Integrating PHP Projects with Jenkins

Sebastian Bergmann

OSCON – July 16th 2012

Sebastian Bergmann

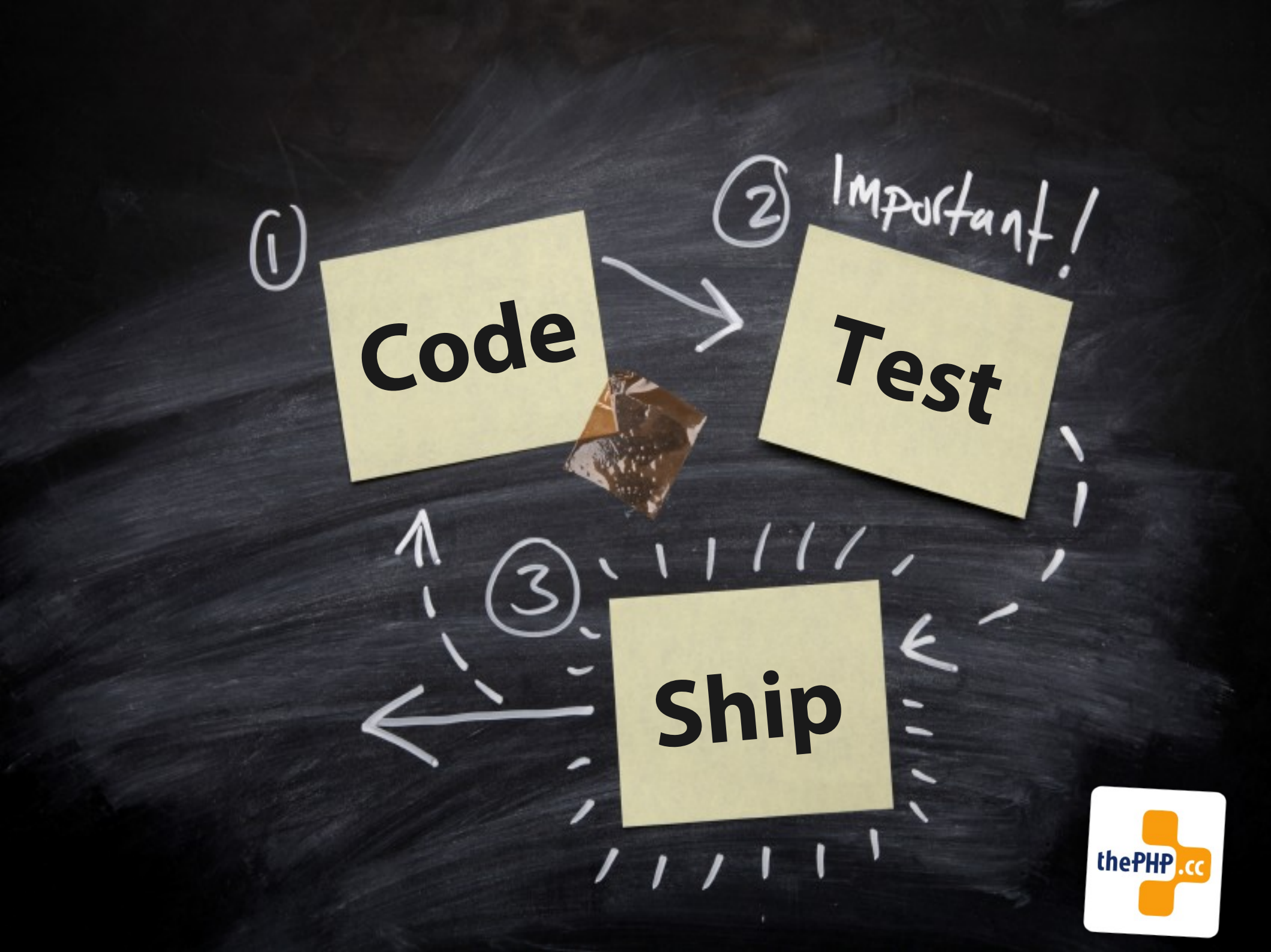
- Has instrumentally contributed to tranforming PHP into a reliable platform for large-scale, critical projects.
- Enterprises and PHP developers around the world benefit from the tools that he has developed and the experience he shares.

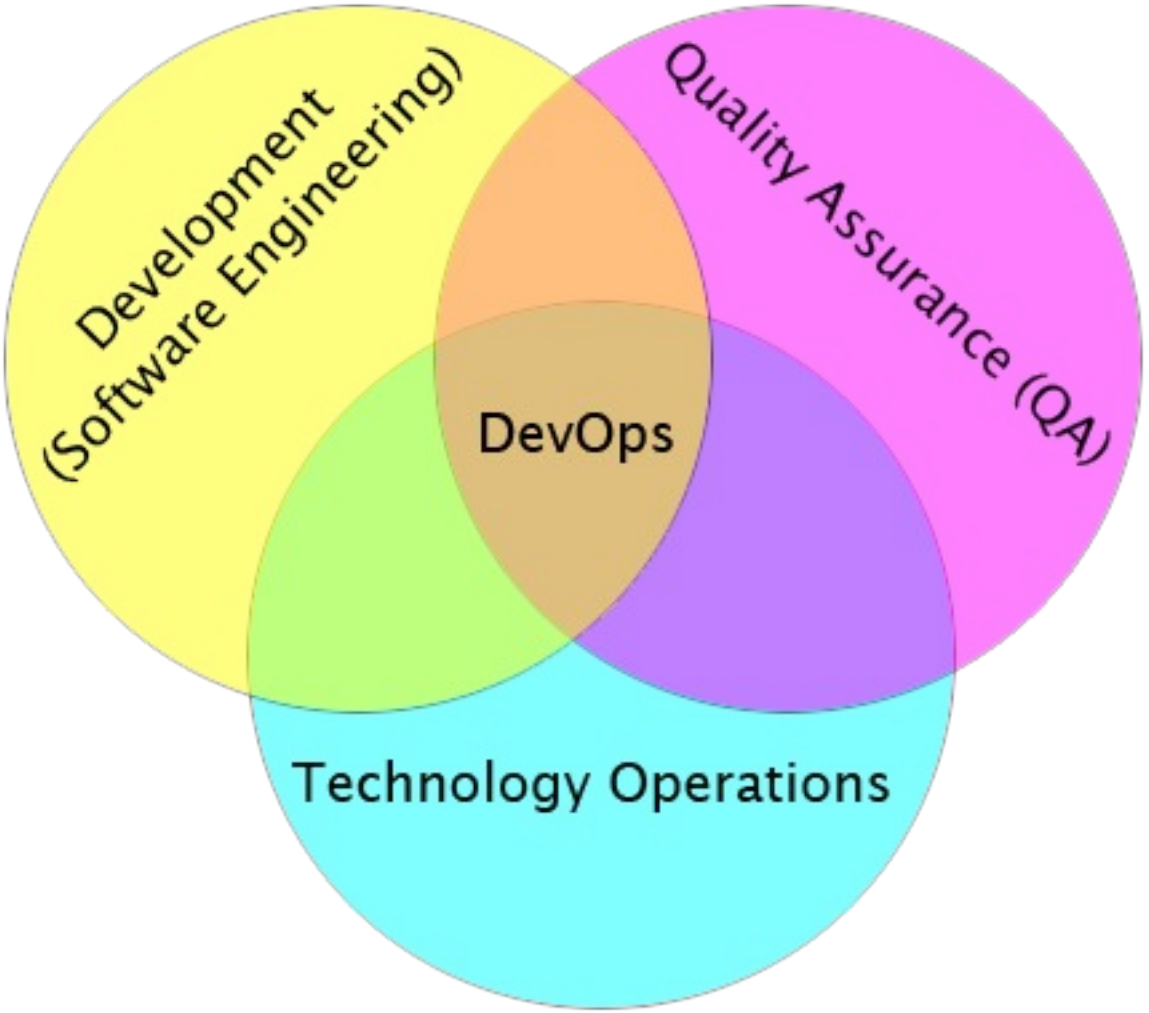
sharing experience

Continuous Integration

„Software development practice where members of a team integrate their work frequently, usually each person integrates at least daily – leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.“

Martin Fowler


1
Code
2
Important!
Test
3
Ship
thePHP.cc



http://en.wikipedia.org/wiki/File:Devops.png

Release Early, Release Often

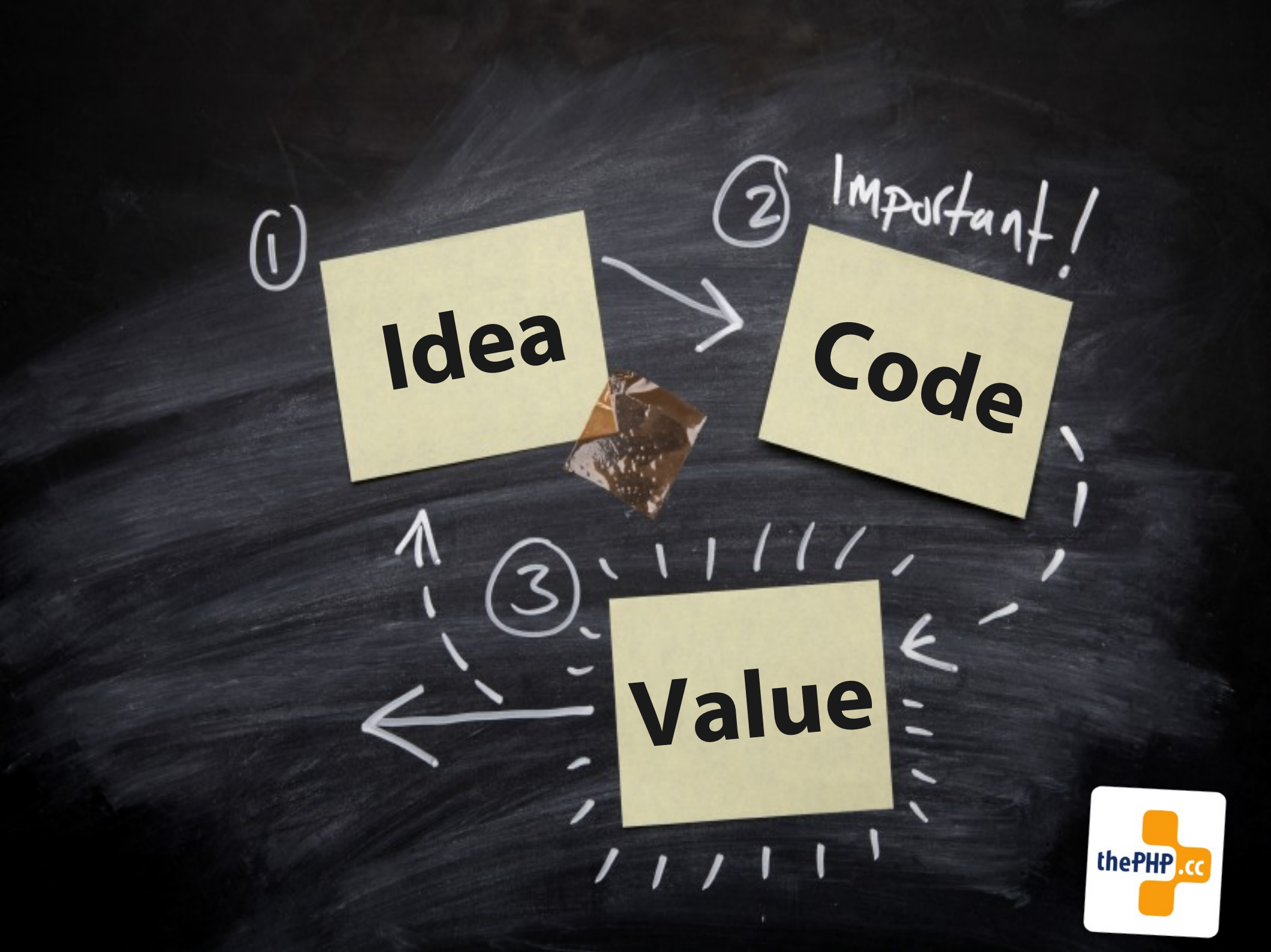
„Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.“

Agile Manifesto

Release Early, Release Often

„Software delivers no revenue until it is in the hands of its users. […] For large companies every week of delay between having an idea and releasing the code that implements it can represent millions of dollars in opportunity costs […]“

Jez Humble and David Farley

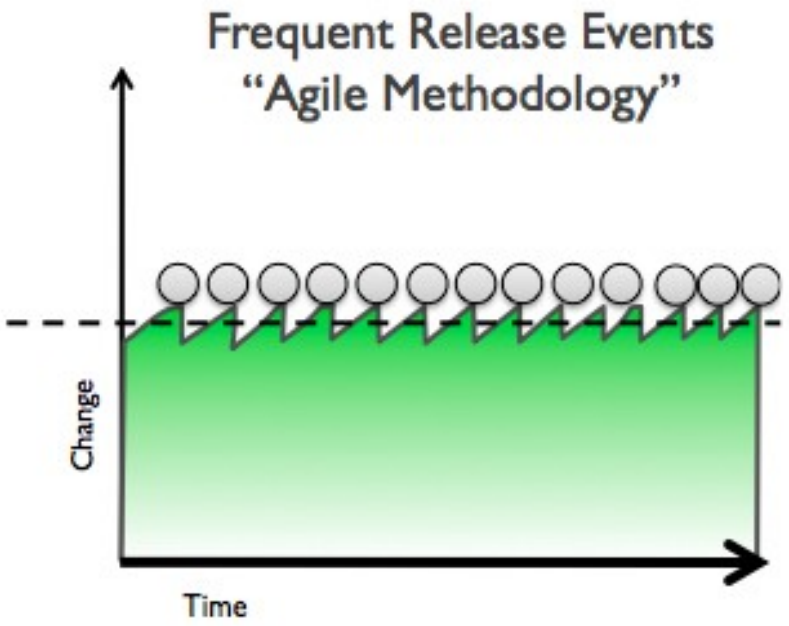

1
Idea
2
Important!
Code
3
Value
thePHP.cc

Release Early, Release Often

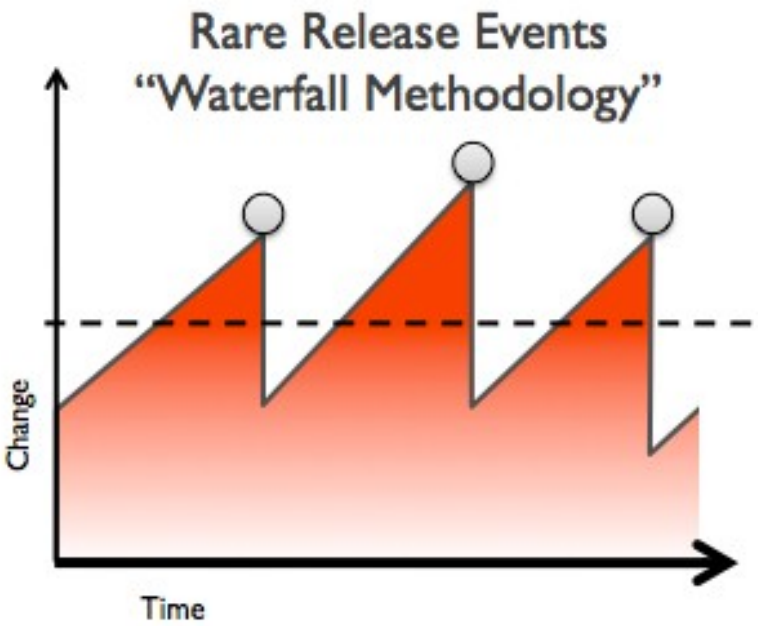
„We need to get rid off the term release cycle.“

Simon Stewart

Release Early, Release Often



Smoother Effort
Less Risk



Effort Peaks
High Risk

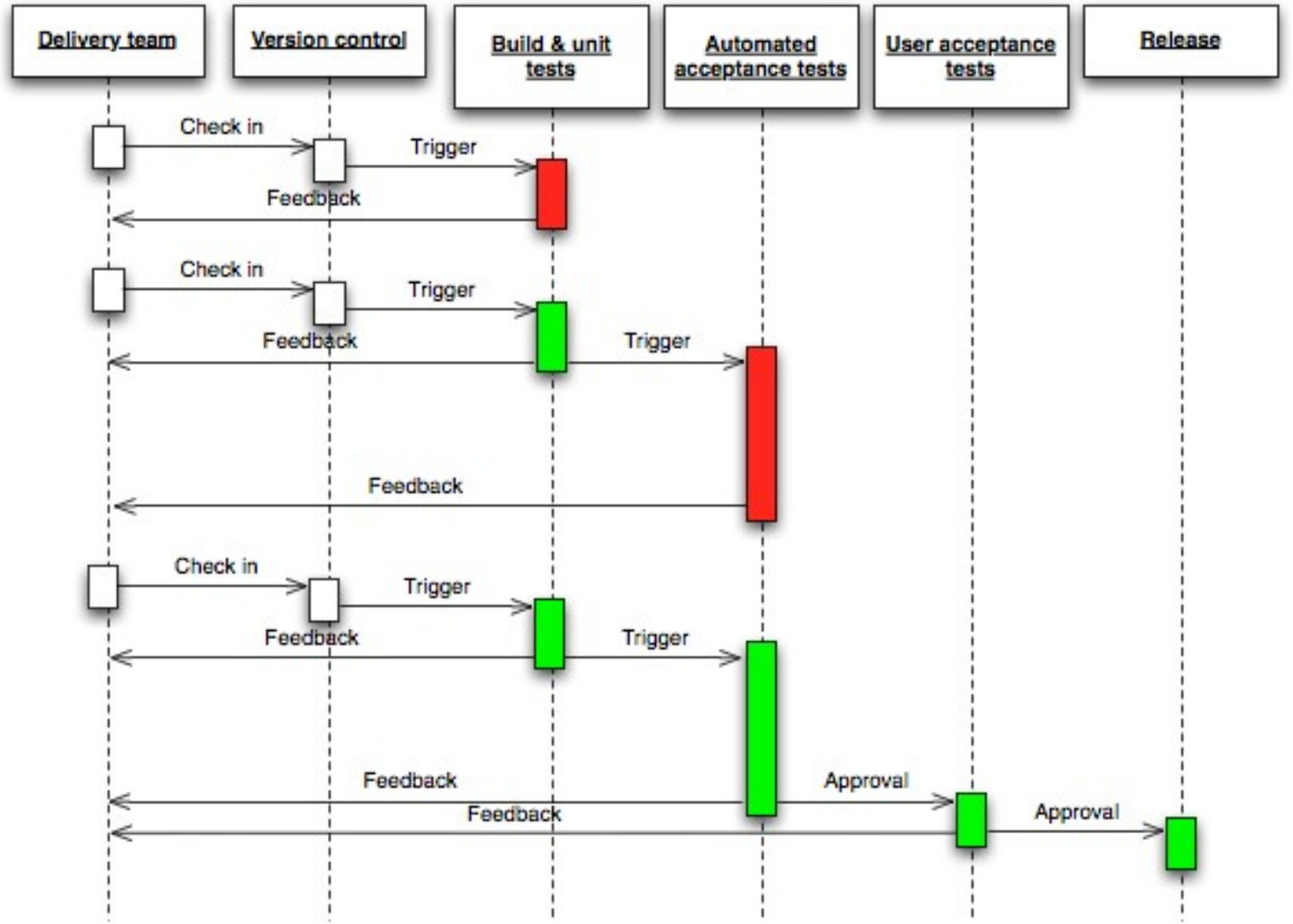
http://en.wikipedia.org/wiki/File:Agile-vs-iterative-flow.jpg

Quantum of Deployment

„The smallest number of steps, with the smallest number of people and the smallest amount of ceremony required to get new code running on your servers.“

http://codeascraft.etsy.com/2010/05/20/quantum-of-deployment

Continuous Delivery



http://en.wikipedia.org/wiki/File:Continuous_Delivery_process_diagram.png


¿¿¿
¿¿¿
thePHP.cc

(Elements of a) Path to Continuous Delivery

- Code
  - Automated Tests
  - Latent Code Patterns (Feature Flags, ...)
- Software Configuration Management
  - Feature Branches
- Automation
  - Build
  - Deployment
- Continuous Integration

Continuous Integration

- Reduce risks
- Reduce repetitive processes
- Prevent low internal software quality
- Generate deployable software
- Enable better project visibility
- Establish greater project confidence

This slide contains material by Paul M. Duvall

Jenkins*

- Continuous Integration Server
- Open Source
- Extendable

* http://jenkins-ci.org/

Installing Jenkins

```
mkdir /usr/local/jenkins
cd /usr/local/jenkins

wget http://mirrors.jenkins-ci.org/war-stable/latest/jenkins.war

export JENKINS_HOME=/usr/local/jenkins
java -jar jenkins.war
```

Installing Plugins for Jenkins (using Jenkins CLI)

```
wget http://localhost:8080/jnlpJars/jenkins-cli.jar

java -jar jenkins-cli.jar -s http://localhost:8080 install-plugin \
checkstyle cloverphp dry htmlpublisher jdepend plot pmd violations \
xunit

java -jar jenkins-cli.jar -s http://localhost:8080 safe-restart
```

Build

„A build acts as the process for putting source code together and verifying that the software works as a cohesive unit.“

Paul M. Duvall

Build Automation

„Build automation is the act of scripting or automating a wide variety of tasks that software developers do in their day-to-day activities including compiling, packaging, running tests, deployment to production“

http://en.wikipedia.org/wiki/Build_Automation

What's in a build?

- Code Generation / Code Transformation
- (Compilation)
- Automated Tests
- Code Analysis
- Documentation Generation
- Packaging / Deployment
- ...

Code Generation / Code Transformation

- Autoloader
  - PHPAB*
- Object-Relational Mapper
  - Model → PHP and SQL code
- Framework
  - Scaffolding

* http://github.com/theseer/Autoload

Apache Ant `build.xml` Script

```
<project name="my-project" default="phpab">
 <target name="phpab" description="Generate autoloader script">
  <exec executable="phpab">
   <arg value="--output" />
   <arg path="${basedir}/src/autoload.php" />
   <arg path="${basedir}/src" />
  </exec>
 </target>
</project>
```

Apache Ant `build.xml` Script

```
<project name="my-project" default="build">
 <target name="build" depends="phpab"/>

 <target name="phpab" description="Generate autoloader script">
  <exec executable="phpab">
   <arg value="--output" />
   <arg path="${basedir}/src/autoload.php" />
   <arg path="${basedir}/src" />
  </exec>
 </target>
</project>
```

Apache Ant `build.xml` Script

```
<project name="my-project" default="build">
 <target name="build" depends="prepare" />

 <target name="clean">
  <!-- ... -->
 </target>

 <target name="prepare" depends="clean,phpab">
  <!-- ... -->
 </target>

 <target name="phpab" description="Generate autoloader script">
  <exec executable="phpab">
   <arg value="--output" />
   <arg path="${basedir}/src/autoload.php" />
   <arg path="${basedir}/src" />
  </exec>
 </target>
</project>
```

Compilation

- Compilation of PHP code to a native binary
  - HipHop
- Syntax Check
  - `php -l`*

* http://www.php.net/manual/en/features.commandline.options.php

Syntax Check

```
<project name="my-project" default="build">
 <target name="build" depends="prepare,lint" />

 <!-- ... -->

 <target name="lint">
  <apply executable="php" failonerror="true">
   <arg value="-l" />

   <fileset dir="${basedir}/src">
    <include name="**/*.php" />
   </fileset>

   <fileset dir="${basedir}/tests">
    <include name="**/*.php" />
   </fileset>
  </apply>
 </target>
</project>
```

Syntax Check

```
<project name="my-project" default="build">
 <target name="build" depends="prepare,lint" />

 <!-- ... -->

 <target name="lint">
  <apply executable="php" failonerror="true">
   <arg value="-l" />

   <fileset dir="${basedir}/src">
    <include name="**/*.php" />
    <modified />
   </fileset>

   <fileset dir="${basedir}/tests">
    <include name="**/*.php" />
    <modified />
   </fileset>
  </apply>
 </target>
</project>
```

PHPUnit*

- De-Facto standard for unit testing in PHP
- Logfiles
  - Test Results in JUnit XML
  - Code Coverage in Clover XML

* http://phpun.it/

PHPUnit

```
<project name="my-project" default="build">
 <target name="build" depends="prepare,lint,phpunit" />

 <target name="clean">
  <delete dir="${basedir}/build/coverage"/>
  <delete dir="${basedir}/build/logs"/>
 </target>

 <target name="prepare" depends="clean,phpab">
  <mkdir dir="${basedir}/build/coverage"/>
  <mkdir dir="${basedir}/build/logs"/>
 </target>

 <target name="phpunit" description="Run unit tests with PHPUnit">
  <exec executable="phpunit" failonerror="true"/>
 </target>
</project>
```

PHPUnit

```
<phpunit bootstrap="src/autoload.php">
 <testsuite name="my-project">
  <directory suffix="Test.php">tests</directory>
 </testsuite>

 <logging>
  <log type="coverage-html" target="build/coverage" />
  <log type="coverage-clover" target="build/logs/clover.xml" />
  <log type="junit" target="build/logs/junit.xml" />
 </logging>

 <filter>
  <whitelist addUncoveredFilesFromWhitelist="true">
   <directory suffix=".php">src</directory>
   <exclude>
    <file>src/autoload.php</file>
   </exclude>
  </whitelist>
 </filter>
</phpunit>
```

Jenkins Plugin: xUnit*

- „This plugin makes it possible to publish the test results of an execution of a testing tool in Jenkins“
- JUnit XML is not really standardized
  - PHPUnit uses nested `<testsuite>` elements

* http://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin

Jenkins Plugin: Clover PHP*

- „This plugin allows you to capture code coverage reports from PHPUnit“
- Only exists because the Clover plugin has some quirks as it expects the real Clover tool (for Java) to be used

* http://wiki.jenkins-ci.org/display/JENKINS/Clover+PHP+Plugin


thePHP.cc

PHPLOC*

- „A tool for quickly measuring the size of a PHP project“
- Logfile: CSV

* http://github.com/sebastianbergmann/phploc

PHPLOC

```
Directories:                                  11
Files:                                        22

Lines of Code (LOC):                         601
  Cyclomatic Complexity / Lines of Code:    0.04
Comment Lines of Code (CLOC):                  7
Non-Comment Lines of Code (NCLOC):           594

Namespaces:                                   11
Interfaces:                                    1
Traits:                                        0
Classes:                                      20
  Abstract:                                    1 (5.00%)
  Concrete:                                   19 (95.00%)
  Average Class Length (NCLOC):               22
Methods:                                      38
  Scope:
    Non-Static:                               38 (100.00%)
    Static:                                    0 (0.00%)
  Visibility:
    Public:                                   26 (68.42%)
    Non-Public:                               12 (31.58%)
  Average Method Length (NCLOC):              11
  Cyclomatic Complexity / Number of Methods: 1.58

Anonymous Functions:                           1
Functions:                                     0

Constants:                                     0
  Global constants:                            0
  Class constants:                             0
```

PHPLOC

```
<target name="phploc">
 <exec executable="phploc">
  <arg value="--log-csv" />
  <arg value="${basedir}/build/logs/phploc.csv" />
  <arg path="${basedir}/src" />
 </exec>
</target>
```

Jenkins Plugin: Plot*

- „This plugin provides generic plotting (or graphing) capabilities in Jenkins“
- Used to plot the metrics collected by PHPLOC over time

* http://wiki.jenkins-ci.org/display/JENKINS/Plot+Plugin

PHP_CodeSniffer*

- „Tokenises PHP, JavaScript and CSS files and detects violations of a defined set of coding standards“
- Logfile: Checkstyle XML

* http://pear.php.net/package/PHP_CodeSniffer

PHP_CodeSniffer

```
PHP CODE SNIFFER VIOLATION SOURCE SUMMARY
----------------------------------------------------------------------------
STANDARD  CATEGORY           SNIFF                                      COUNT
----------------------------------------------------------------------------
CodeRevi  Functions          Global function found                      2297
CodeRevi  PHP                Global keyword not allowed                 938
Generic   PHP                No silenced errors discouraged             523
CodeRevi  PHP                Forbidden functions found                  77
Generic   Code analysis      For loop with test function call not allowe  53
Generic   Code analysis      Empty statement not allowed warning        34
Generic   PHP                Deprecated functions found                 28
Generic   Code analysis      Useless overriding method found            4
Generic   Classes            Duplicate class name found                 2
Generic   Code analysis      Unconditional if statement found           1
----------------------------------------------------------------------------
A TOTAL OF 3957 SNIFF VIOLATION(S) WERE FOUND IN 10 SOURCE(S)
----------------------------------------------------------------------------

Time: 08:02, Memory: 1750.25Mb
```

PHP_CodeSniffer

```
FILE: /tmp/wordpress/wp-includes/admin-bar.php
--------------------------------------------------------------------------------
FOUND 7 ERROR(S) AND 16 WARNING(S) AFFECTING 23 LINE(S)
--------------------------------------------------------------------------------
  18 | WARNING | Consider refactoring "_wp_admin_bar_init" to avoid global
     |         | functions.
  19 | ERROR   | Use of the "global" keyword is forbidden
  52 | WARNING | Consider refactoring "wp_admin_bar_render" to avoid global
     |         | functions.
  53 | ERROR   | Use of the "global" keyword is forbidden
  78 | WARNING | Consider refactoring "wp_admin_bar_my_account_menu" to avoid
     |         | global functions.
  79 | ERROR   | Use of the "global" keyword is forbidden
 101 | WARNING | Consider refactoring "wp_admin_bar_dashboard_view_site_menu"
     |         | to avoid global functions.
 119 | WARNING | Consider refactoring "wp_admin_bar_my_sites_menu" to avoid
     |         | global functions.
 120 | ERROR   | Use of the "global" keyword is forbidden
 154 | WARNING | Consider refactoring "wp_admin_bar_shortlink_menu" to avoid
     |         | global functions.
 176 | WARNING | Consider refactoring "wp_admin_bar_edit_menu" to avoid global
     |         | functions.
     .
     .
     .
```

PHP_CodeSniffer

```
<target name="phpcs">
 <exec executable="phpcs" output="/dev/null">
  <arg value="--report=checkstyle" />
  <arg value="--report-file=${basedir}/build/logs/checkstyle.xml" />
  <arg value="--standard=${basedir}/build/phpcs.xml" />
  <arg value="--ignore=autoload.php" />
  <arg path="${basedir}/src" />
 </exec>
</target>
```

PHP_CodeSniffer

```
<ruleset name="name-of-your-coding-standard">
 <description>Description of your coding standard</description>

 <rule ref="Generic.PHP.DisallowShortOpenTag"/>
 <!-- ... -->
</ruleset>
```

Jenkins Plugin: Checkstyle*

- „This plugin generates the trend report for Checkstyle, an open source static code analysis program“
- Used to report PHP_CodeSniffer results

* http://wiki.jenkins-ci.org/display/JENKINS/Checkstyle+Plugin

PHP Copy/Paste Detector (PHPCPD)*

- Copy/Paste Detector (CPD) for PHP code
- Logfile: PMD-CPD XML

* http://github.com/sebastianbergmann/phpcpd

PHP Copy/Paste Detector (PHPCPD)

```
phpcpd 1.3.2 by Sebastian Bergmann.

Found 26 exact clones with 459 duplicated lines in 4 files:

  - wp-content/plugins/akismet/admin.php:488-500
    wp-content/plugins/akismet/admin.php:537-549

  - wp-content/plugins/akismet/legacy.php:234-248
    wp-content/plugins/akismet/legacy.php:301-315

    .
    .

  - wp-includes/class-snoopy.php:165-172
    wp-includes/class-snoopy.php:225-232

  - wp-includes/class-snoopy.php:181-187
    wp-includes/class-snoopy.php:339-345

  - wp-includes/class-snoopy.php:317-331
    wp-includes/class-snoopy.php:384-398

0.27% duplicated lines out of 171170 total lines of code.

Time: 5 seconds, Memory: 73.25Mb
```

PHP Copy/Paste Detector (PHPCPD)

```
<target name="phpcpd">
 <exec executable="phpcpd">
  <arg value="--log-pmd" />
  <arg value="${basedir}/build/logs/pmd-cpd.xml" />
  <arg path="${basedir}/src" />
 </exec>
</target>
```

Jenkins Plugin: DRY*

- „This plugin generates the trend report for duplicate code checkers like CPD or Simian“
- Used to report PHPCPD results

* http://wiki.jenkins-ci.org/display/JENKINS/DRY+Plugin

PHP_Depend*

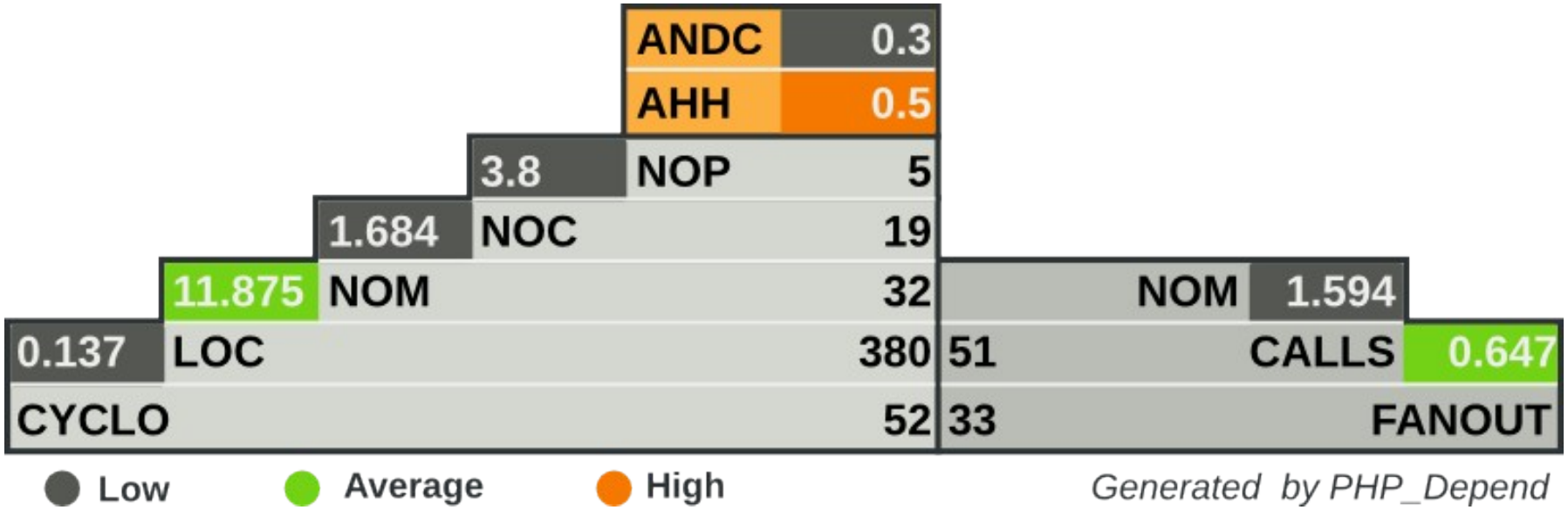
- PHP port of JDepend
- Logfile: JDepend XML
- Also: Software visualizations

* http://pdepend.org/

PHP_Depend

```
pdepend --overview-pyramid=overview_pyramid.svg src

PHP_Depend 1.0.7 by Manuel Pichler

Parsing source files:
.....................                                              22

Executing Coupling-Analyzer:
.....                                                             108

Executing CyclomaticComplexity-Analyzer:
.....                                                             105

Executing Inheritance-Analyzer:
.                                                                  36

Executing NodeCount-Analyzer:
...                                                                70

Executing NodeLoc-Analyzer:
....                                                               91

Generating pdepend log files, this may take a moment.

Time: 00:01; Memory: 18.00Mb
```

PHP_Depend

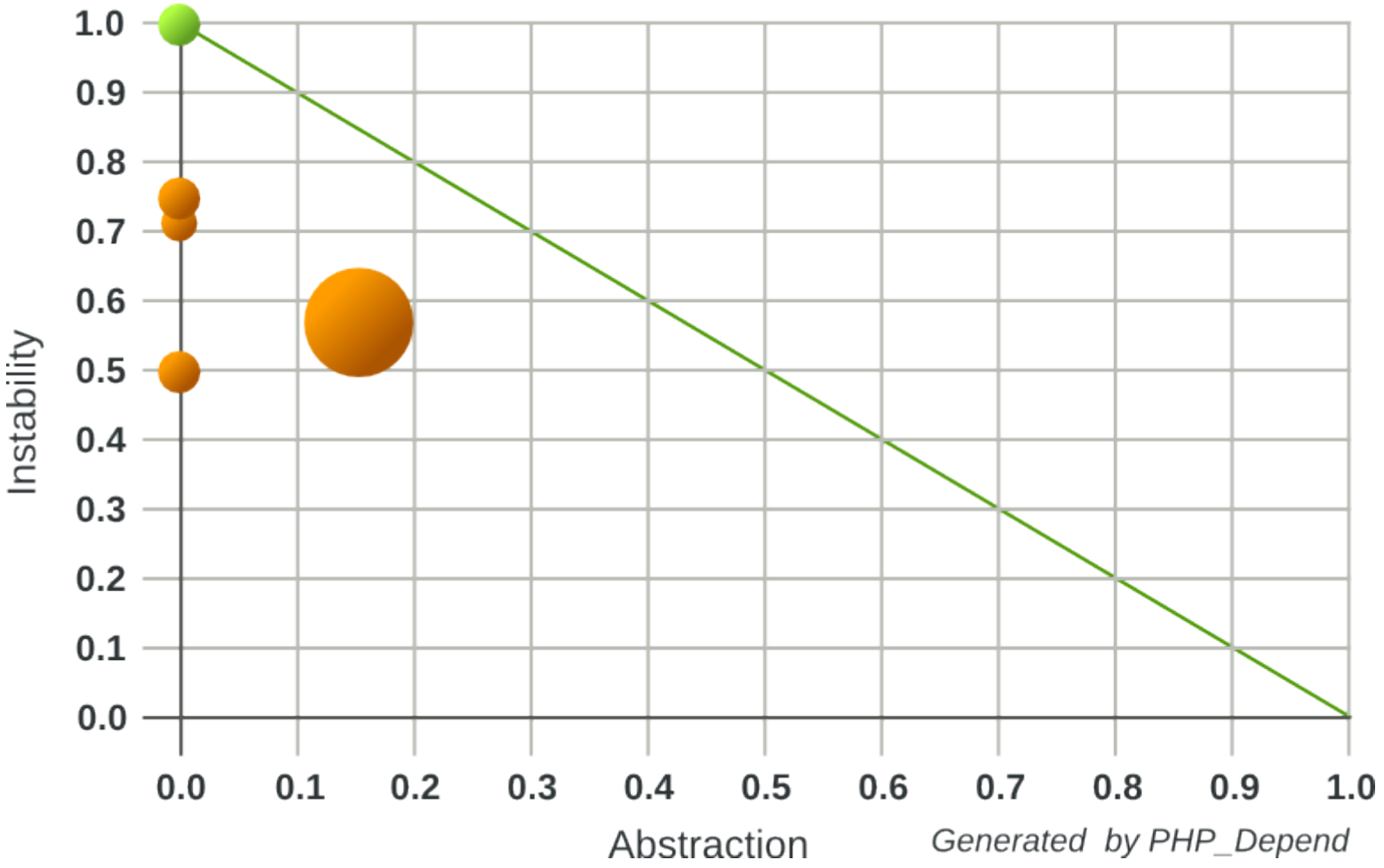


ANDC	Average Number of Derived Classes
AHH	Average Hierarchy Height
NOP	Number of Packages
NOC	Number of Classes
NOM	Number of Methods
LOC	Lines of Code (non-comment, non-whitespace)
CYCLO	Cyclomatic Complexity
CALLS	Number of Operation Calls
FANOUT	Number of Called Classes

PHP_Depend

```
pdepend --jdepend-chart=abstraction_instability.svg src

PHP_Depend 1.0.7 by Manuel Pichler

Parsing source files:
......................                                          22

Executing Dependency-Analyzer:
...                                                             70

Generating pdepend log files, this may take a moment.

Time: 00:00; Memory: 9.75Mb
```

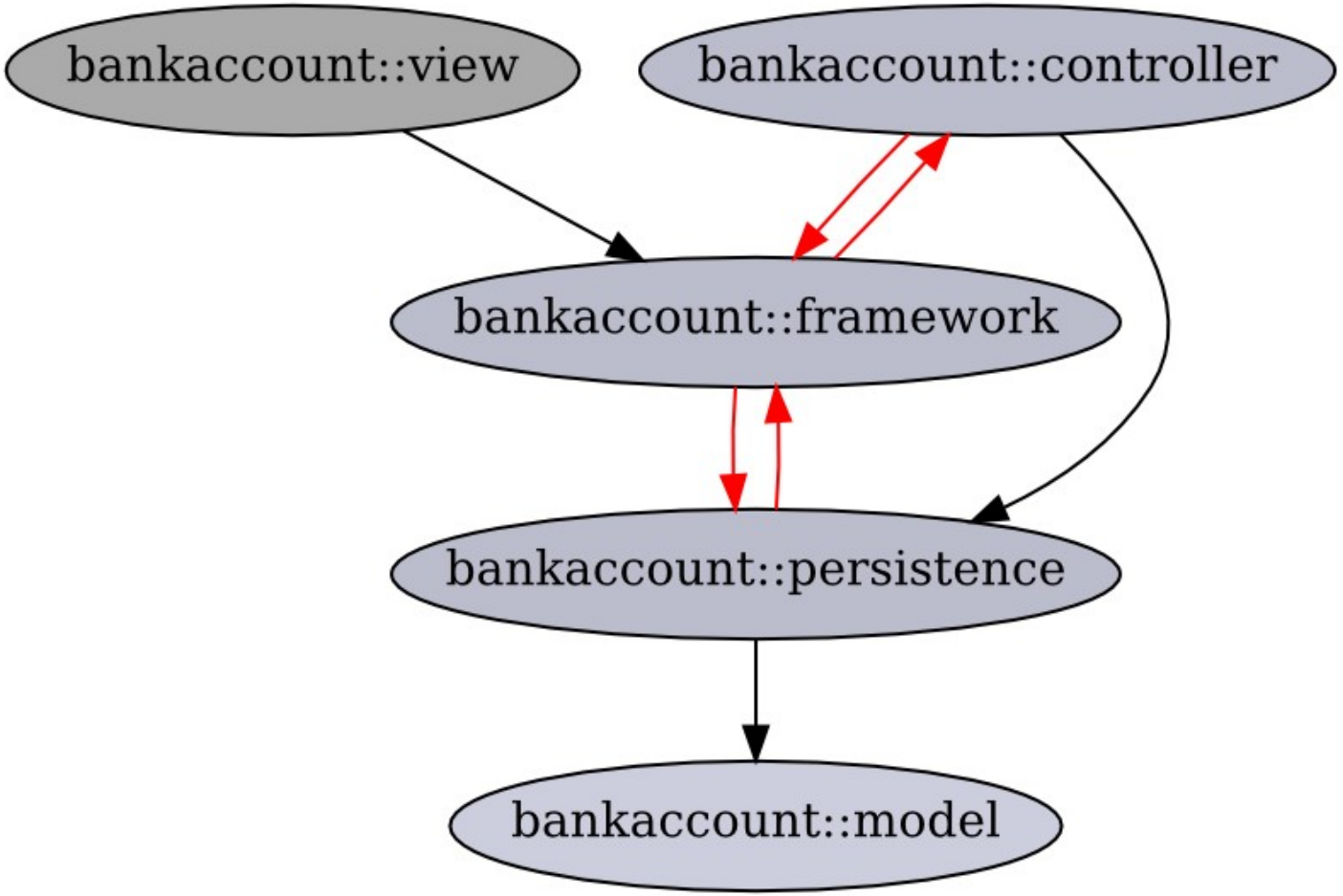
PHP_Depend


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Instability
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Abstraction
Generated by PHP_Depend

PHP_Depend

```
pdepend --jdepend-xml=jdepend.xml src

PHP_Depend 1.0.7 by Manuel Pichler

Parsing source files:
......................                                          22

Executing Dependency-Analyzer:
...                                                             70

Generating pdepend log files, this may take a moment.

Time: 00:00; Memory: 9.75Mb


dependencies.php jdepend.xml -o dependencies.svg
```

PHP_Depend


bankaccount::view
bankaccount::controller
bankaccount::framework
bankaccount::persistence
bankaccount::model

PHP_Depend

```
pdepend --summary-xml=summary.xml src

PHP_Depend 1.0.7 by Manuel Pichler

Parsing source files:
......................                                          22

Executing CyclomaticComplexity-Analyzer:
.....                                                          105

Executing ClassLevel-Analyzer:
....                                                            85

Executing CodeRank-Analyzer:
.                                                               32

Executing Cohesion-Analyzer:
.......                                                        154

Executing Coupling-Analyzer:
.....                                                          108

Executing Hierarchy-Analyzer:
....                                                            87

Executing Inheritance-Analyzer:
.                                                               36

Executing NPathComplexity-Analyzer:
.....                                                          105

Executing NodeCount-Analyzer:
...                                                             70

Executing NodeLoc-Analyzer:
....                                                            91

Generating pdepend log files, this may take a moment.

Time: 00:00; Memory: 19.00Mb
```

Cyclomatic Complexity

Number of possible decision paths in a program or program unit

Thomas J. McCabe, “A Complexity Measure,” IEEE Transactions on Software Engineering 2, No. 4 (IEEE Computer Society Press, Los Alamitos, CA, USA, 1976).

Cyclomatic Complexity

```
inspect.php summary.xml --metric0 ccn

=============================================================
 Name                                                      | Value
=============================================================
 BankAccountMapper::findById()                             | 4.0000
 Router::route()                                           | 4.0000
 BankAccountController::execute()                          | 3.0000
 Request::__call()                                         | 3.0000
 ControllerFactory::getController()                        | 3.0000
 BankAccount::setBalance()                                 | 2.0000
 MapperFactory::getMapper()                                | 2.0000
 BankAccountMapper::getAllIds()                            | 2.0000
 BankAccountMapper::insert()                               | 2.0000
 BankAccountMapper::delete()                               | 2.0000
 BankAccountMapper::update()                               | 2.0000
 BankAccountListView::render()                             | 2.0000
 HashMap::get()                                            | 2.0000
 BankAccount::depositMoney()                               | 1.0000
 .
 .
 .
```

NPath Complexity

Number of acyclic execution paths in a program or program unit

Brian A. Nejmeh, “NPATH: A Measure of Execution Path Complexity and its Applications”, Communications of the ACM 31, Issue 2 (February 1988): 188–200. ISSN 0001-0782.

NPath Complexity

```
inspect.php summary.xml --metric0 npath

=============================================================
 Name                                                      | Value
=============================================================
 Router::route()                                           | 8.0000
 Request::__call()                                         | 6.0000
 BankAccountMapper::findById()                             | 6.0000
 BankAccountController::execute()                          | 4.0000
 ControllerFactory::getController()                        | 3.0000
 BankAccountMapper::getAllIds()                            | 2.0000
 BankAccountListView::render()                             | 2.0000
 BankAccount::setBalance()                                 | 2.0000
 MapperFactory::getMapper()                                | 2.0000
 BankAccountMapper::update()                               | 2.0000
 BankAccountMapper::delete()                               | 2.0000
 BankAccountMapper::insert()                               | 2.0000
 HashMap::get()                                            | 2.0000
 BankAccount::withdrawMoney()                              | 1.0000
  .
  .
  .
```

PHP_Depend

```
<target name="pdepend">
 <exec executable="pdepend">
  <arg value="--jdepend-xml=${basedir}/build/logs/jdepend.xml" />
  <arg path="${basedir}/src" />
 </exec>
</target>
```

Jenkins Plugin: JDepend*

- „The JDepend Plugin is a plugin to generate JDepend reports for builds“
- Used to report PHP_Depend results

* http://wiki.jenkins-ci.org/display/JENKINS/JDepend+Plugin

PHP Mess Detector (PHPMD)*

- „[PHPMD] is a spin-off project of PHP_Depend and aims to be a PHP equivalent of the well known Java tool PMD.

  PHPMD can be seen as an user friendly and easy way to configure frontend for the raw metrics measured by PHP_Depend.“
- Logfile: PMD XML

* http://phpmd.org/

PHP Mess Detector (PHPMD)

```
<target name="phpmd">
 <exec executable="phpmd">
  <arg path="${basedir}/src" />
  <arg value="xml" />
  <arg value="${basedir}/build/phpmd.xml" />
  <arg value="--reportfile" />
  <arg value="${basedir}/build/logs/pmd.xml" />
 </exec>
</target>
```

PHP Mess Detector (PHPMD)

```
<ruleset name="name-of-your-coding-standard">
  <description>Description of your coding standard</description>

  <rule ref="rulesets/codesize.xml/CyclomaticComplexity" />
  <!-- ... -->
</ruleset>
```

Jenkins Plugin: PMD*

- „This plugin generates the trend report for PMD, an open source static code analysis program“
- Used to report PHPMD results

* http://wiki.jenkins-ci.org/display/JENKINS/PMD+Plugin

Jenkins Plugin: Violations*

- „This plug-in generates reports static code violation detectors such as checkstyle, pmd, cpd, findbugs, codenarc, fxcop, stylecop and simian“
- Used to report the results of
  - PHP_CodeSniffer
  - PHP Copy/Paste Detector (PHPCPD)
  - PHP Mess Detector (PHPMD)

* http://wiki.jenkins-ci.org/display/JENKINS/Violations

PHP_CodeBrowser*

» „Generates a browsable representation of PHP code where sections with violations found by quality assurance tools such as PHP_CodeSniffer or PHPMD are highlighted“

* http://github.com/mayflower/PHP_CodeBrowser

PHP_CodeBrowser

```
<target name="phpcb">
 <exec executable="phpcb">
  <arg value="--log" />
  <arg path="${basedir}/build/logs" />
  <arg value="--source" />
  <arg path="${basedir}/src" />
  <arg value="--output" />
  <arg path="${basedir}/build/code-browser" />
 </exec>
</target>
```

Jenkins Plugin: HTML Publisher*

- „This plugin publishes HTML reports“
- API Documentation (from phpdox, for instance)
- Output from PHP_CodeBrowser

* http://wiki.jenkins-ci.org/display/JENKINS/HTML+Publisher+Plugin


thePHP.cc

Jenkins PHP*

- Template for Ant build scripts for PHP projects
- Template for Jenkins jobs for PHP projects

* http://jenkins-php.org/


thePHP.cc

Deployment

- Are the required features implemented?
- Did the required tests pass?
- Are the required dependencies satisfied?
  - Version of PHP
  - PHP Extensions
  - Framework and Libraries
  - Database
  - ...

Package Management

- PEAR Installer
- PHP Archive (PHAR)
- OS Package Manager (RPM, DEB, ...)

PHP Archive (PHAR)

```
<target name="phar">
 <exec executable="phpab">
  <arg value="--phar" />
  <arg value="--output" />
  <arg path="${basedir}/build/bankaccount.phar" />
  <arg path="${basedir}/src" />
 </exec>
</target>
```

RPM/DEB/... Packages

- Version requirements (min/max)
- Dependencies on other packages
- Configuration
- Pre/Post Installation Scripts

RPM/DEB/... Packages

- Reproducible deployment
  - Redeployable
  - Reversible
- Automatable
  - Single Machine
  - Multiple Machines

Automated Deployment

- Build package
  - Continuous Integration Server
  - Manually when needed
- Deploy to test / stage system(s)
- Add to production repository

Automated Deployment: Push vs. Pull

- Push updates to server(s)
  - On-demand action with full control over update
  - Can lead to inconsistent server infrastructure
- Pull updates on server(s)
  - Update only pushed to one place (repository)
  - Fully automated process where the server(s) automatically pull(s) updates

Pull Deployment with RHEL / CentOS / Fedora

- YUM Update Daemon
  - Enable automatic download
  - Enable automatic update installation
- Red Hat Network Satellite

Pull Deployment in General

- Chef
- Puppet
- ...

- Web http://thePHP.cc/
  http://Sebastian-Bergmann.de/
- Mail sebastian@thePHP.cc
- Twitter @S_Bergmann

- Slides http://talks.thePHP.cc/