




DevOps World

 Jenkins World

Static Analysis Plugins
White Mountain Release

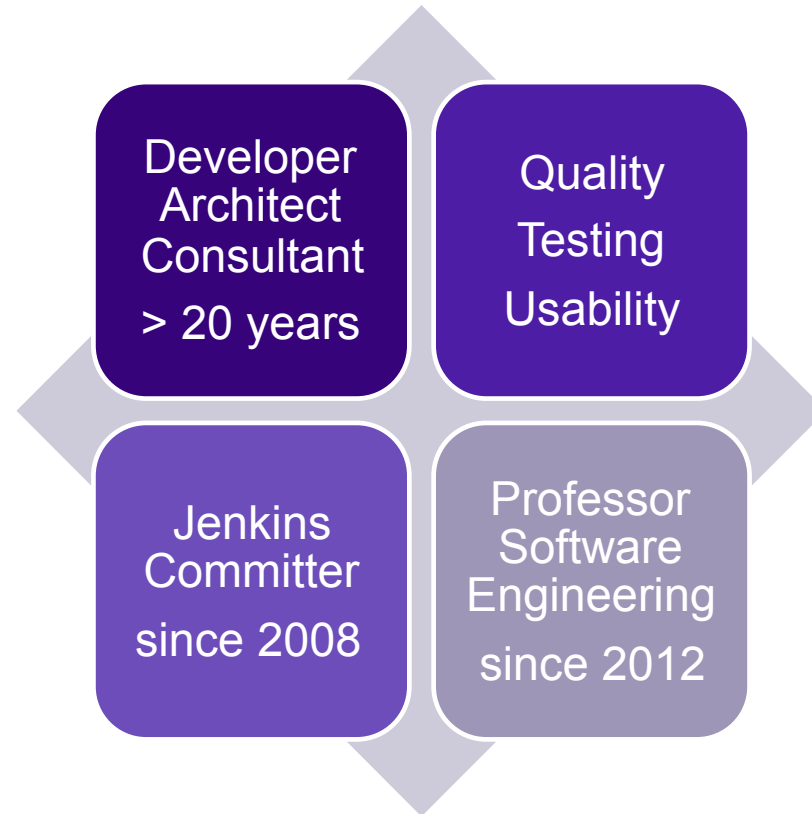
Ullrich Hafner

About me

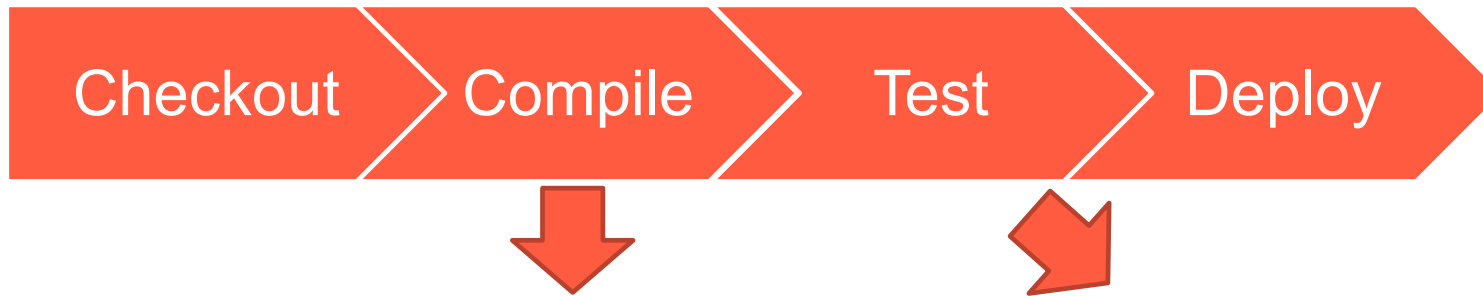


Prof. Dr. Ullrich Hafner
ullrich.hafner@hm.edu

Professor for Software Engineering
University of Applied Sciences Munich



Static analysis plugins – typical use case



```
Started by user unknown or anonymous
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on node1 in /tmp/node1/workspace/Full Analysis
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout)
[Pipeline] git
  > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
  > git config remote.origin.url file:///Users/hafner/De
timeout=10
Fetching upstream changes from file:///Users/hafner/Dev
```

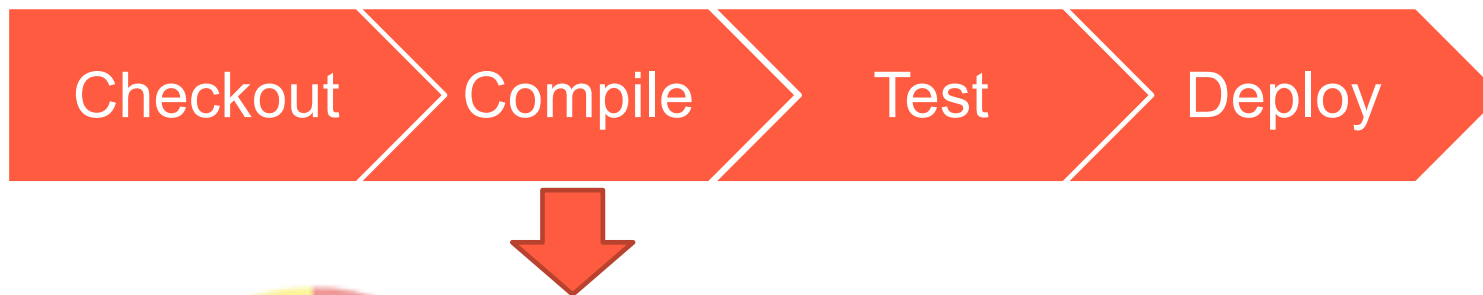
Test Result

0 failures (±0) , 3 skipped (±0)

All Tests

Package	Duration	Fail
edu.hm.hafner	3 sec	0
edu.hm.hafner.analysis	2.3 sec	0
edu.hm.hafner.analysis.parser	7.5 sec	0

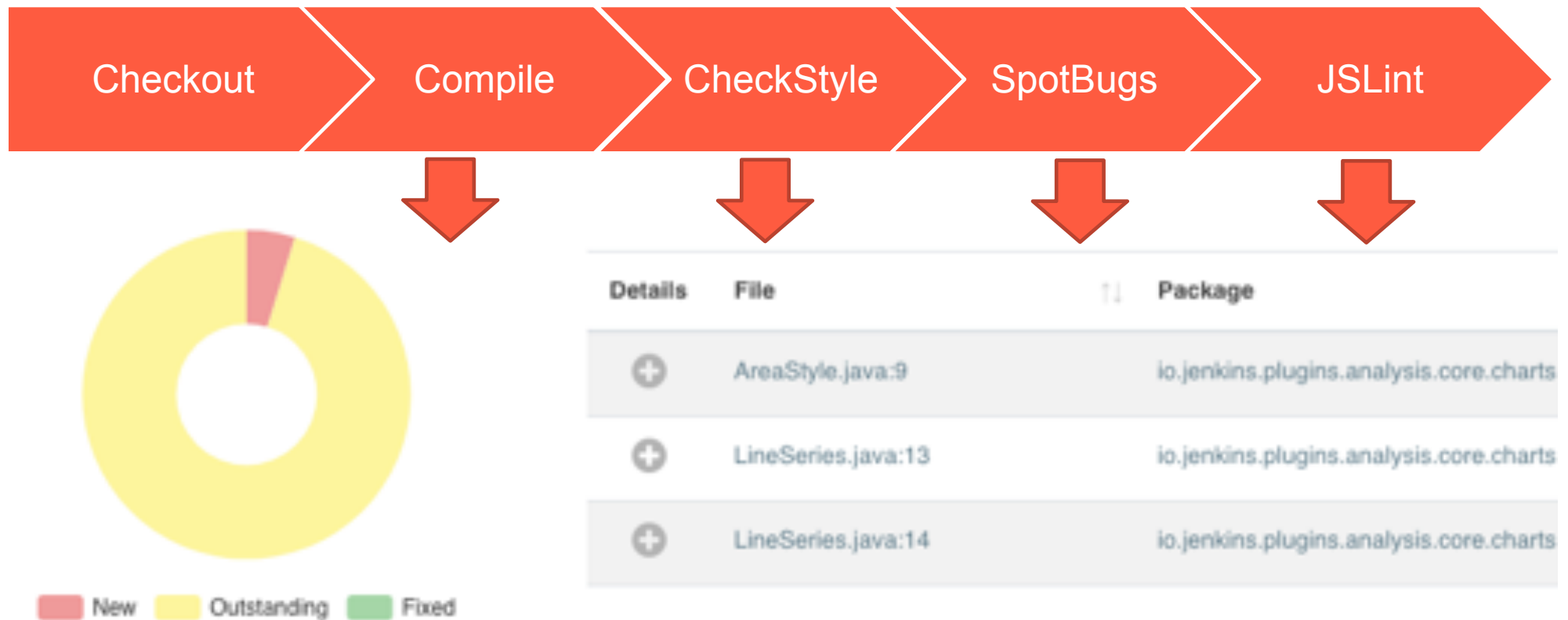
Static analysis plugins visualize compile step results



```
Started by user unknown or anonymous
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on node1 in /tmp/node1/workspace/Full Analysis
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout)
[Pipeline] git
  > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
  > git config remote.origin.url file:///Users/hafner/Dev
timeout=10
Fetching upstream changes from file:///Users/hafner/Dev
```

	Package
Style.java:9	io.jenkins.plugins.analysis.core.charts
Series.java:13	io.jenkins.plugins.analysis.core.charts
Series.java:14	io.jenkins.plugins.analysis.core.charts

Support for almost 100 static analysis reports



Jenkins' static analysis suite



Android Lint



SpotBugs



CheckStyle



...



Warnings



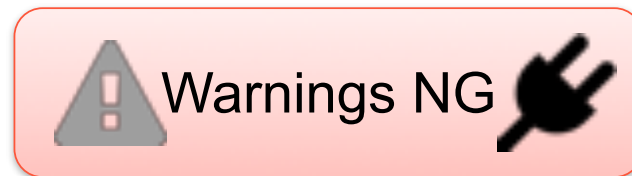
PMD



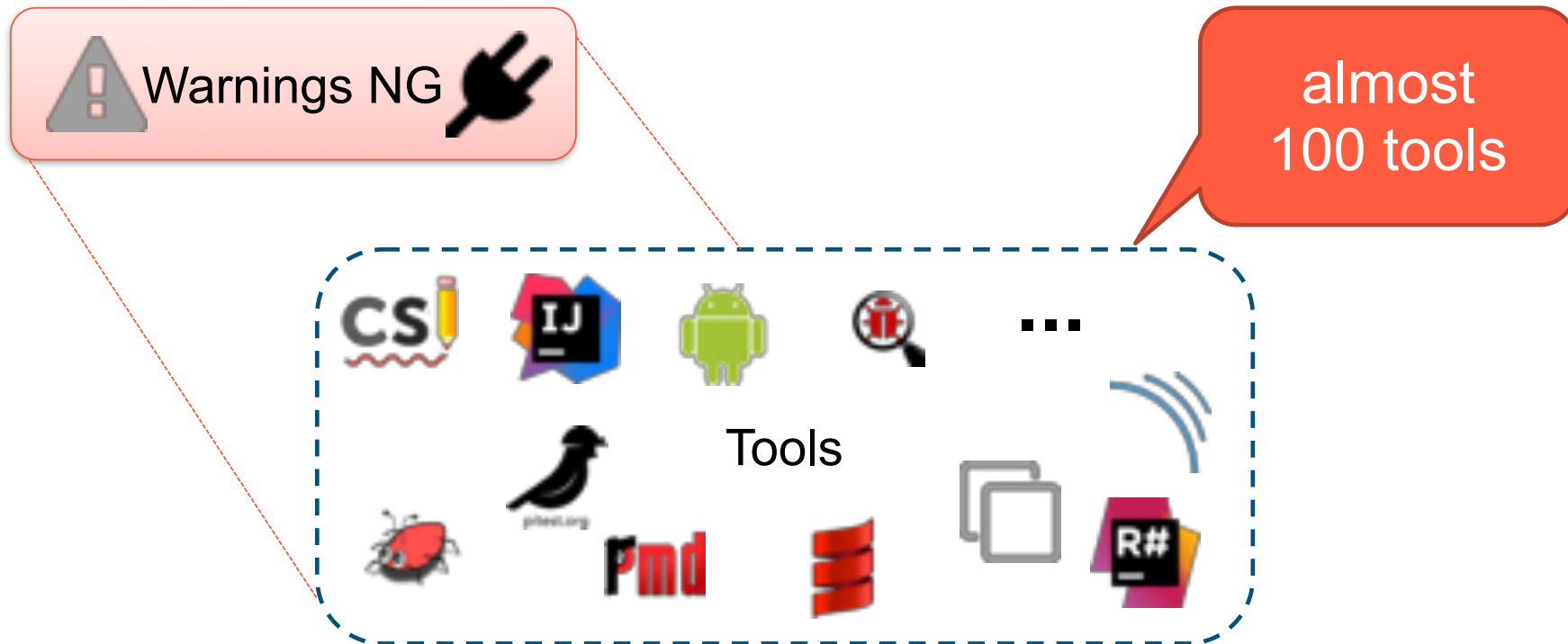
DRY



~~Jenkins' static analysis suite~~ Warnings Next Generation



Warnings Next Generation plugin content





Static Analysis Plugins White Mountain Release

Features (Demo)

Individual tool reports or aggregated reports

Report for each tool



CheckStyle: 91 warnings ⓘ

- Quality gate: ⚠ Unstable



SpotBugs: 21 warnings ⚠

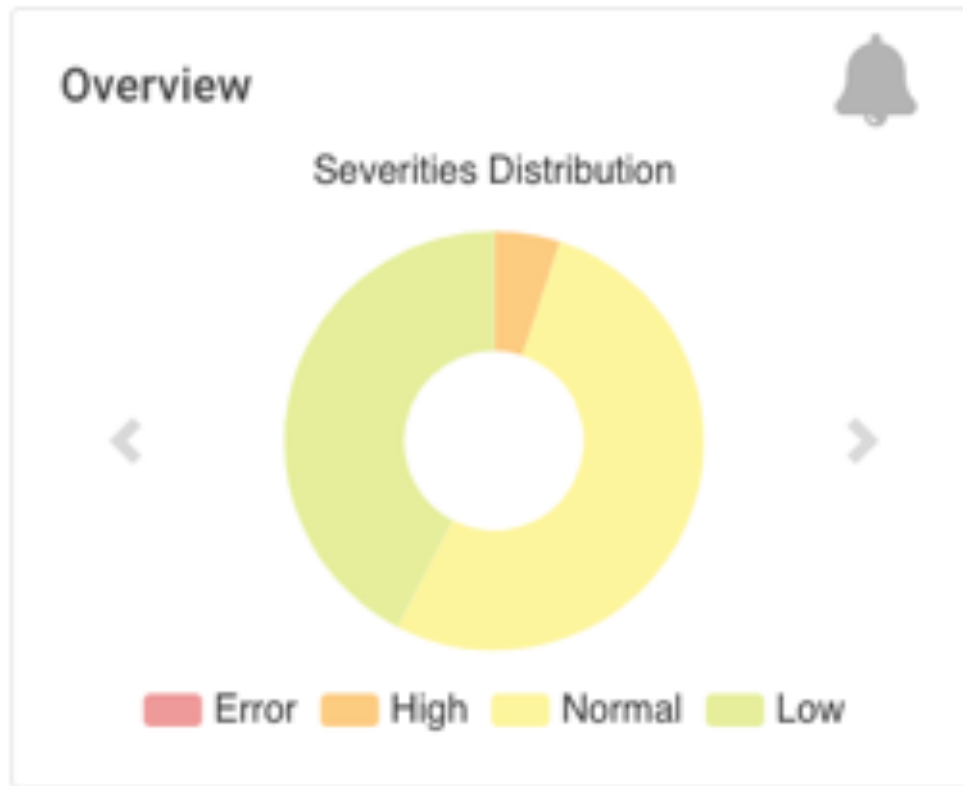


Static Analysis: 795 warnings ⚠

- Static analysis results from: FindBugs, CPD, CheckStyle, PMD, JavaDoc, Java, SpotBugs

Aggregated report

Severity, priority, warnings, errors, issues, etc.



- Same classification used for **all** issue types of **all** tools
- Errors indicate compile errors
- Warnings have a priority
 - High
 - Normal
 - Low
- Tool report parsers use hard-coded mapping (make it configurable?)

Detection of new, fixed, and outstanding issues



- Issues delta for each build
 - New issues since build $\#n$
 - Fixed issues since build $\#n$
- Build ' $\#n$ ': Reference (or baseline)
- Reference is configurable
 - Only stable builds
 - Passed Quality Gate
 - Different Job (e.g., master Branch)
- Optimum in theory
 - SCM branch point
 - Not implemented yet

Details view with paging, filter and sorting

Details

Files Categories Types **Issues**

Show entries Search:

Details	File	Package	Category	Type	Severity	Age
	RuboCop.java:26	io.jenkins.plugins.analysis.warnings	BAD_PRACTICE	SE_NO_SERIALVERSIONID	Low	1
	SonarQube.java:26	io.jenkins.plugins.analysis.warnings	BAD_PRACTICE	SE_NO_SERIALVERSIONID	Low	1
	ToolsList.java:35	io.jenkins.plugins.analysis.warnings	I18N	DM_DEFAULT_ENCODING	Low	1
	ToolsList.java:36	io.jenkins.plugins.analysis.warnings	BAD_PRACTICE	OS_OPEN_STREAM	Low	1

Showing 1 to 4 of 4 entries 1

Details with embedded detail description

Show entries Search:

Details	File	Package	Category	Type
	HealthReportBuilderTest.java:186	io.jenkins.plugins.analysis.core.quality	CORRECTNESS	RV_RETURN_VALUE_IGNORED

Return value of `org.assertj.core.api.Assertions.assertThat(Comparable)` ignored in `io.jenkins.plugins.analysis.core.quality.HealthReportBuilderTest.createTestHealthReport(int, int, Severity, int, int, int, int)`

The return value of this method should be checked. One common cause of this warning is to invoke a method on an immutable object, thinking that it updates the object, for example, in the following code fragment,

```
String dateString = getHeaderField(name);
dateString.trim();
```

the programmer seems to be thinking that the `trim()` method will update the `String` referenced by `dateString`. But since `Strings` are immutable, the `trim()` method returns a new `String` value, which is being ignored here. The code should be corrected to:

```
String dateString = getHeaderField(name);
dateString = dateString.trim();
```

Showing 1 to 1 of 1 entries (filtered from 21 total entries) 1

Source Code View

```
5  /**
6   * A blamer that does nothing.
7   *
8   * @author Ulrich Hafner
9   */
10 public class NullBlamer implements Blamer {
11     @Override
12     public Blames blame(final Report report) {
13         report.logInfo("Skipping blaming as requested in the job configuration");
14         return new Blames();
15     }
16 }
```

 io.jenkins.plugins.analysis.core.scm.NullBlamer is Serializable; consider declaring a serialVersionUID 

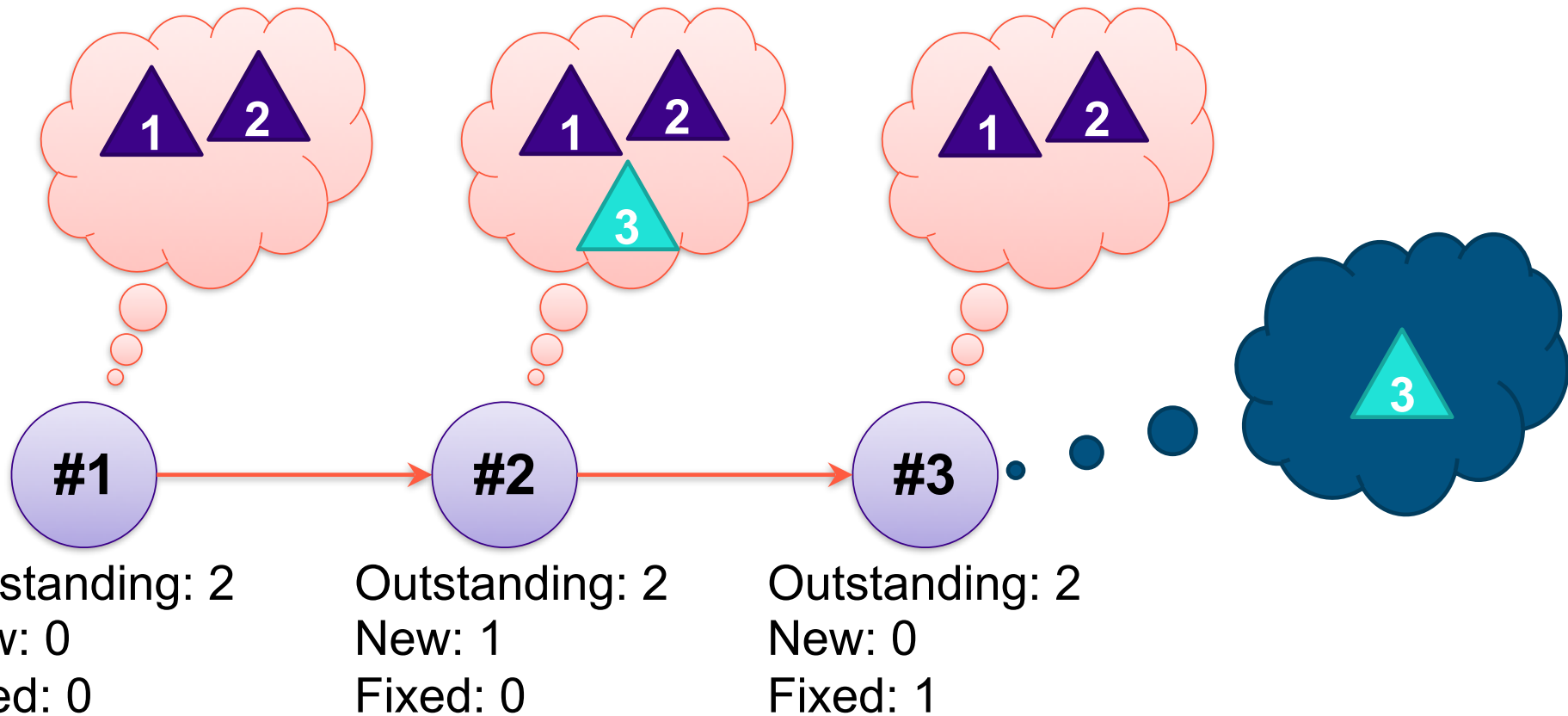
This class implements the `Serializable` interface, but does not define a `serialVersionUID` field. A change as simple as adding a reference to a `.class` object will add synthetic fields to the class, which will unfortunately change the implicit `serialVersionUID` (e.g., adding a reference to `String.class` will generate a static field `class$java$lang$String`). Also, different source code to bytecode compilers may use different naming conventions for synthetic variables generated for references to class objects or inner classes. To ensure interoperability of `Serializable` across versions, consider adding an explicit `serialVersionUID`.



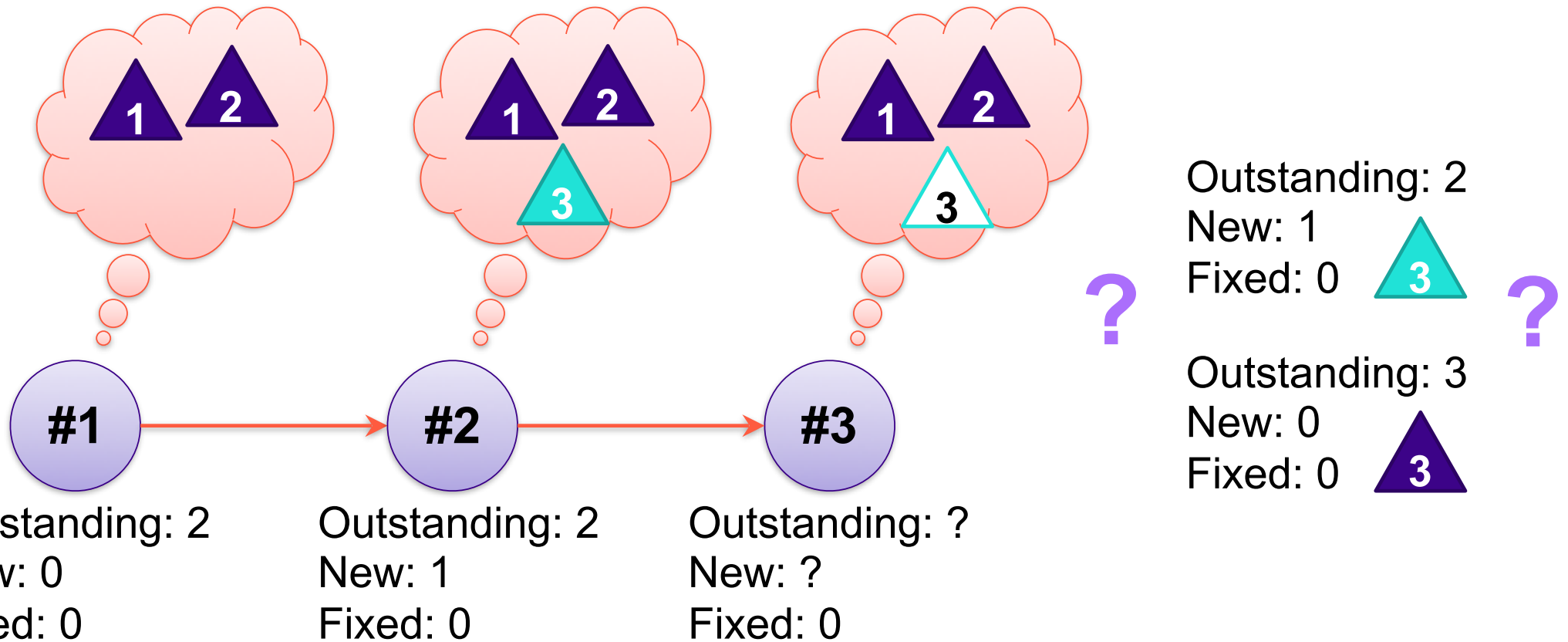
Static Analysis Plugins White Mountain Release

Reference Job
Quality Gate

Baseline examples – simple case



Baseline examples – tracking of new warnings

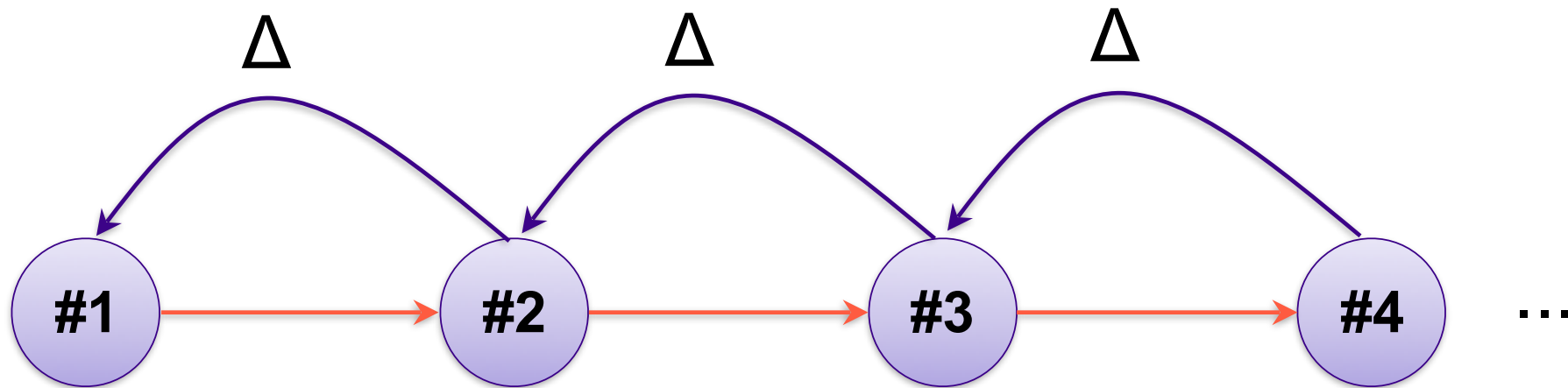


Warnings Plugin – Reference Build

- Warnings supports both ways
 - Delta computation: current with previous
 - Delta computation: current with reference
- Reference
 - Baseline where to start
 - Automatically computed

Reference build – previous build

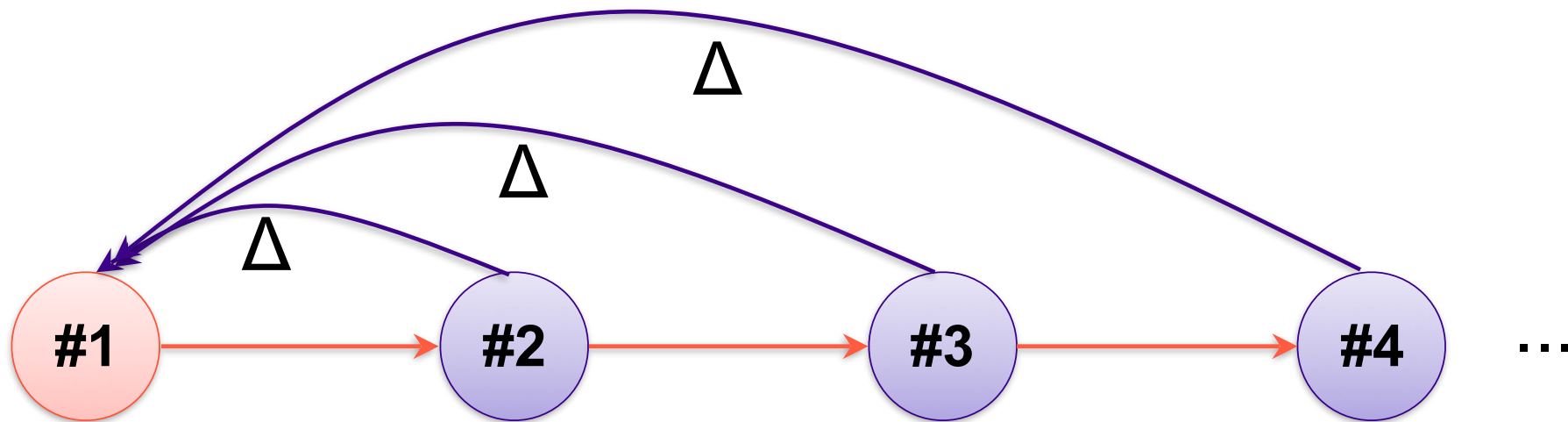
- Delta computation compares current with previous



- Builds will be skipped if job fails (optional)

Reference build – baseline build

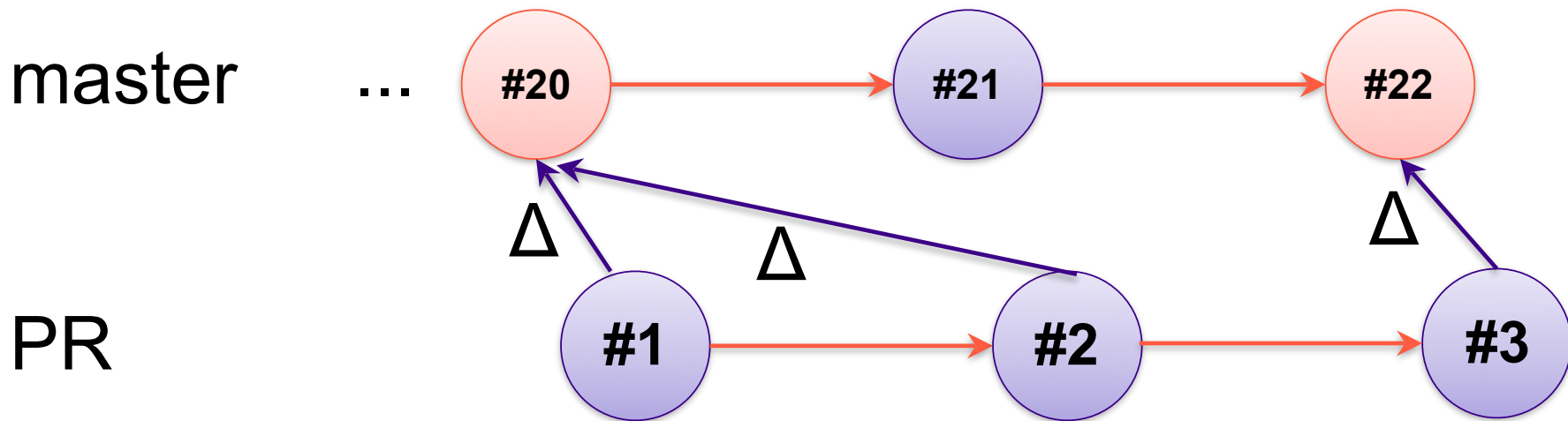
- Delta computation compares current with baseline



- Baseline will automatically adjust if no new warnings

Reference build – baseline from other job

- Delta computation compares current with latest from another job



- Ideal solution: baseline will use branch point from SCM

Quality gates

- Provide a quality gate for each tool
 - Count number of warnings (by severity)
 - ▶ New warnings
 - ▶ Total warnings
- Set build result
 - Quality gate passed: **Success**
 - Quality gate failed: **Unstable** or **Failed**
- Quality gate operates on delta to baseline
- Baseline will be adjusted if quality gate passed
- Jenkins health report is supported as well





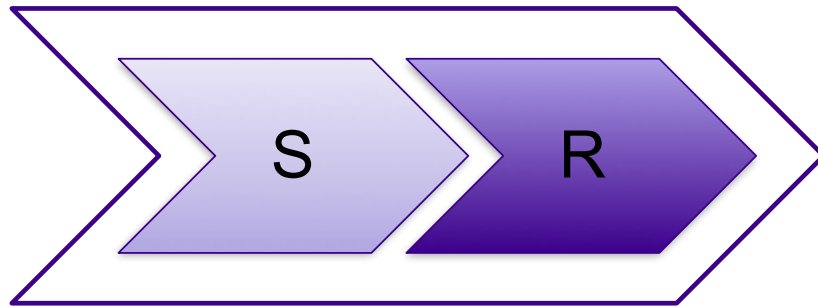
Static Analysis Plugins White Mountain Release

Pipeline Support

Pipeline support

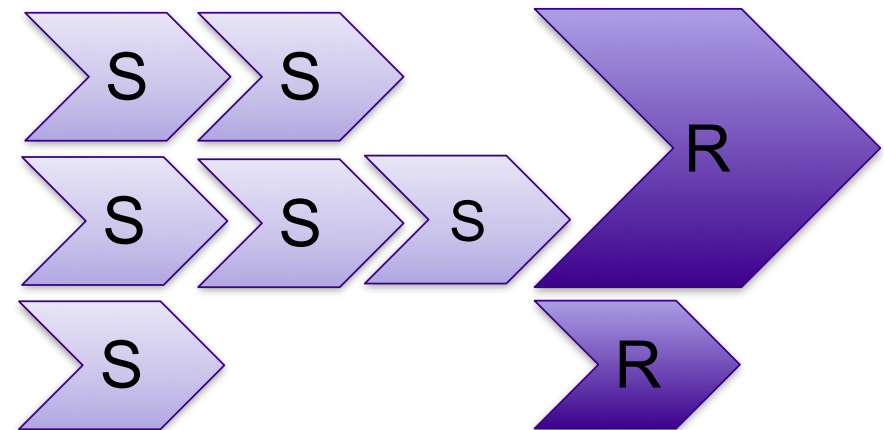
Simple Mode

- Single Step scans for warnings and publishes report



Advanced Mode

- Step to scan for warnings
- Step to publish report



Declarative Pipeline example

```
post {
    always {
        junit testResults: '**/target/surefire-reports/TEST-*.xml'

        recordIssues enabledForFailure: true, tool: mavenConsole()
        recordIssues enabledForFailure: true, tool: errorProne()
        recordIssues enabledForFailure: true, tool: java()
        recordIssues enabledForFailure: true, tool: checkStyle()
        recordIssues enabledForFailure: true, tool: spotBugs()
        recordIssues enabledForFailure: true, unstableNewAll: 1,
            tool: cpd(pattern: '**/target/cpd.xml')
        recordIssues enabledForFailure: true,
            tool: pmdParser(pattern: '**/target/pmd.xml')
        recordIssues enabledForFailure: true,
            tool: taskScanner(includePattern: '**/*.java', highTags: 'FIXME',
                normalTags: 'TODO')
    }
}
```

Scripted Pipeline example

```
stage ('Build') {
    def mvnHome = tool 'mvn-default'

    sh "${mvnHome}/bin/mvn --batch-mode -V -U -e clean verify

    junit testResults: '**/target/surefire-reports/TEST-*.xml'

    def java = scanForIssues tool: [$class: 'Java']
    def javadoc = scanForIssues tool: [$class: 'JavaDoc']

    publishIssues issues:[java,javadoc], unstableTotalAll:1
}
```



Static Analysis Plugins White Mountain Release

Summary

Additional features

- Filtering of warnings
 - Type and category
 - Module, package or file name
 - Include or exclude filters
- View for internal messages
 - No need to open console log
 - Error and info messages
- Source code view
 - Highlighting of all languages
 - Navigation to affected line
- Trend charts
 - Uses modern charting library
 - Rendering is done on the client
 - More to come soon
- Improved overall user experience
- Parsers do not depend on Jenkins API
- Add Git author and commit blame

Next steps

Coding

- Responsive UI (screen size)
- Configurable trend charts
- Analysis dashboard
- More dashboard view portlets
- Server side paging
- Internationalization

Research

- New warnings computation
 - Based on Abstract Syntax Tree
 - Based on plagiarism algorithm
 - SCM delta analysis
- Technical debt for warnings
 - File change frequency
 - Number of authors
 - Size of file



Static Analysis Plugins White Mountain Release

End