12

December 2010

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Open Source Tools

© diego cervo - Fotolia.com

© karaboux - Fotolia.com

Take a daily photo of your development process!

Improve your development process with a Continuous Integration System

by Antonio Robres Turón

One of the major problems in the software development process among the development teams is the integration phase. In this phase the different software components developed by different persons must be integrated, and several problems are often detected.

There are several solutions to solve these problems. The most common solution is performing the integration in phases (within the software development process), another solution is starting the integration at the beginning of software development.

Now imagine you can obtain a built and integrated product every day. Imagine an automatic system that compiles and integrates the source code every day. Many of the integration problems would be solved quickly!

One possible solution is implementing a Continuous Integration System. These systems take the code from the CVS (Concurrent Version System) or repositories. They compile all the source code and finally integrate all the components of the project. There are many applications to implements this system; many of them are Open Source applications, such as Hudson.

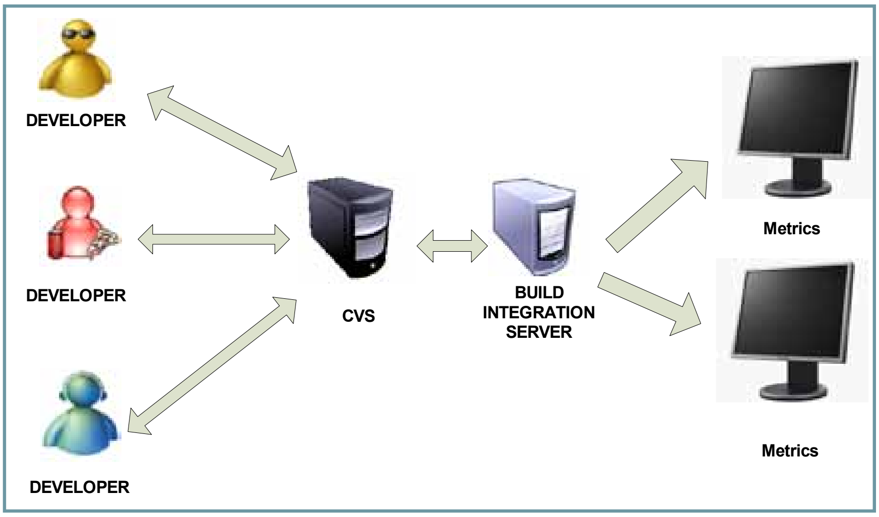
In this article we will share an overview of open source Continuous Integration System called Hudson. Moreover, we will obtain daily metrics to help to improve the development process and the quality of the product.

How it works?

The Continuous Integration System has a simple and intuitive process divided in four phases:

1. The Continuous Integration server connects with the repository and downloads the source code created by the development team and the unit tests.
2. All the source code of the project is compiled and integrated between them by the dependencies rules. The result is an integrated, built product.
3. The unit and integration tests are executed in the Continuous Integration server.
4. Quality analysis tools are executed to obtain the metrics.

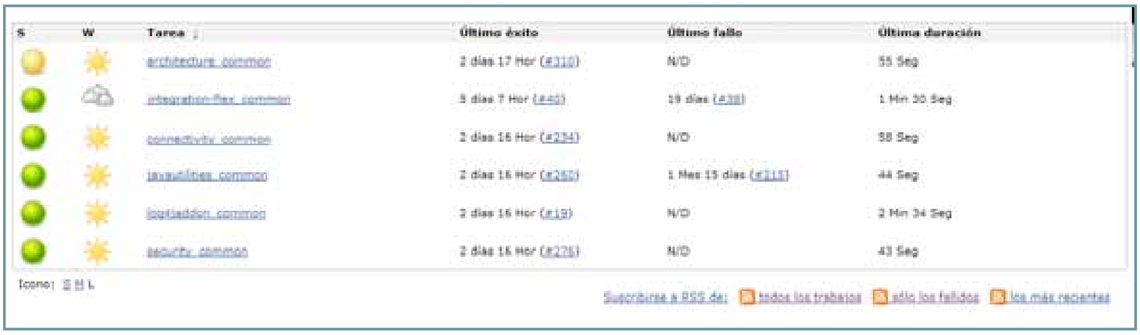
The architecture of the system is shown in the diagram below.



All the process can be automatic and scheduled to perform a periodic build (normally daily). Moreover, it can be executed manually to obtain an integrated version at any time. The process is easily configurable and flexible. It allows at any moment the selection of the components in order to compile or integrate and executes the unit and integrated tests.

Hudson provides a useful interface showing the result of every compilation and execution done with the historic results. Therefore it facilitates the analysis of every module build. The next picture shows the principal view of Hudson showing the different components, the state of the last execution (red, green, yellow or blue), the stability of the last executions (weather icon), the last correct build, the last fail and the duration of the last execution.

Now you can obtain every day a functional and integrated software to deliver to the testing team. However, we can also extend the Hudson capacities with quality plug-ins.



S	W	Tarea ↓	Último éxito	Último fallo	Última duración
●	☀	architecture_common	2 días 17 Hor (#310)	N/D	55 Seg
●	☁	integration-flex_common	5 días 7 Hor (#40)	19 días (#38)	1 Min 30 Seg
●	☀	connectivity_common	2 días 16 Hor (#234)	N/D	58 Seg
●	☀	javautilities_common	2 días 16 Hor (#260)	1 Mes 15 días (#215)	44 Seg
●	☀	logtaddon_common	2 días 16 Hor (#19)	N/D	2 Min 34 Seg
●	☀	security_common	2 días 16 Hor (#276)	N/D	43 Seg

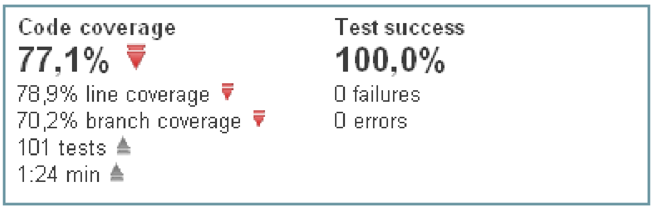
Icono: S M L

Suscribirse a RSS de: todos los trabajos sólo los fallidos los más recientes

Obtain a daily quality metrics

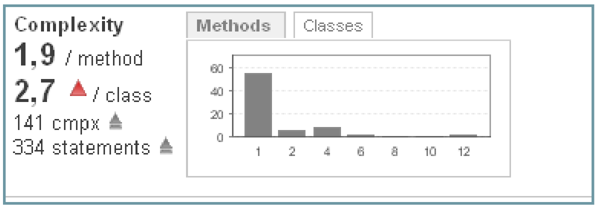
One of the advantages of Hudson is the flexibility to load different plug-ins to execute several tools. Many of them are able to obtain quality metrics depending on the code language used.

As discussed in the previous section, Hudson allows us to execute the unit tests automatically every day. Furthermore, it allows the plug-in execution to measure the code coverage with several open source plug-ins like "Cobertura" or "EmmaECL". These plug-ins are executed with the unit test and produce several metrics such as line coverage, branch coverage or the cyclomatic complexity.



Code coverage
77,1% ▼
78,9% line coverage ▼
70,2% branch coverage ▼
101 tests ▲
1:24 min ▲

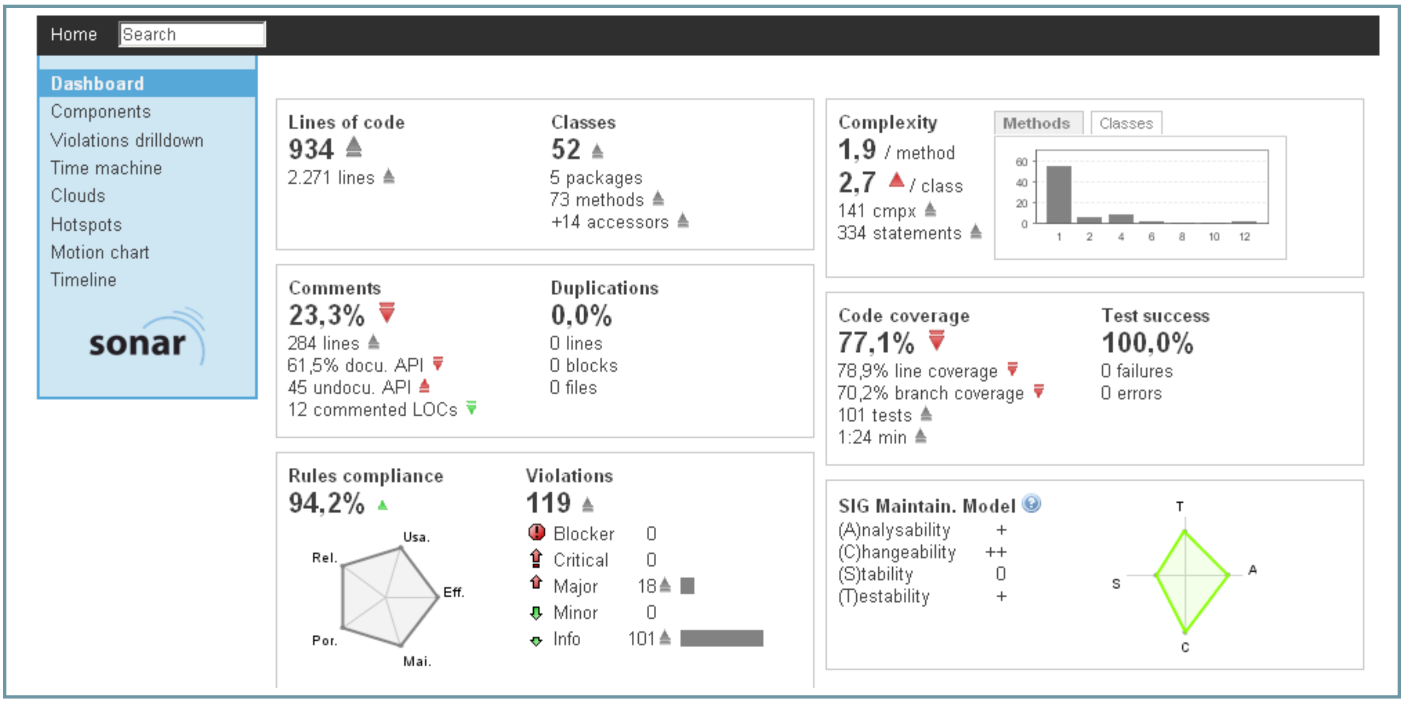
Test success
100,0%
0 failures
0 errors



Complexity
1,9 / method
2,7 ▲ / class
141 cmpx ▲
334 statements ▲

Other features you can add in the Continuous Integration System are the code static analysis such as PMD, FindBugs or Checkstyle. These tools provide an analysis of the source code without executing it by making some checks against predefined rules. The result is a collection of metrics which can be used to obtain a quality statement about the source code.

To facilitate the managing of all the metrics and results obtained through the Continuous Integrated System, the Sonar application can be used. Sonar is an open platform to manage code quality and can be connected with Hudson through a plug-in to execute all the code analysis tools and collect the results. The next picture shows the Sonar dashboard with a usable and intuitive interface.



Lines of code
934 ▲
2.271 lines ▲

Classes
52 ▲
5 packages
73 methods ▲
+14 accessors ▲

Complexity
1,9 / method
2,7 ▲ / class
141 cmpx ▲
334 statements ▲

Comments
23,3% ▼
284 lines ▲
61,5% docu. API ▼
45 undocu. API ▲
12 commented LOCs ▼

Duplications
0,0%
0 lines
0 blocks
0 files

Code coverage
77,1% ▼
78,9% line coverage ▼
70,2% branch coverage ▼
101 tests ▲
1:24 min ▲

Test success
100,0%
0 failures
0 errors

Rules compliance
94,2% ▲

Violations
119 ▲

Blocker	0	
Critical	0	
Major	18 ▲	▮
Minor	0	
Info	101 ▲	▬▬▬

SIG Maintain. Model

(A)nalysability	+
(C)hangeability	++
(S)tability	0
(T)estability	+

Thus you can obtain every day the following metrics:

- Lines of code, classes, package and method numbers
- Comment lines, documented API
- Dead code
- Duplicated code
- Rules compliance divided in five categories (efficiency, maintainability, portability, reliability and usability) and divided according to their critical nature.
- Test unitary execution results
- Line coverage and branch coverage.
- Class and method complexity
- SIG Maintain Model divided in four categories (analyzability, changeability, stability and testability.
- Quality Index. This index is composed of different metrics (rules violations, complexity, coverage and code style).

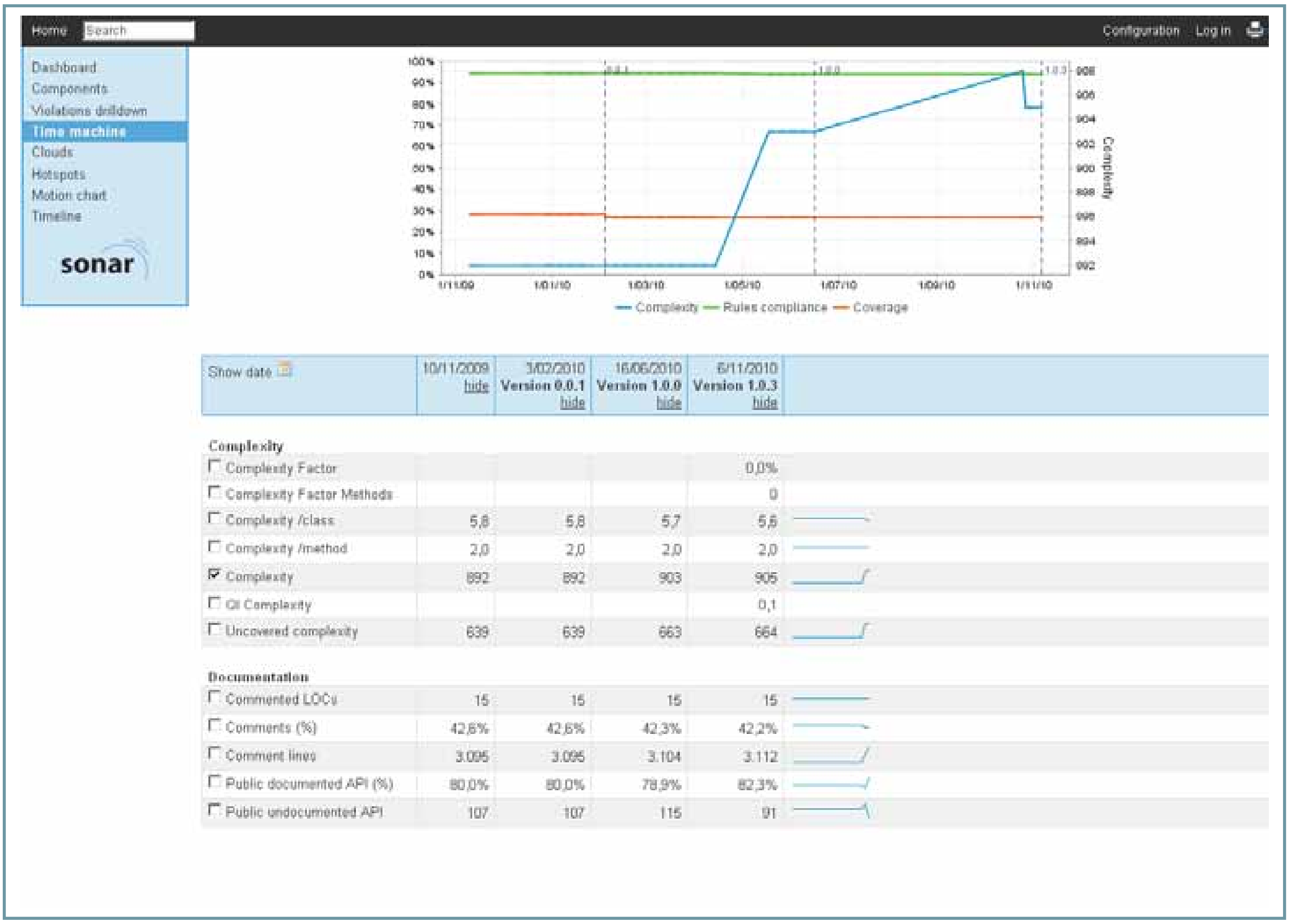
Sonar plug-in also obtains different trends of every metric to analyze the evolution of the software quality. Sonar helps you replay the past and show you how quality metrics evolve in time. Therefore, it makes the actions taken easier in order to improve the quality of the source code.

Why use Hudson + Sonar?

There are different benefits to using the Continuous Integrated System:

1. Early detection of compilation and integration errors because the source code is integrated and built every day.
2. Automatic execution of the unit and integration test. This execution can detect every-day errors introduced in the code and the development department can fix these errors in an early development phase.
3. Automatic execution of the code analyzers. This feature takes a daily photo of the development process state and helps the project manager to take actions to improve the development process and the quality of the product.

All these benefits can be provided by the combination of Hudson + Sonar providing a flexible system which allows all the metrics and rules configurations and provides a usable and intuitive interface to all users.



Show date	10/11/2009 hide	3/02/2010 **Version 0.0.1** hide	16/06/2010 **Version 1.0.0** hide	6/11/2010 **Version 1.0.3** hide
Complexity				
□ Complexity Factor				0,0%
□ Complexity Factor Methods				0
□ Complexity /class	5,8	5,8	5,7	5,6
□ Complexity /method	2,0	2,0	2,0	2,0
☑ Complexity	892	892	903	905
□ QI Complexity				0,1
□ Uncovered complexity	639	639	663	664
Documentation				
□ Commented LOCs	15	15	15	15
□ Comments (%)	42,6%	42,6%	42,3%	42,2%
□ Comment lines	3.095	3.095	3.104	3.112
□ Public documented API (%)	80,0%	80,0%	78,9%	82,3%
□ Public undocumented API	107	107	115	91

Biography

Antonio Robres is a Test Engineer at Diagnostic Grifols in Barcelona, Spain. He studied Telecomunications Science at Universidad Politéctica de Cataluña in Spain and has a Master in telecommunication administration and is an ISTQB© certified Tester Foundation Level. He has been working for 5 years in the field of Software Testing and Quality engineering in companies such as Telefonica, Gas Natural and Grifols. His work focuses on the design and execution of several testing projects, mainly in embedded systems and web applications. He is also involved in the design and development of test automation projects with Open Source tools. He was a speaker in the last QA&TEST edition explaining all the testing and QA structure of Diagnostic Grifols in detail.

© Wolfgang Zintl - Fotolia.com