



**LISTEN AND LOOK
AT YOUR PHP CODE!**

Gabriele Santini
Forum AFUP 2010

Gabriele Santini



- Architect/Consultant at SQLI
- Contributor to PHP_CodeSniffer
 - So expect a special focus on this...
- Sonar PHP Plugin
 - I have to show you!
- Ex-mathematician :
 - love business modelling
 - love architectures
 - love quality assurance



Static Analysis



- All you can say about your program without actually execute the code
 - The rest is also interesting, let's talk about it another time !
- Examples ?
 - Syntax check, coding style, anti-patterns, metrics, OO design analysis, ...
- *What PHP does before executing the code ?*

Levels of analysis



- Lexical analysis
 - Read sources linearly searching for known patterns
 - Convert them to a sequence of tokens

Levels of analysis



- Lexical analysis
 - Read sources linearly searching for known patterns
 - Convert them to a sequence of tokens
- Syntactic Analysis
 - Parse the tokens to find their logical structure

Levels of analysis



- Lexical analysis
 - Read sources linearly searching for known patterns
 - Convert them to a sequence of tokens
- Syntactic Analysis
 - Parse the tokens to find their logical structure
- Opcode (Bytecode) Generation
 - Generates an intermediary code that the Zend Engine will be able to execute

Lexical Analysis



- Tokenizer

```
<?php
if (1 < 2) {
    echo "Hello";
}
?>
```

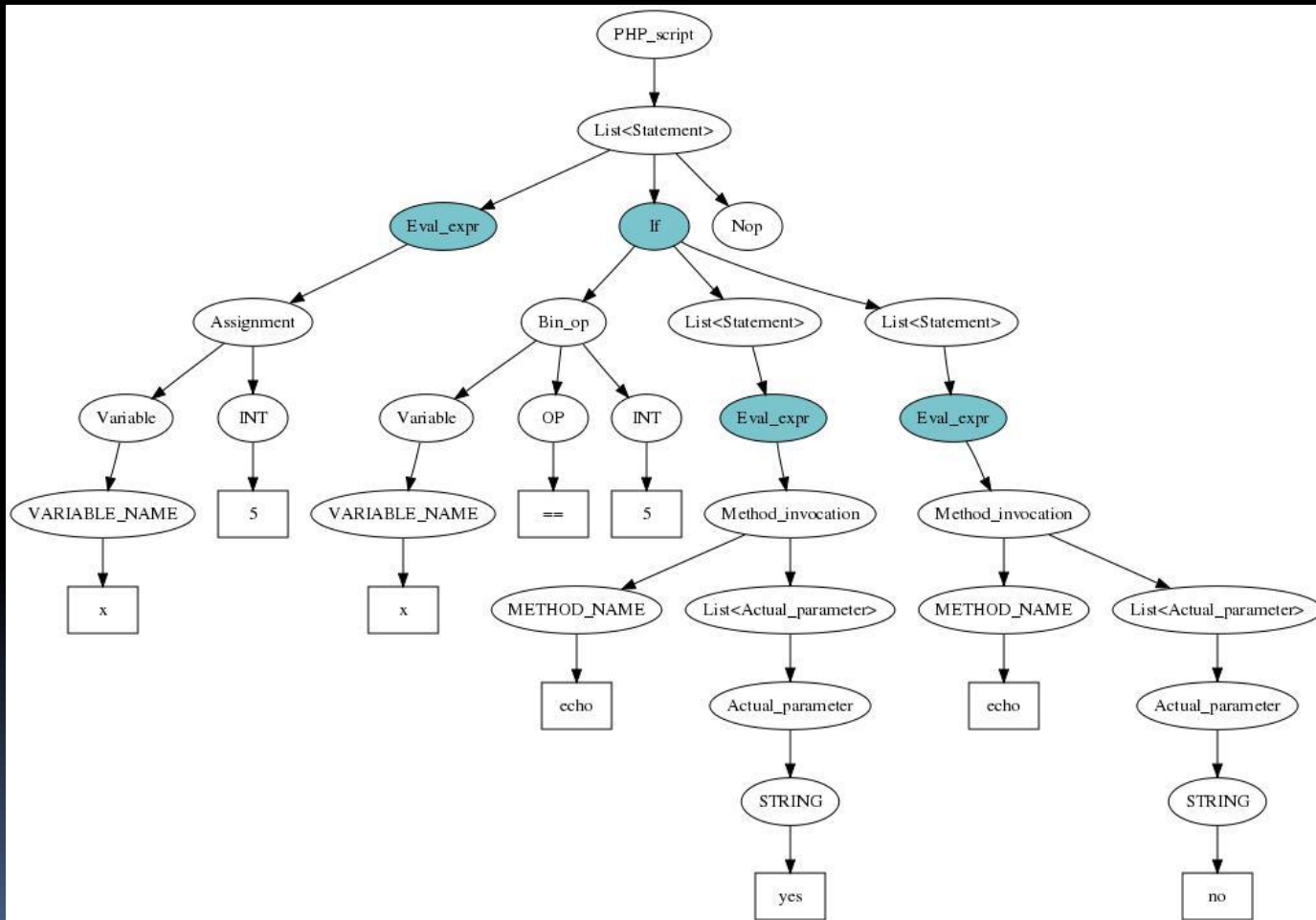
```
1  <?php  T_OPEN_TAG
2  if     T_IF
2  T_WHITESPACE
   (
2  1      T_LNUMBER
2  T_WHITESPACE
   <
2  T_WHITESPACE
2  2      T_LNUMBER
   )
2  T_WHITESPACE
   {
2  T_WHITESPACE
3  echo   T_ECHO
3  T_WHITESPACE
3  "Hello" T_CONSTANT_ENCAPSED_STRING
   ;
3  T_WHITESPACE
   }
4  T_WHITESPACE
5  ?>    T_CLOSE_TAG
```

Syntactic Analysis



- Produces an AST
 - Abstract Syntax Tree
 - Decompose code in a tree-like form
 - Can be executed once a context is given
 - Used in compilers

Syntactic Analysis



Opcodes Generation



- Mystery tool...

```
<?php
if (1 < 2) {
    echo "Hello";
}
?>
```

line	#	*	op	return	operands
2	0	>	EXT_STMT		
	1		IS_SMALLER	~0	1, 2
	2	>	JMPZ		~0, ->6
3	3	>	EXT_STMT		
	4		ECHO		'Hello'
4	5	>	JMP		->6
6	6	>	EXT_STMT		
	7	>	RETURN		1



GIVE US THE TOOLS !

PHP_CodeSniffer



- By Greg Sherwood
 - PEAR library
 - Venerable project
- Code Style
 - But also a lot more
- Works at lexical analysis level
 - Heavily use the tokenizer extension

PHP_CodeSniffer



- Hands on

```
blacksun@blacksun-laptop:/demo$ phpcs test_phpcs.php
FILE: /media/sdal/tests/test_phpcs.php
-----
FOUND 7 ERROR(S) AND 0 WARNING(S) AFFECTING 7 LINE(S)
-----
 2 | ERROR | Missing file doc comment
 3 | ERROR | Multi-line function call not indented correctly; expected 4
   |       | spaces but found 0
 7 | ERROR | Closing parenthesis of a multi-line function call must be on a
   |       | line by itself
 9 | ERROR | Missing function doc comment
20 | ERROR | Line indented incorrectly; expected 4 spaces, found 7
24 | ERROR | Line indented incorrectly; expected 4 spaces, found 7
28 | ERROR | Closing brace indented incorrectly; expected 0 spaces, found 4
-----
```

PHP_CodeSniffer



- Sniffs
 - Classes that detect Violations
 - One or more type per class
 - Grouped in folders by subject:
 - Commenting, Formatting, WhiteSpace
 - Files, ControlStructures, Strings
 - Functions, Classes, NamingConventions
 - CodeAnalysis, Metrics
 - You can create your own!

PHP_CodeSniffer



- Standards
 - Sets of Sniffs that define your coding style
 - Installed :
 - PEAR,
 - Generic,
 - Zend*,
 - Squiz, MySource
 - PHPCS

PHP_CodeSniffer 1.3



- Rulesets XML!

```
<ruleset name="MyPEAR">
  <description>A variation of the PEAR coding standard.</description>
  <!-- Include some additional sniffs from the Generic standard -->
  <rule ref="Generic.Functions.FunctionCallArgumentSpacing"/>
    <message>Please review spacing in function '%s' </message>
  </rule>
  <rule ref="Generic.NamingConventions.UpperCaseConstantName"/>
  <!-- Lines can be 90 chars long, but never show errors -->
  <rule ref="Generic.Files.LineLength">
    <properties>
      <property name="lineLimit" value="90"/>
      <property name="absoluteLineLimit" value="0"/>
    </properties>
  </rule>
  <!-- Not so important for us -->
  <rule ref="Generic.PHP.DisallowShortOpenTag">
    <severity>2</severity>
  </rule>
</ruleset>
```


Inside PHP_CodeSniffer



- Sniff class main methods
 - **register()**
 - Make the sniff a listener for the declared tokens
 - **process(\$phpcsFile, \$stackPtr)**
 - Called by the file during parsing when a declared token is found
- File
 - Represents a parsed file
 - Holds the tokens structure and offers convenience methods

Inside PHP_CodeSniffer



- Life of a Sniff
 - DisallowMultipleStatementsSniff

```
<?php
echo $y;
$x = 10; echo $y;

for ($i = 1; $i < $length; $i++) {
    echo 'x';
}
echo $x;
$y = 2;;
$this->wizardid = 10; $this->paint(); echo 'x';
?>
```

Inside PHP_CodeSniffer



- Life of a Sniff
 - DisallowMultipleStatementsSniff

```
public function register()  
{  
    return array(T_SEMICOLON);  
}
```

```
public function process($phpcsFile, $stackPtr)  
{  
    [...]  
}
```

Inside PHP_CodeSniffer



- Life of a Sniff
 - DisallowMultipleStatementsSniff

```
<?php
echo $y;
$x = 10; echo $y;

for ($i = 1; $i < $length; $i++) {
    echo 'x';
}
echo $x;
$y = 2;;
$this->wizardid = 10; $this->paint(); echo 'x';
?>
```

Inside PHP_CodeSniffer



- Life of a Sniff
 - DisallowMultipleStatementsSniff

```
<?php
echo $y;
$x = 10; echo $y;

for ($i = 1; $i < $length; $i++) {
    echo 'x';
}
echo $x;
$y = 2;;
$this->wizardid = 10; $this->paint(); echo 'x';
?>
```

A diagram consisting of a green arrow pointing upwards from a green rectangular box labeled '\$stackPtr' to the '\$i' variable in the for loop condition of the PHP code above.

Inside PHP_CodeSniffer



- Life of a Sniff
 - DisallowMultipleStatementsSniff

```
public function register()  
{  
    return array(T_SEMICOLON);  
}
```

```
public function process($phpcsFile, $stackPtr)  
{  
    $tokens = $phpcsFile->getTokens();  
    $previous = $phpcsFile->findPrevious(...);  
  
    if ($previous === false) {  
        return;  
    }  
  
    // Continue =>
```

Inside PHP_CodeSniffer



- Life of a Sniff (2)

```
// Ignore multiple statements in a FOR condition.
// First some gym for nested parenthesis
[...]
if ($tokens[$owner]['code'] === T_FOR) {
    return;
}
[...]
// If the previous semicolon is on the same line we add an error
// to this file
if ($tokens[$previous]['line'] === $tokens[$stackPtr]['line']) {
    $error = 'Each PHP statement must be on a line by itself';
    $phpcsFile->addError($error, $stackPtr);
    return;
}
} //end process()
```

PHP_CodeSniffer



- At SQLI we have some framework standards
 - Zend Framework
 - Based on Thomas Weidner work
 - Symfony
 - In collaboration with Fabien Potencier
 - Waiting for a serious release after 1.3 release

PHP_CodeSniffer



- At SQLI we have some framework standards
 - Zend Framework
 - Based on Thomas Weidner work
 - Symfony
 - In collaboration with Fabien Potencier
 - Waiting for a serious release after 1.3 release
- That's nice, but...
- Where are the standards for the other tools ?
 - I'd expect a Drupal, Wordpress, Cake official standard

PHP_CodeSniffer



- How far a standard can go in detection ?

PHP_CodeSniffer



- How far a standard can go in detection ?
- Interestingly far for generic PHP Code

PHP_CodeSniffer



- How far a standard can go in detection ?
- Interestingly far for generic PHP Code
- Very far if you know your tool's structure
 - Imagine for example forcing PHP alternative syntax in Symfony views...
 - Or checking for escaping in Zend Views !

PHP_Depend



- By Manuel Pichler
- Functional port of JDepend
 - OO design analysis
 - Metrics visualisation
 - Dependency analyzer
- Works at the syntactic analysis level

PHP_Depend



- How it works
- PHP_Depend first makes an AST off your code
 - A « personal » one, made by PHP objects
 - ASTComment, ASTClosure, ASTEvalExpression, ...
 - This is made by the Builder/Parser component
 - Using PHP Reflection

PHP_Depend

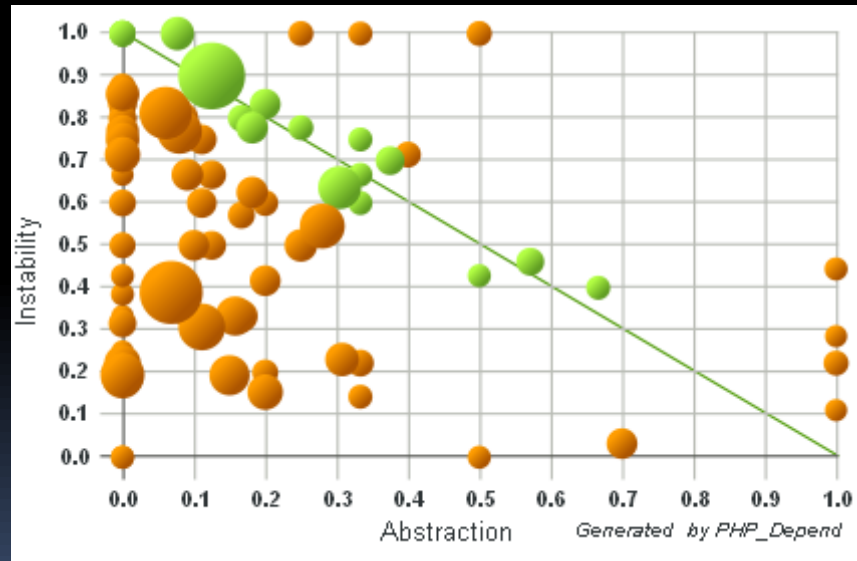


- How it works (2)
- Then PHP_Depend can answer questions by « visiting » the AST
 - Task of Metrics Analyzers, that extend AbstractVisitor
 - IOC, the visitor decides what to do according to AST Class : visitMethod, visitForStatement(), ...
 - Analyzers can fire listeners during analyze() call
 - To get ongoing informations about the visit process

PHP_Depend



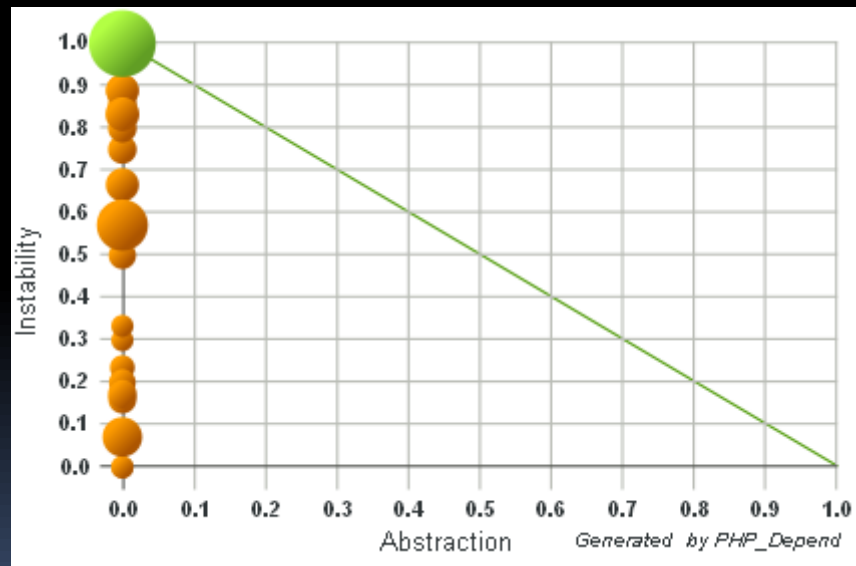
- What it gives:
 - The Abstraction/Instability graph



PHP_Depend



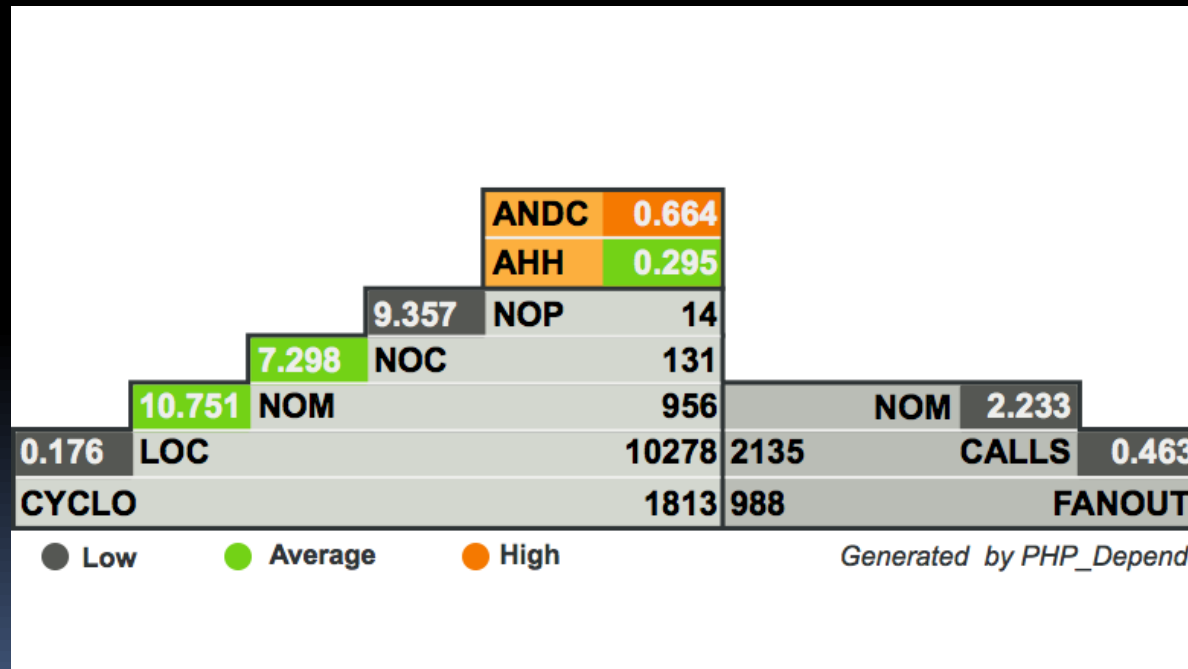
- What it gives:
 - The Abstraction/Instability graph



PHP_Depend



- What it gives:
 - The Pyramid !!



PHPMD



- By Manuel Pichler
- Detects rules violations
 - Analog to PHP_Codesniffer
- Works at syntactic analysis level
 - Actually on the same AST
 - Depends on PHP_Depend
 - Has rulesets !

PHPMD



- What it gives:
- Code Size Rules
 - complexities, lengths, too many, ...
- Design Rules
 - OO, exit, eval
- Naming Rules
 - Too short/long identifiers, old constructors, ...
- Unused Code Rules
 - Methods, members, parameters

phploc



- By Sebastian Bergmann
- Simple tool to give basic metrics
 - Fast, direct to the goal
- Works mostly on lexical level
 - But use bytekit for ELOC if it can

phpcpd



- By Sebastian Bergmann
- Simple tool to detect duplicated code
- Works at lexical analysis level
 - Use the tokenizer to minimize differences
 - Comments, whitespaces, ...
 - Takes a minimum number of lines and tokens
 - Encodes according to this
 - Uses an hash table to find duplicates

vld



- Vulcan Logic Disassembler
 - By Derick Rethans
- Works at bytecode level
 - Shows generated bytecodes
 - Calculates possible paths (CFG)
 - Find unreachable code
 - Could be used for code coverage path metrics

vld



- Output

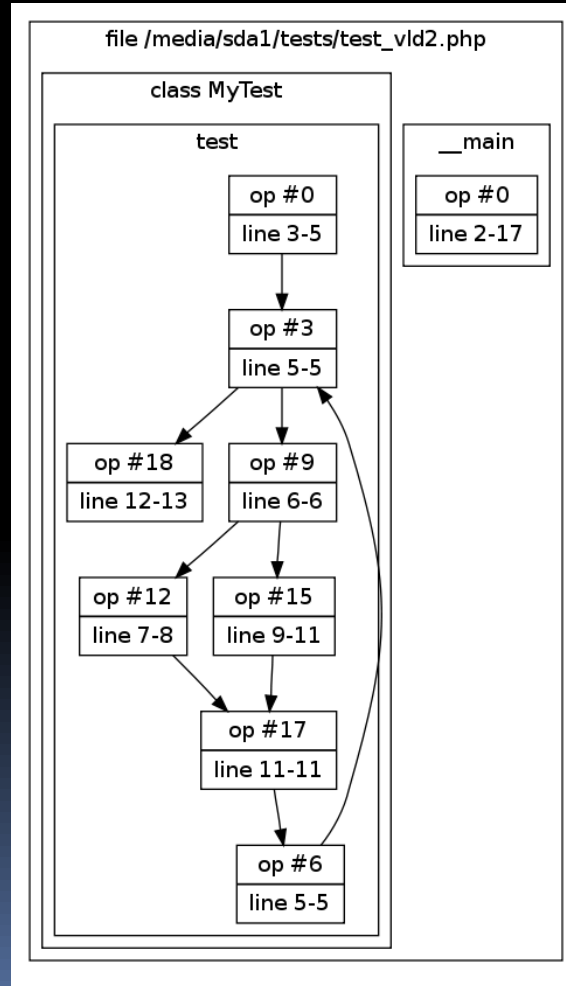
```
blacksun@blacksun-laptop:/media/sdal/tests$ php -dvld.active=1 -dvld.dump_paths=1 -dvld.save_paths=1 -dvld.save_dir=/media/sdal/tests test_vld2.php
filename:      /media/sdal/tests/test_vld2.php
function name: (null)
number of ops: 8
compiled vars: none
line   # * op
-----
   2   0 > EXT_STMT
       1   NOP
  16   2   EXT_STMT
       3   ZEND_INIT_STATIC_METHOD_CALL      'MyTest', 'test'
       4   EXT_FCALL_BEGIN
       5   DO_FCALL_BY_NAME                  0
       6   EXT_FCALL_END
  17   7   > RETURN                          1

branch: # 0; line: 2- 17; sop: 0; eop: 7
path #1: 0,
```


vld



- Output

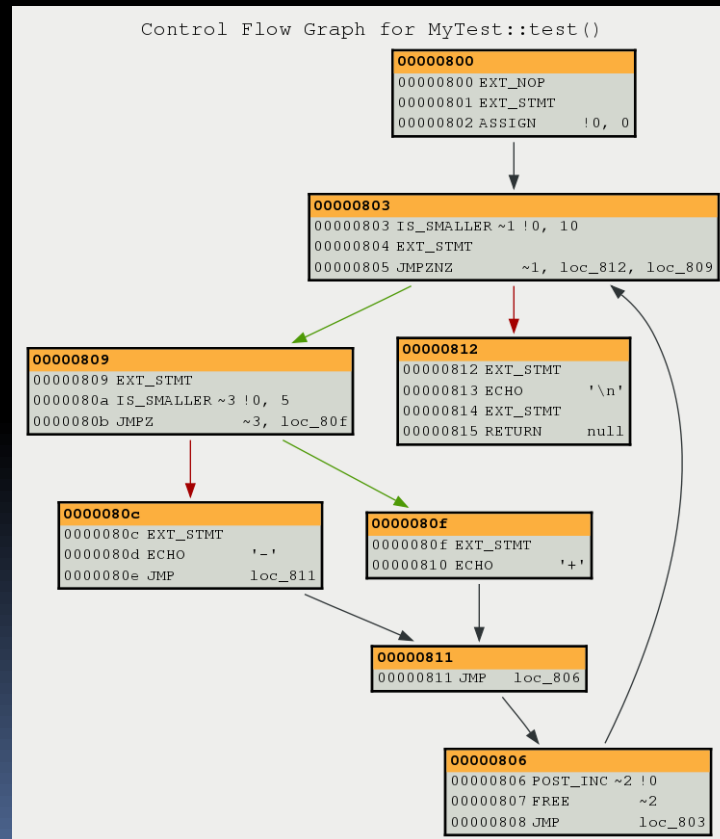


Bytekit



- By Stefan Esser (SektionEins)
- Works at ... bytecode level
 - Similar to vld
 - Exposes opcodes to a PHP array
 - `bytekit_disassemble_file($filename)`
- Can be used directly for a custom script

- CFG visualisation



Bytekit-cli



- By Sebastian Bergmann
- PHP Interface to use bytekit to spot violation rules
 - Initial state
- Implemented rules :
 - Check for disallowed opcodes (example eval, exit)
 - Check for direct output of variables
 - In svn, check for unescaped ZendView

Padawan



- By Florian Anderiasch
- Focus on anti-pattern detection
 - alpha (?)
- Works on syntactic analysis level
 - Based on PHC (PHP compiler)
 - Use an XML dump of the AST PHC generates
 - Makes xpath searches on it

Padawan



- Interesting approach
 - Rules are fairly simple to write
 - Already many interesting tests :
 - Empty constructs (if, else, try,..), unsafe typecasts, loop repeated calls, unused keys in foreach, ...
- PHC not easy to install
- Risk on PHC maintenance

Phantm



- By Etienne Kneuss
- Highly experimental
 - Severe limitation on PHP dynamic features
 - False positives
- Works on syntax analysis level
 - Based on Java tools (Jflex, CUP, Scala)
- Reports violations
 - Non top-level declarations, call-time pass-by-ref, nontrivial include calls, assign in conditional, ...
- Exploring Type Flow Analysis
 - Tries to infer types and check for type safety

Conclusion



- Use the right tool for the right job
 - Coding style is better analysed at the lexical level
 - OO design is better viewed after syntactic analyses
 - Unreachable code after bytecoding
- Contribute !
 - Plenty of things still to implement
 - Easy to have new ideas
 - At least use them (you should!) and give feedback

Restitution



- Once all this is collected what to do with it ?
- At least, show it in a suitable form
- At best, integrate this in your CI system

phpUnderControl

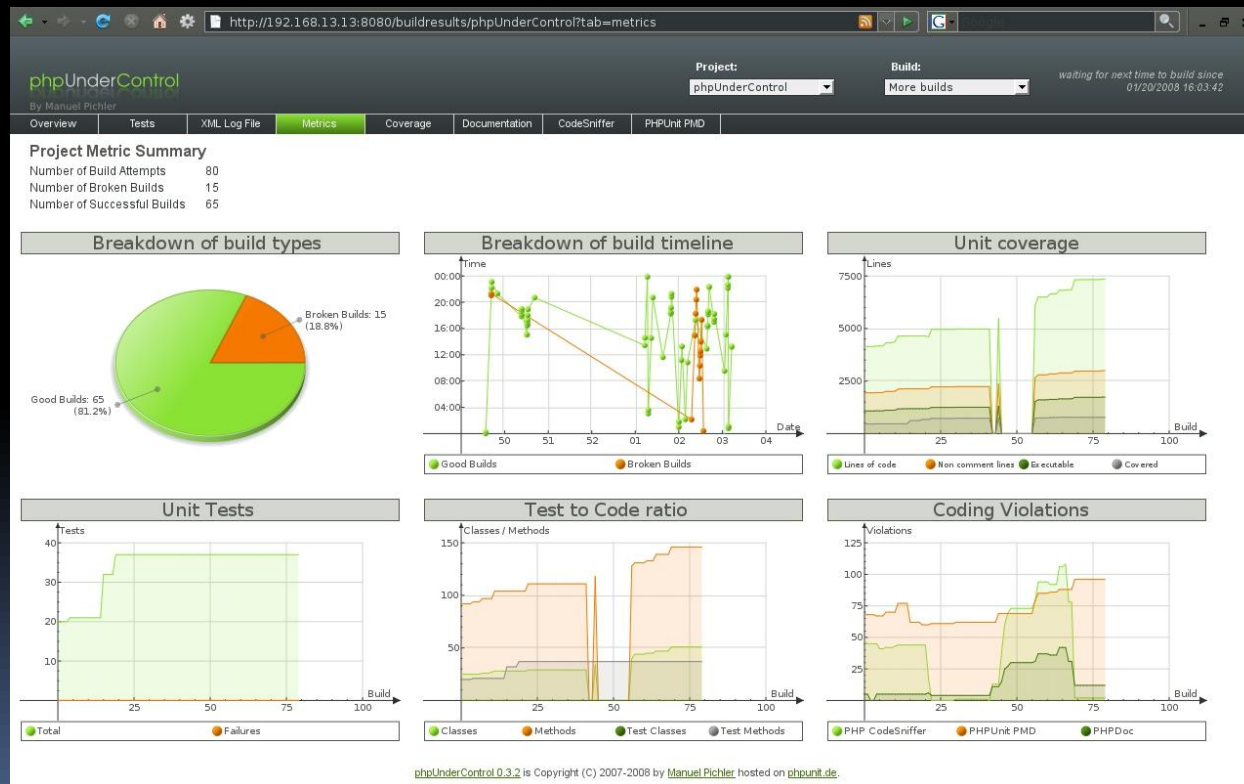


- By Manuel Pichler
- CI for PHP
- Based on CruiseControl
 - Integrates natively various tools :
 - PHPUnit (+XDebug for code coverage),
 - PHP_CodeSniffer
 - PHPDocumentor
 - PMD via PHPUnit (now PHPMD)

phpUnderControl



- What it gives : metrics graphs



phpUnderControl



- What it gives : report lists

The screenshot shows the phpUnderControl web interface. At the top, there's a navigation bar with tabs: Overview, Tests, XML Log File, Metrics, Coverage, Documentation, CodeSniffer, and PHPUnit PMD (selected). Below the navigation bar, there's a 'PHPUnit PMD Summary' section with the following data:

PHPUnit PMD rule	Files	Error/Warnings
PHPUnit PMD / CodeCoverage	40	79
PHPUnit PMD / CRAP	9	16
PHPUnit PMD / NPathComplexity	1	1

Below the summary, there's a 'Console/ConsoleInput.php (9)' section with a list of violations. Each violation is preceded by a red circle icon and a line number. The violations are:

- 119 The CRAP index is 64. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
- 151 The CRAP index is 64. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
- 194 The code coverage is 50.00 which is considered medium. 1
- 249 The NPath complexity is 642. The NPath complexity of a function or method is the number of acyclic execution paths through that method. A threshold of 200 is generally considered the point where measures should be taken to reduce complexity. 1
- 249 The CRAP index is 3387. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
- 249 The code coverage is 61.97 which is considered medium. 1
- 351 The CRAP index is 64. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
- 377 The CRAP index is 64. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
- 424 The CRAP index is 216. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1

Below the console output, there's a 'Data/ConfigProject.php (7)' section with a list of violations. Each violation is preceded by a red circle icon and a line number. The violations are:

- 137 The code coverage is 0.00 which is considered low. 1
- 152 The code coverage is 50.00 which is considered medium. 1
- 176 The CRAP index is 127. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
- 176 The code coverage is 52.94 which is considered medium. 1
- 218 The code coverage is 0.00 which is considered low. 1
- 232 The code coverage is 0.00 which is considered low. 1
- 247 The code coverage is 66.67 which is considered medium. 1

Finally, there's a 'Data/ConfigArtifactsPublisher.php (6)' section with a list of violations. Each violation is preceded by a red circle icon and a line number. The violation is:

- 68 The code coverage is 0.00 which is considered low. 1

phpUnderControl



- What it gives : PHPCodeBrowser

The screenshot displays the phpUnderControl interface. At the top, it shows the project name 'phpUnderControl' and the author 'By Manuel Pichler'. The main navigation bar includes tabs for Overview, Tests, Metrics, Coverage, Code Browser (selected), Documentation, CodeSniffer, and PHPUnit PMD. The Code Browser tab is active, showing a file named 'Math.php' with 34 errors and 1 notice. The code is displayed in a syntax-highlighted editor. A floating window titled 'errors/notices' is overlaid on the code, showing a table of errors and notices.

start	end	comment	type of error	severity
43	43	PHP version not specified	Checkstyle	warning
43	43	Missing @category tag in file comment	Checkstyle	error
54	54	Missing @category tag in class comment	Checkstyle	error
83	83	Doc comment for "\$v1" missing	Checkstyle	error
83	83	Doc comment for "\$v2" missing	Checkstyle	error
85	85	Missing @return tag in function comment	Checkstyle	error
89	89	Expected "if (...) {\n"; found "if (...) {\n"	Checkstyle	error
92	92	Expected "for (...) {\n"; found "for (...) {\n"	Checkstyle	error
104	104	Expected "for (...) {\n"; found "for (...) {\n"	Checkstyle	error
106	106	Space after opening parenthesis of function call prohibited	Checkstyle	error
106	106	Space before closing parenthesis of function call prohibited	Checkstyle	error
111	111	Expected "foreach (...) {\n"; found "foreach (...) {\n"	Checkstyle	error
117	117	Expected "if (...) {\n"; found "if (...) {\n"	Checkstyle	error
120	120	Expected "for (...) {\n"; found "for (...) {\n"	Checkstyle	error

At the bottom of the interface, a footer note states: 'phpUnderControl 0.5.0 is Copyright (c) 2007-2009 by Manuel Pichler, hosted on phpunit.de. phpUnderControl is an extension for CruiseControl.'

Arbit



- By Qafoo (with Manuel Pichler)
- Basically a project multi-services tool
 - Ticketing system
 - Repository browser
 - Continuous integration
- As Manuel is in it, some graphical presentations are unique for this tool
- Still alpha

Arbit



- What it gives : more metrics graphs



4

- What it gives : PHP_Depend overview

The screenshot shows the Arbit project tracking interface. The main content area is titled "Project classes" and "Importance by: Comment lines of code". A list of classes is shown with various metrics. A dropdown menu is open, showing a list of metrics including "Comment lines of code", "Code rank", "Class size", "Depth of inheritance tree", "Executable lines of code", "Number of implemented interfaces", "Lines of code", "Non-comment lines of code", "Number of methods", "Source code rank", "Number of defined class properties", "Number of own and inherited class properties", "Number of public class properties", "Required Method per Class (Sum of method Cyclomatic Complexity)", "Weighted Method per Class + inherited MNC", and "Weighted Method per Class for all public class methods".

On the right side, there is an "Overview pyramid" chart showing various metrics:

Metric	Value
LOC	14541
LOC	1815
CYCLO	1946
LOC	1484
FANOUT	8.818

Below the pyramid is a "JDepend chart" showing a scatter plot of instability versus AFDelta90N. The chart shows a negative correlation between the two metrics, with a green trend line. The y-axis is labeled "Instability" and ranges from 0.0 to 1.0. The x-axis is labeled "AFDelta90N" and ranges from 0.0 to 1.0. The chart is generated by PHP_Depend.

Arbit



- What it gives : Annotated sources

```
94 | 1 kore | }
95 | 1 kore |
96 | 1 kore |     return $mechanisms;
97 | 1 kore | }
98 | 1 kore |
99 | 1 kore | /**
100 | 336 kore |  * User registration
101 | 1 kore |  *
102 | 1 kore |  * Display a registration form with the available registration mechanisms
103 | 1 kore |  * and dispatch to the authentication mechanisms implementation on data
104 | 1 kore |  * retrieval for the actual user registration.
105 | 1236 kore |  *
106 | 1 kore |  * @param arbitRequest $request
107 | 336 kore |  * @return arbitViewCoreUserRegistrationModel
108 | 332 kore |  */
109 | 1 kore | public function register( arbitRequest $request )
110 | 1 kore | {
111 | 1 kore |     // Fetch auth mechanisms to use from project configuration
112 | 334 kore |     $authMechanisms = $this->getAuthMechanisms();
113 | 332 kore |
114 | 334 kore |     // Show form for selected auth mechanism in view, after matching
115 | 1 kore |     // against the whitelist of existing auth mechanisms
116 | 1043 kore |     if ( !isset( $request->path ) &&
117 | 1 kore |         preg_match( '/(?:?P<id>' . implode( '|', array_map( 'preg_quote', $authMechanisms
118 | 332 kore |         ) . ')/', $request->path, $match ) )
119 | 334 kore |         {
120 | 332 kore |             $selected = $match['id'];
121 | 1 kore |         }
122 | 1 kore |         else
123 | 1 kore |         {
124 | 1 kore |             $selected = reset( $authMechanisms );
125 | 1 kore |         }
126 | 335 kore |         $model = new arbitViewModuleModel(
127 | 332 kore |             $request->action,
128 | 1 kore |             $this->getMenu(),
129 | 1 kore |             new arbitViewCoreUserRegistrationModel(
130 | 1 kore |                 $authMechanisms,
131 | 1 kore |                 $selected
132 | 1 kore |             )
133 | 1 kore |         );
134 | 335 kore |
135 | 1 kore |         // Check if there is submitted data and process it
136 | 1 kore |         if ( arbitHitPools::get( 'submit' ) !== null )
137 | 1 kore |         {
138 | 1 kore |             $authClasses = $this->conf->auth;
139 | 1 kore |             $authClass = $authClasses[array_search( $selected, $authMechanisms )];
140 | 1236 kore |
141 | 335 kore |             // Let the current authentication class handle the registration,
142 | 1 kore |             $authClass->register( $request );
```

Plugins Sonar for PHP



- By me 😊
 - Really by the Java guys at SQLI
 - Frédéric Leroy, Akram Ben Aissi, Jérôme Tama
- Sonar is the state of the art for Open Source QA Reporting in Java
 - Thought for multilanguage
- Can easely integrate all PHP reportings ported from Java tools
 - Junit => PHPUnit
 - JDepend => PHPDepend
 - Java PMD => PHPMD

Plugins Sonar for PHP



- Ok, not always so *easily*
 - CheckStyle is not PHP_CodeSniffer
 - Formats are not identical
 - Multi-language doesn't mean no work to add one
- First release on May 2010
 - 0.2 Alpha state, but workable
 - Easy to install : give it a try !
 - Last version demo : sonar-php.sqli.com
- Ok, enough, here are the screenshots

Plugins Sonar for PHP



■ Dashboard

The screenshot shows the Sonar dashboard for the 'RejectsV2' project. The interface includes a navigation menu on the left with options like 'Dashboard', 'Components', and 'Violations drilldown'. The main content area displays various quality metrics:

- Lines of code:** 968 (1,963 lines)
- Classes:** 16 (48 methods)
- Complexity:** 4,6 / method, 13,9 / class, 223 cplx (with a bar chart showing complexity distribution)
- Comments:** 45,0% (792 lines)
- Duplications:** 53,9% (1,058 lines, 142 blocks, 29 files)
- Code coverage:** 13,7% (1 tests, 1.3 sec)
- Test success:** 100,0% (0 failures, 0 errors)
- Rules compliance:** 58,0%
- Violations:** 139 (Breakdown: Blocker: 0, Critical: 0, Major: 134, Minor: 5, Info: 0)
- Quality Gate:** A radar chart showing scores for Reliability (94,1%), Usability (84,6%), Maintainability (79,2%), and Efficiency (100,0%).

At the bottom, the dashboard indicates the project key is 'org.sqli:RejectsV2' and the language is 'php'. A footer note states: 'Powered by SonarSource - Open Source LGPL - v.1.12 - Plugins - Documentation - Ask a question - Bug/feature request'.

Plugins Sonar for PHP



- Components : treemaps

The screenshot displays the SonarQube interface for a project named 'phpMyAdmin'. The left sidebar contains navigation links: Dashboard, Components (selected), Violations drilldown, Time machine, Clouds, and Hotspots. The main content area shows a table of components with the following data:

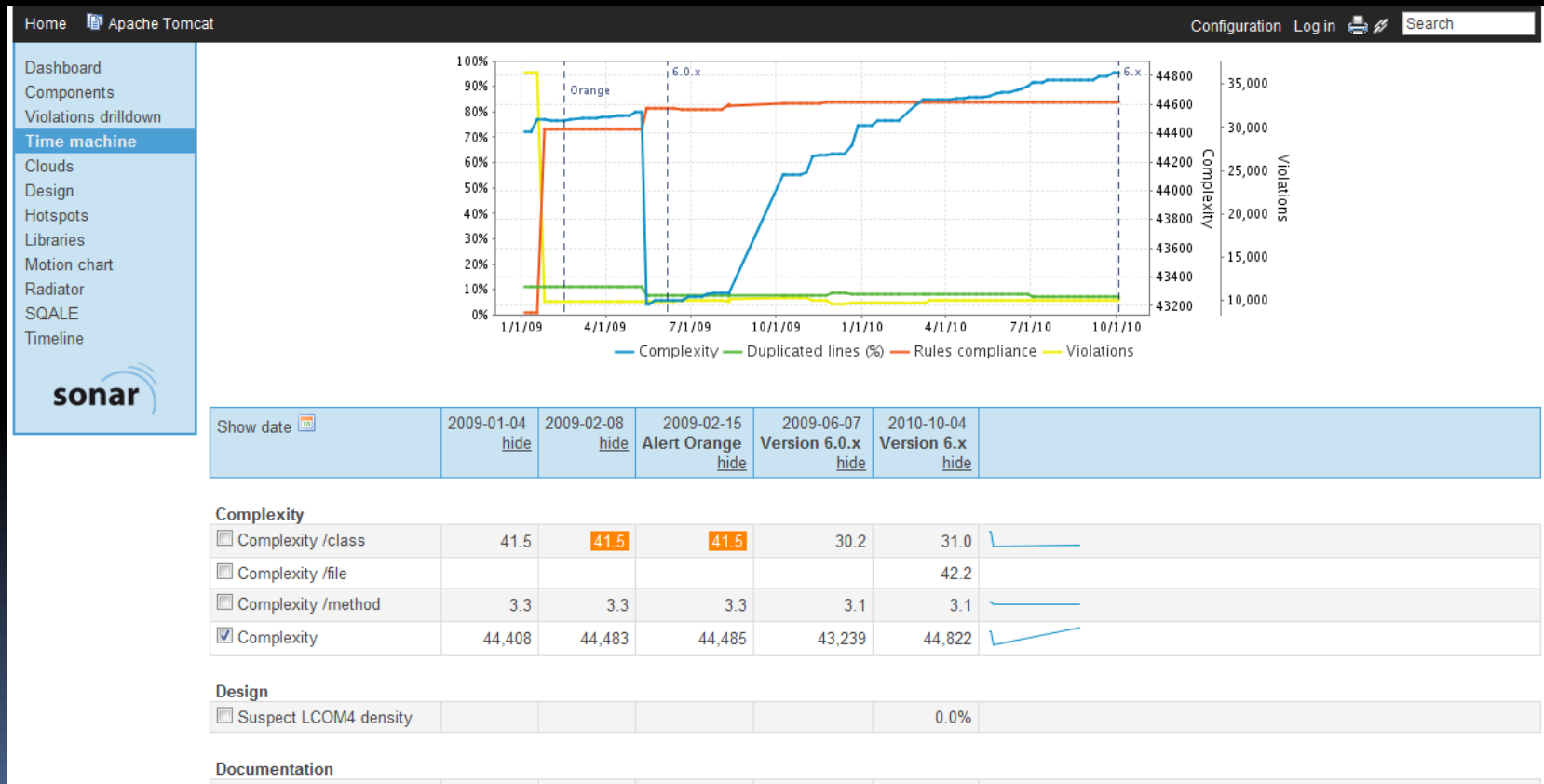
Name	Rules compliance	Coverage	Build time	Links
phpMyAdmin	100.0%		2010-09-24	
Name				
Name	Rules compliance	Coverage	Build time	Links
(default)	100.0%		2010-09-24	
setup			2010-09-24	
setup.lib	100.0%		2010-09-24	
setup.frames			2010-09-24	
test	100.0%		2010-09-24	
libraries	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.Style	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.RichText	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.Shared	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.Shared.Escher.DgContainer.SpgrContainer	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.Shared.Escher.DgContainer	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.Shared.Escher	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.Shared.Escher.DggContainer.BstoreContainer.BSE	100.0%		2010-09-24	
libraries.PHPExcel.PHPExcel.Shared.Escher.DggContainer.BstoreContainer	100.0%		2010-09-24	

On the right side, a treemap visualization shows the hierarchical structure of the components. The largest block is 'libraries', which is further divided into sub-components like 'libraries.PHPExcel.PHPExcel', 'libraries.PHPExcel.PHPExcel.Style', etc. The treemap uses color coding to represent quality metrics, with a legend indicating 'Color' (0.0% to 100.0%) and 'Comments (%)'. The 'Size' of the blocks represents 'Lines of code'.

Plugins Sonar for PHP



Time machine



Plugins Sonar for PHP



Hotspots

The screenshot displays the Sonar - RejectsV2 interface. The browser address bar shows the URL: `http://localhost:9000/plugins/resource/1?page=org.sonar.plugins.core.hotspots.GwtHotspots`. The interface includes a navigation menu on the left with options like Dashboard, Components, and Hotspots. The main content area is divided into several sections:

- Règles les moins respectées**: A table listing rules with their counts and progress bars.

Rule	Count
MISSING_AUTHOR_TAG_CLASS_COMMENT	14
COMMENTS_NOT_ALIGN_FUNCTION_COMMENT	13
STYLE_FUNCTION_COMMENT	11
VARIABLES_NAMES_NOT_ALIGN_FUNCTION_COMMENT	11
OPTIONAL_PARAM_START_FUNCTION_COMMENT	11
- Les moins respectueux des règles**: A table listing components with their rule counts and progress bars.

Component	Count
ChampFormSelect	23
FormulesBd	18
ChampFormRadio	15
RejetController	15
ChampFormCheck	11
- Les plus long tests**: A table listing tests with their execution times.

Test	Time
IndexControllerTest	1.3 sec
- Les plus complexes**: A table listing components with their complexity scores.

Component	Score
RejetController	76
ValiderFormule	22
Initializer	18
ChampFormSelect	18
FormulesBd	18
- Les plus dupliqués**: A table listing components with their duplication counts.

Component	Count
javascript	93
RejetController	79
500	66
ChampFormSelect	56
probleme	56
- Le plus de lignes non testées**: A table listing components with their line coverage metrics.

Component	Metric
Aucune mesure	-
- Avec les méthodes les plus complexes**: A table listing components with their complexity scores.

Component	Score
ValiderFormule	22,0
RejetController	15,2
ChampFormSelect	9,0
ChampFormRadio	9,0
AjaxController	8,0
- Les API les moins documentées**: A table listing components with their documentation metrics.

Component	Metric
Aucune mesure	-

At the bottom of the page, it states: "Powered by SonarSource - Open Source LGPL - v.1.12 - Plugins - Documentation - Ask a question - Bug/feature request"

Plugins Sonar for PHP



■ Violations

The screenshot shows the Sonar - RejetsV2 web interface. The browser address bar displays `http://localhost:9000/drilldown/violations/org.sqli:RejetsV2?trids[]=10#`. The page title is "RejetsV2". The left sidebar contains navigation links: Dashboard, Components, Violations drilldown (selected), Time machine, Clouds, Hotspots, Settings, and Project roles. The main content area shows a summary of violations for "Version 1.0-SNAPSHOT - 18 décembre 2009 14:01 - profile Default PHP".

Priority	Category	Rule	Count
Blocker	0	MISSING_AUTHOR_TAG_CLASS_COMMENT	14
Critical	0	COMMENTS_NOT_ALIGN_FUNCTION_COMMENT	13
Major	134	STYLE_FUNCTION_COMMENT	11
Minor	5	VARIABLES_NAMES_NOT_ALIGN_FUNCTION_COMMENT	11
Info	0	OPTIONAL_PARAM_START_FUNCTION_COMMENT	11
		TYPEHINT_MISSING_FUNCTION_COMMENT	10

Below the summary, a table lists violations by source:

Source	Count	Rule
default.views.helpers	52	RejetController
default.models	42	IndexController
default.controllers	39	AideController
	4	ExitController
default.forms	2	WebserviceController
		AuthController

The breadcrumb path is: Any priority > Any rule > default.controllers > clear > default.controllers.IndexController. The page title for this view is "default.controllers.IndexController".

At the bottom, the "Violations" tab is active, showing a list of violations for the selected controller. A dropdown menu is open, showing the filter "MAJOR (5)". The violations listed are:

- MISSING_AUTHOR_TAG_CLASS_COMMENT (1): Le tag @author est manquant dans le commentaire de classe
- MISSING_VARIABLE_COMMENT (1): Commentaire de variable manquant
- MISSING_FUNCTION_COMMENT (1): Commentaire de fonction manquant

```
14 *
15 * @category Rejets
16 * @package Rejets_Application
17 * @subpackage Controllers
18 *
19 class IndexController extends Zend_Controller_Action {
20     protected $_oIdent = null;
21     MISSING_VARIABLE_COMMENT : Commentaire de variable manquant
22
23     function init() {
24         MISSING_FUNCTION_COMMENT : Commentaire de fonction manquant
25         $this->_oIdent = new Zend_Session_Namespace('oIdent');
26         if (Zend_Session::namespaceIsset('flash_message')) {
27             $this->view->message = Zend_Session::namespaceGet('flash_message');
28             Zend_Session::namespaceUnset('flash_message');
```


Plugins Sonar for PHP



- Editing Code Profile

92 rules found : [export](#) [expand/collapse](#)

Active	Priority	Title	Plugin	Category
<input checked="" type="checkbox"/>	Minor	ATTR_NOT_VALID_CAMEL_VALID_VARIABLE	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Minor	BLANK_LINE_BEFORE_VARIABLE_COMMENT	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Blocker	BLANK_LINE_BETWEEN_CLASS_COMMENT	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Major	BLANK_LINE_FUNCTION_COMMENT	Php checkstyle	Usability
<input type="checkbox"/>	Major	BLANK_LINE_TAGS_FUNCTION_COMMENT	Php checkstyle	Usability
<input type="checkbox"/>	Major	CLASS_COMMENT_EMPTY	Php checkstyle	Maintainability
<input checked="" type="checkbox"/>	Major	COMMENTS_NOT_ALIGN_FUNCTION_COMMENT	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Minor	CONTAINS_NUMBER_VALID_VARIABLE	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Major	DOCCOMMENT_MISSING_FUNCTION_COMMENT	Php checkstyle	Maintainability
<input checked="" type="checkbox"/>	Blocker	DOCCOMMENT_NOT_MATCH_FUNCTION_COMMENT	Php checkstyle	Reliability
<input checked="" type="checkbox"/>	Critical	EMPTY_LINE_LAST_PARAMETER_FUNCTION_COMMENT	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Major	EMPTY_LINE_LAST_PARAMETER_FUNCTION_COMMENT	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Minor	ERROR_PARSING_CLASS_COMMENT	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Info	ERROR_PARSING_CLASS_COMMENT	Php checkstyle	Usability
<input checked="" type="checkbox"/>	Major	ERROR_PARSING_FUNCTION_COMMENT	Php checkstyle	Usability

Conclusion



- Sonar really goes further
 - Best integrates with Hudson
- Still is java...
 - But SonarSource really cooperates
- How to interact with phpUnderControl, Arbit ?
 - (actually our solution – PIC PHP SQLI- is based on phpUC + Sonar)
 - This needs to evolve