Automated Codebase Analysis

Simon Harris
simon@redhillconsulting.com.au

James Ross
james@thoughtworks.com

Agile Programming Practices

- Deliver business value first
- YAGNI – You Ain't Gonna Need It (Yet)
  - Simplest Thing That Could Possibly Work(tm)
- Pair Programming
- Automated builds
- Automated tests
- Red-Green-Refactor
- Check-in Often

Codebase Analysis in Context

- Perform code reviews
- Expose hidden design flaws
- Identify candidates for re-factoring
- Assess open source and enterprise components for re-usability
- Assess unfamiliar codebases

Analysis Methods

- Code Cop
  - Difficult to check every check-in
- Static analysis
  - Source (and executable) code
  - Heuristics
  - Often syntax tree only
  - Cannot fix the code itself
- Dynamic analysis
  - Requires a **working** build environment

Static Analysis Tools

- Open Source
  - Checkstyle
  - PMD
  - CPD
  - JDepend
- Commercial
  - Simian (free for Open Source/Education Projects)
  - PASTA (OptimalAdvisor)

Continuous Integration

- Integration with Ant
- Integration with NAnt
- Break the build
- Publishing results for management
- Incremental improvement over project lifecycle

Types of Checks

- Programming 101
  - Formatting, line-Length, whitespace, etc?
  - Basic metrics
  - Duplicate code
- Advanced
  - More metrics
  - Design heuristics
  - Package Dependencies

Sample Checks

- Cyclomatic complexity
- NPath complexity
- Executable statement count
- Final fields
- Duplicate code
- Missing abstraction

Demonstration

- JDK 1.4.2_05
  - 3,975 files
  - 1.2 million lines of raw source code
  - 350,000 significant lines of code
  - 108,000 duplicate lines of code in 1,789 files
  - Max executable statement count: 255
  - Max cyclomatic complexity: 212
  - Max npath complexity: 1,077,581,313

The Future of Codebase Analysis

- Look for higher level abstractions
  - Design patterns, Anti-patterns
- More expressive rule language
- Suggest improvements
- Maybe one day even make the necessary changes for you

Summary

- Highly customizable
- Save time and money spent on manual code reviews
- Codify your standards (tests!)
- Rapid evaluation of quality
- Reviews based on facts not opinions
- Raise the bar on quality as the project progresses
- Constraints promote good design
- You still need Good People(tm)

References

- Checkstyle - http://checkstyle.sf.net
- Simian - http://www.redhillconsulting.com.au/products/simian
- JDepend - http://www.clarkware.com/software/JDepend.html
- JOODI - http://sf.net/projects/joodi
- PMD and CPD - http://pmd.sf.net
- FindBugs - http://findbugs.sf.net
- PASTA - http://javacentral.compuware.com/pasta