CODING STANDARDS CASE STUDY

With Open Source Tools

phaonmneal pweor of satitc anlasyis.

100

times expensive fix defect after production.

-IBM SSIR

IMPORTANCE

- Time dedicated to testing, debugging & verification is half of entire development cycle.

TARGETTED IMPORTANT PROBLEMS

- Already existing large codebase maintenance & improvements.
- Consistency in code standards & customizations of code standards.
- Identifying possible optimizations & duplication of code.
- Verification of code at design level itself with multiple criteria.
- Analysis of Code Standardization trends.

Tools

With their specialization

ToolSet-1

CheckStyle
Pmd
SpotBugs
CheckerFramework

Alternatives

SonarLint
SonarQube

ToolSet-1

For coding standards defined by users.
Optimizations, Unused & Duplicate Code.
Potential Bug Detector.
Code verification to remove runtime errors.

Alternatives

Coding Style, bug and code issue detection.
To generate reports, analyze trends at server.

What to use out the following tools ?

Analyze Fundamental Workings

CHECKSTYLE

Important Functional Checks

Categories Of Checks :

Annotations related checks like inline annotations, missing annotations.

Block Checks like nested, empty blocks, empty catch.

Class design checks like MutableExceptions, OnTopLevelClass.

Coding Standards like Declaration Order, Covariant Equals, EqualAvoidNull, HiddenField, Illegal Catch, MissingSwitchDefault, MutlipleStringLiterals, SimplyBooleanExpression, StringLiteralEquality, SuperFinalize.

Header define header properties.

Imports AvoidStarImport, AvoidStaticImport.

Javadoc checks like AtclauseOrder, JavadocMethod.

Use with :

- **Command line**
- **Eclipse Plugin**

For Class Design, Method Design, Code Layout & Formatting issues.

CHECKSTYLE

Important Functional Checks

Categories Of Checks :

Metric BooleanExpressionComplexity, DAC, CyclomaticComplexity.

Miscellaneous like Indetations, NewLineEOF.

Modifier checks like ModifierOrder, RedundantModifier.

Naming Conventions like Method, Abbreviations, Static, ConstantName, MemberName, ParameterName.

Regexp like regex expression exists.

Sizeviolations like FileLength, LineLength.

Whitespaces like FileTabCharacter.

Addition/Removal of checks allowed.

PMD

Important Functional Checks

- Common programming flaws like unused variables, empty catch blocks, unnecessary object creation. Additional CPD detector.
- pmd & cpd are separate tools to be used on command line interface, UI based interaction Eclipse Plugin.
- Custom rules additions in our own custom xml files and inclusions of
- Best Practices, Code Style, Design, Documentation, Error Prone, Multithreading, Performance and Security.
- Best practices is to use custom rules & Suppress Warnings in code if specific code analysis not needed.
- Works for Java as well as for JSPs.

Use with :

- **Command line**
- **Eclipse Plugin**

Coding Standards Antipatterns
CTRL+C/V detector

SPOTBUGS

Important Functional Checks

- **Bad Practice** like include hashcode & equals method.
- Comparison of String using == or !=
- Class define hashCode() but not uses equals() from Object. Or vice-versa.
- Class name capital, confusing method names differ only in capitalization.
- **Correctness** non-null fields null, infinite loops, method argument might be null, Invalid Min & Max method combination.

- **Other categories are multithreaded, vulnerability, performance, security, dodgy code.**

Use with :
- **Command line**
- **Eclipse Plugin**

Potential Bugs

CheckerFramework Type Analysis

This tool verifies the code, if no error pointed by this tool against particular system then no error exists.

Superior behavior then Findbugs. It is a bug detector not remover.

Important General Type Systems

- **Nullness Checker** verifies for NullPointerExceptions, Initialization and raw types errors. It also consist of MapKeyChecker.
- **Interning Checker** verifies at all places correct use '==' is there instead of equals().
- **Lock Checker** for checking concurrency errors and check guards.
- **Tainting Checker** for checking untrusted SQL queries doesn't get parsed.
- **I18n Format Checker** for checking that formatted strings are correctly formed.
- **Regex Checker** valid regex expression are checked.

- **CLI usage**
  - With javac
  - With jars

Runtime Exception caught at Compile time.

Annotations are required for this.

- CLI usage
  - With javac
  - With jars

Runtime Exception caught at Compile time.

Integration with Jenkins for reports.

SonarLint & SonarQube

This tool verifies the code, if no error pointed by this tool against particular system then no error exists.

Superior behavior then Findbugs. It is a bug detector not remover.

SonarQube Features

- **Unit testing and code coverage by these test cases.**
- **Duplicity, design/architecture & Code Complexity.**
- **Historical Reports for, code quality improvements.**
- **Long reports & time consumption is a disadvantage. But, on the fly feature is a help.**

Static Analysis

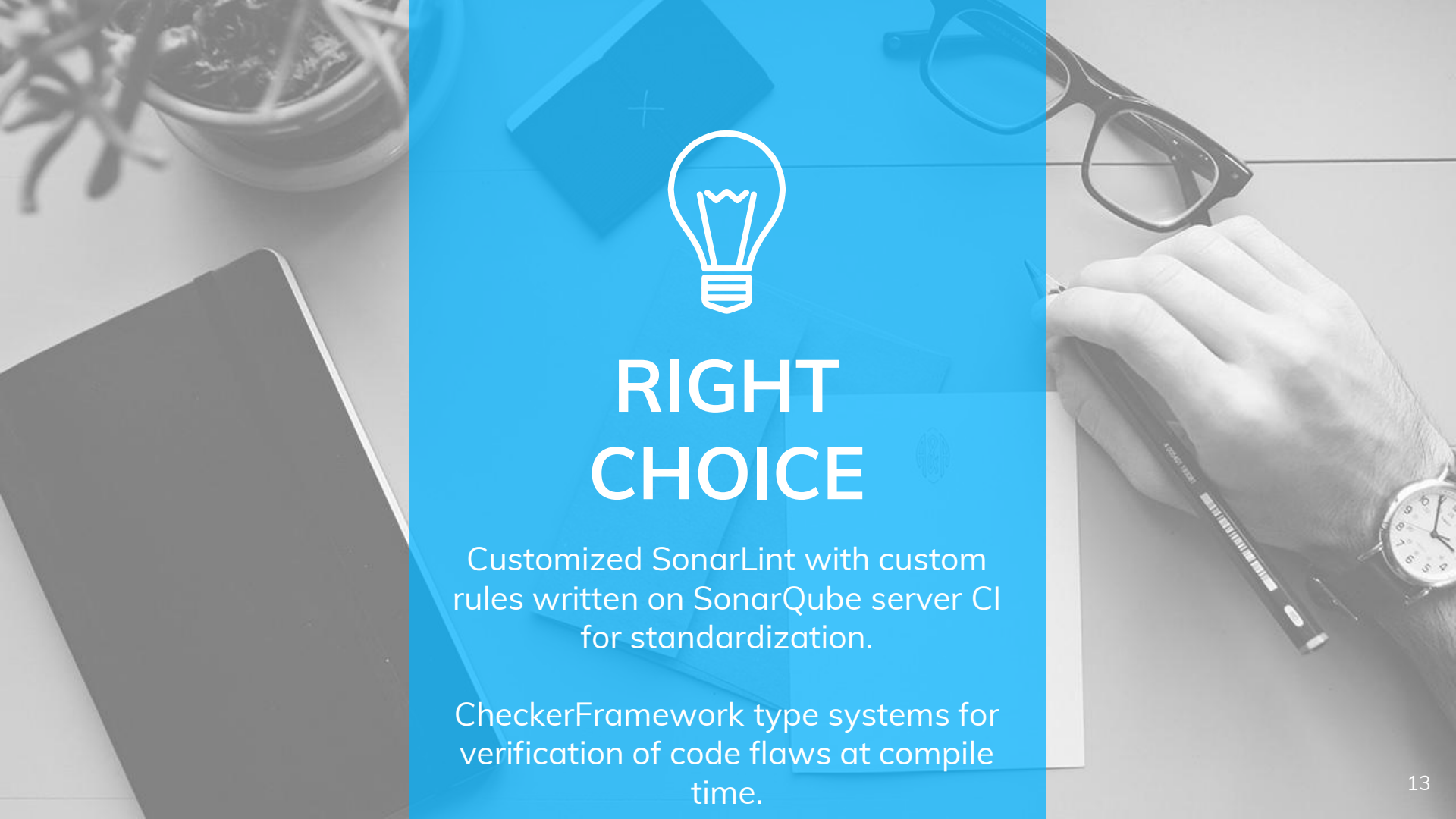
- A violation might not result in failing build or will be an error at all.
- Tools are needed to be modified & used according to our need.
- Too many rules, lead too much verbose, false positives.

Conclusion – Project Specific

- Need to check against few coding standards for java and jsp programs.
- See the most critical coding properties being it vulnerabilities, bad practices and few potential bugs to detect.
- Verification of code is a plus for static analysis of code.

Introduction : Tools & Major Feature of Each

- **CheckStyle**: Sun suggested standard coding practices, Javadoc enhancements. Code layout & Formatting issues with minor design issues & coding metrics. Command Line & Eclipse Plugin.
- **PMD**: For strong design analysis, code optimizations possible, coding metrics & copy-paste issues. Method suggestions, unnecessary instantiations. Command Line & Eclipse Plugin. Java, JSPs, JS.
- **Spot Bugs**: Finding bugs like NullPointerException, bad practices in code. Vulnerability, Performance & security analysis is there. Eclipse & CLI is there. But customizations are hard & not being maintained.
- **CheckerFramework**: Catches runtime exceptions at compile time, like NullPointerException, Security SQL Issues, International String Format Issues with programmers written checks on code. CLI is there only.
- **Sonarlint**: Tools points out possible bugs, design issues, complexity issues & complies with OWASP, SANS Top 25, CERT etc. Standards. Good tool for web applications. Eclipse Plugin, with connectivity with SonarQube available.
- **SonarQube**: Also needs sonar-scanner with to scan code & generate reports. Covers wide number of languages, can sync rules for multiple Sonarlints. Integration with Jenkins is possible as plugin.



RIGHT CHOICE

Customized SonarLint with custom rules written on SonarQube server CI for standardization.

CheckerFramework type systems for verification of code flaws at compile time.

“Quality is never an accident. It is always an intelligent effort.”

Thanks!

Any questions?

You can find me at:
linkedin.com/in/ashishrana160796