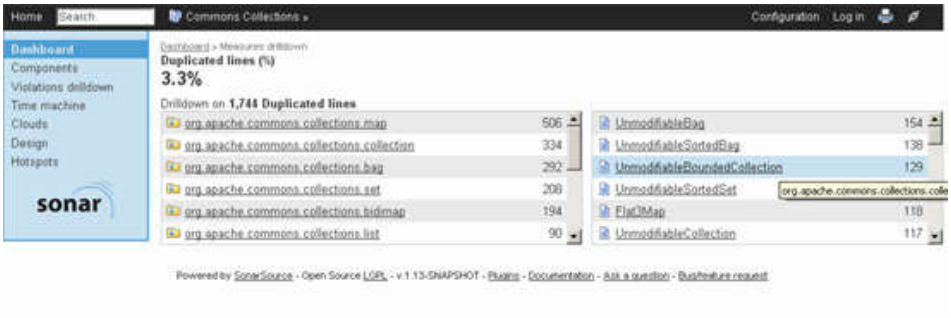


From there you have access to a series of hunting tools amongst which:

The hotspot to find out about the files that have "the most" or "the least" ...

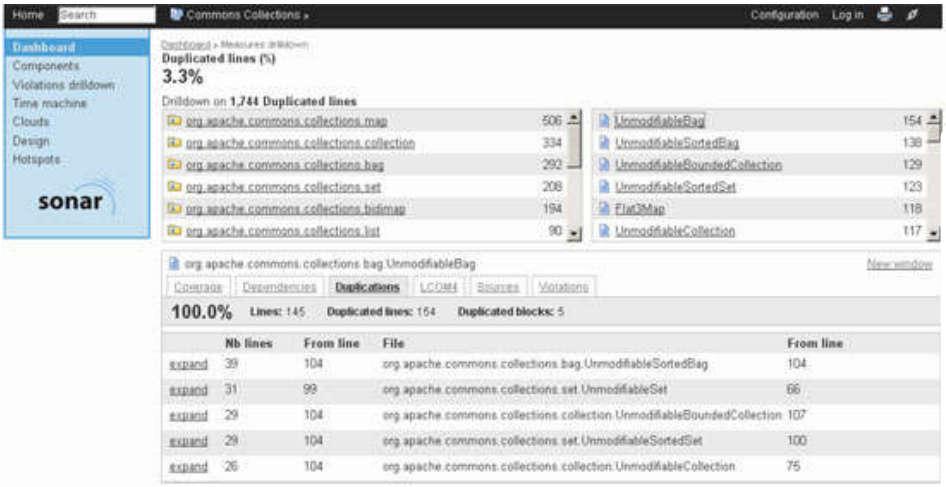


But also, any metric in the dashboard is clickable to jump to behind the scene and get a view of the metric by underlying component



Each hunting tool eventually brings the hunter to the source code where the preys are highlighted

SHARES



The Sonar Ecosystem

There is a very dynamic ecosystem around Sonar

- An active community made of 300+ people on the user mailing list and 150+ people on the development mailing list
- 35+ plugins on the forge (http://docs.codehaus.org/display/SONAR/Sonar+Plugin+Library/) that are divided into four categories
- Integration with external tools such as Jira, Hudson, Bamboo, GateIn, AnthillPro, Crowd
- Direct extension of core functionality by adding new behavior, calculate advanced metrics or consolidate projects, add new metrics
- Add coverage of languages such as PL/SQL, ActionScript3
- Integration with IDEs to get defects information on the code when it is edited
- 3,000+ downloads per month
- A core development team led by SonarSource (http://www.sonarsource.com)

Conclusion

After the massive adoption of continuous integration engines and Tests Driven Development practices, managing source quality looks like the natural next step for development teams in their effort of industrialization. Sonar enables to reach this objective with few efforts and with fun.

In 2010, the Sonar platform is going to continue to evolve, the main axes of development being covering new languages and improving integration with IDEs.

To stay connected, you can follow the Sonar blog : https://blog.sonarsource.com/category/blog/.

More Agile Content

Scrum Expert for Agile Project Management and Software Development

Software Testing Magazine

Agile Tutorials and Videos

Click here to view the complete list of tools reviews

This article was originally published in the Spring 2010 issue of Methods & Tools

deliver:Agile, April 30 - May 2 2018, Austin, USA - Explore Agile Engineering practices.

Agile2018, August 6-10 2018, San Diego, USA - The largest North American Agile conference

Discover the best available **Open Source Project Management Tools**

Browse a selected list of upcoming **Software Development Conferences**

SHARES

Follow Methods & Tools on 

SHARES