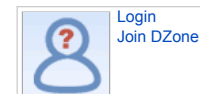




Home Zones Library Refcardz Forums Links Snippets



Home

## 7 Code Review Tools and Techniques

Submitted by Mitchell Pronsc... on Mon, 2010/04/19 - 11:00pm

DZone



Tags: Code Review static analysis Issue Tracking

Compared to other development activities, code review tends to have less published [resources available](#) to help developers stay abreast of the latest review methods, tooling, and best practices. Arming developers with the knowledge of code review tools and techniques can increase efficiency and reduce the painfulness of the code inspection process. Making checklists, delegating roles, and implementing reading techniques help developers conduct more focused code reviews, resulting in higher code quality. Code review and inspection doesn't have to be manual in all areas. Compiler warnings, static analysis software, and issue/defect tracking can make reviews and inspections more efficient, structured, and semi-automated. A code review toolset is also a blessing to the reviewer who hates to read code that hasn't been properly prepared.

### We Recommend These Resources

- [The Automation Evolution: An analysis of the Evolution of Automation in Software Quality Management](#)
- [Getting Requirements Right: Avoiding the Top 10 Traps](#)
- [The IBM Rational Jazz Strategy for Collaborative Application Lifecycle Management](#)
- [Best Practices for Building Linux Installers](#)
- [FREE Agile Development Poster](#)

**Preparation, Inspection, Rework, and Follow-up** are generally the four stages of code review. In the Preparation stage, you determine inspection/review targets and the roles for each participant. Reading techniques should also be specified at this stage. In the Inspection/Defect Detection stage, each inspector looks for defects according to their assigned role and perspective. In this stage, compiler warnings, static analysis tools, review tools, and defect tracking tools are put to use. These tools organize defects for the Rework stage, which is when reviewers go back and fix defects. Finally there's the Follow-up stage where corrections are verified and reviewers make sure that new bugs haven't been introduced. Defect tracking and review tools are used here too. That was a quick run-through. Now let's look at some of the techniques and tools in detail for each of these stages.

### Roles

**Reviewer/Inspector:** All members assume this role. Their goal is to detect and identify defects.

**Moderator:** The moderator is responsible for leading the review group and keeping them focused on their goals.

Moderators set up the environment and regulate the discussion to keep reviews focused. They also manage the time.

**Recorder:** This person is responsible for documenting the issues that developers find. The recorder must summarize the list of defects found.

**Author:** The author is a developer who wrote the code being inspected. It is their responsibility to correct the defects once they are found.

**Organizer:** The organizer develops a plan for development and review. They determine the time schedule for the inspection/review.

**Reader/presenter:** The reader explains the product being inspected/reviewed on behalf of the author.

Search

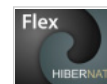


### About the author



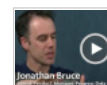
**First Name:** Mitchell  
**Last Name:** Pronschinske  
**Zone Leader**  
**Posts:** 527  
**Joined:** 10/05/2009  
[View full user profile](#)

### Spotlight Features



#### Combining Flex & Hibernate

Learn how Flex & Hibernate can provide more than just a viable persistence option for your Rich Internet Applications.



#### The Future of Java Persistence

Jonathan Bruce speculates on the future of Java data persistence in light of Oracle's Sun acquisition and the advent of NoSQL.



#### ColdFusion Performance Tips

Mike Brunt shows you how to architect your ColdFusion applications for maximum performance and scalability.

*Note that each developer can have more than one role, and each role can have more than one developer.*

## Reading Techniques

The most common, and probably the best reading technique reviewers can use, is a simple checklist. These lists contain questions that help developers focus on certain areas in the code. Questions can be geared towards novice or expert developers and the questions might, for example, ask: "Have resources been properly freed?" "Will performance suffer from excessively nested loops?" "Are the data structures used appropriately, taking into consideration trade-offs between performance and memory?"

Perspective-based reading is another reading technique that assigns a certain point of view to each reader. Perspectives such as "user" or "tester" will determine how a reader looks at the code and what areas they focus on. Conducting the review from multiple perspectives is a proven way to have comprehensive code inspections/reviews. Someone with the user perspective will read the code with use cases in mind while someone assigned to the tester perspective will read through the code thinking about test cases that could prove the code works properly.

## Issue Tracking, Static Analysis, and Code Review Tools

Although they're usually not as comprehensive as static analysis tools, most compilers have the ability to issue warnings for statements they think may cause bugs, even though the statements don't cause a compilation error. Code review tools come with useful productivity features for browsing the source code. They let reviewers make annotations at any location within the code and then the tool lists those annotations for quick, organized reference. Issue tracking systems bring another layer of organization to code review by recording and tracking detected defects. Code review and issue tracking tools are not limited to source-code review alone. They can also be used for the reviewing project plans, requirements definitions, specifications, design documents, test plans, operations manuals, and user manuals..

*Jupiter Eclipse plugin for code review*

## Popular at DZone

[NetBeans IDE 6.9 Stabilization Update](#)

[Java platform roundtable, Spring 2010](#)

[Java 7 : New I/O features with JSR 203](#)

[Getting To Grips With the Equinox Console](#)

[Design Patterns Uncovered: The Memento Pattern](#)

[Bruce Eckel is Wrong on Checked versus Runtime Exceptions](#)

[LiveCycle Data Services 3.1 and BlazeDS 4 Available](#)

[See more popular at DZone](#)

[Subscribe to the RSS feed](#)

[Terms of Service](#) - [Privacy](#) - © 1997-2009, DZone, Inc.

**Phase Selection**

**Review Table**

Severity	Type	Summary
Normal	GG-01:F...	[takuyay] Should I check the most active file with the most r
Normal	Suggestion	Inner class can be eliminated
Normal	Suggestion	Make reduction function parameter case-insensitive
Normal	Defect	Non-constructive exception message
Normal	Suggestion	12 final strings duplicated in source and test code
Minor	Suggestion	Incomplete or unnecessary comment
Normal	GG-01:F...	Final strings are duplicated in source code and test code
Minor	Question	Should inner class be public?
Normal	Suggestion	Refactor method name - misleading (2)

**Review Editor**

Assigned To: takuyay Resolution: Unset

Annotation:

throw new ReductionException("ReviewIssueReducer can take 2 or 3 options.");

The exception message will be returned to end user (by telemetry infrastructure) when user specifies wrong parameter for the reduction function. Try to use a more user friendly error message.

'Source code static analysis tool' is a term that generally refers to a tool that extracts information from the source code without executing the program. The analysis part includes calculating source-code metrics, using defined patterns to detect potential bugs, and finding instances where coding conventions and rules were broken. Metrics include the number of function or method arguments, the number of function or method calls, the number of branches and loops, and the number of source code lines. Reviewers benefit from this analysis because it pinpoints areas of the code that are fault-prone. Static analysis tools also find resource leaks.

Depending on the sophistication of the static analysis software, there are some tools that will detect areas in source code that are likely to access null pointers or addresses beyond array boundaries. Certain static analysis tools can also estimate

the execution flow of the code. State-of-the-art static analysis tools are able to identify the location and type of design patterns in the code. This can help reviewers find potential defects that are common in specific design patterns.

The screenshot shows the Eclipse IDE with the Klocwork static analysis tool. The main editor displays the following Java code:

```

141 { line.hasOption( "d" ) } {
142     try {
143         String fileName = line.getOptionValue("d");
144         pathPart = fileName.substring(0, fileName.lastIndexOf(File.separ
145         BufferedReader in = new BufferedReader(new FileReader(fileName)
146         String s;
147         while ((s = in.readLine()) != null) {
148             s = s.trim();
149             // # - this signals the beginning of the comment in the inp
150             if (s.length()==0 || s.startsWith("#")) continue;
151             files.add(s);
152         }
153         in.close();
154     } catch (Exception e) {
155         e.printStackTrace();
156         System.exit(1);

```

The Klocwork Details window shows the following error:

```

RLK.IN (Error) More information
Input stream 'new FileReader(...)' is not closed on exit.
Traceback:
C:\Documents and Settings\pmurphy\My Documents\Solo\csvtosql_jdk50\sr
ExtendedConsoleMain.java:145: [source] 'new FileReader(...)'
ExtendedConsoleMain.java:147: 'java.io.IOException' is thrown by read

```

The Klocwork Issues window shows a table of 97 items, grouped by None, sorted by Severity, then by Resource. The table has the following columns: Description, Resource, Location, Severity, Status, and State.

Description	Resource	Location	Severity	Status	State
RLK.OUT: Output stream 'new FileWriter(...)' is not closed on exit.	SqlFileWriter.java	43	Error (3)	Analyze	New
RLK.OUT: Output stream 'new FileWriter(...)' is not closed on exit.	SimpleGui.java	108	Error (3)	Analyze	New
RLK.SQLOBJ: Sql object 'stmt' is not closed on exit.	JdbcWriter.java	90	Error (3)	Analyze	New
RLK.IN: Input stream 'url.openStream()' is not closed on exit.	JFrameHelp.java	37	Error (3)	Analyze	New
RLK.IN: Input stream 'new FileReader(...)' is not closed on exit.	ExtendedConsoleMain.java	145	Error (3)	Analyze	New
RLK.OUT: Output stream 'raf' is not closed on exit.	Disk.java	45	Error (3)	Analyze	New
RLK.OUT: Output stream 'raf' is not closed on exit.	Disk.java	65	Error (3)	Analyze	New
RLK.OUT: Output stream 'raf' is not closed on exit.	Disk.java	84	Error (3)	Analyze	New
FIN.NOSUPER: Implementation of the finalize() method should call su	Disk.java	133	Unexpected (4)	Analyze	New
ECC.EMPTY: Empty catch clause	StringField.java	78	Investigate (5)	Analyze	New
ECC.EMPTY: Empty catch clause	DB2StringField.java	78	Investigate (5)	Analyze	New

*Klocwork's Insight static analysis tool*

## Conventions

Line breaks, treatment of whitespace (tabs and spaces), and the formatting of conditional statements are conventions associated with readability. Conventions for extensibility focus on things like using defined constants instead of explicit numbers so you can change the value of a constant to change all instances of that constant throughout the code. Reliability conventions encourage developers not to use recursive functions or `goto` statements because they often make the code complex and buggy. `goto` statements and recursive function calls may be supported in a language, but it's better if you can find other methods so your code is easier to understand.

Are there any other best practices in your experience with code review? Suggest some resources.

Your rating:

Average: 4 (1 vote)

[Login](#) or [register](#) to post comments 3256 reads [Printer-friendly version](#)

*(Note: Opinions expressed in this article and its replies are the opinions of their respective authors and not those of DZone, Inc.)*

## Comments



Steve Banks replied on Tue, 2010/04/20 - 5:18am

Did you mean Jupiter instead of Juniper? :-)

I'd be interested to see some more detailed checklists for these reviews. Would anyone care to share?

[Login](#) or [register](#) to post comments



Gregg Sporar replied on Tue, 2010/04/20 - 8:50am

We have a free book on peer code review inspection techniques: <http://codereviewbook.com>. Full disclosure: we sell a code review tool. The advice in the book, however, is useful whether you buy our tool or not.

[Login](#) or [register](#) to post comments

## Comment viewing options

Flat list - expanded  Date - oldest first  30 comments per page  [Save settings](#)

Select your preferred way to display the comments and click "Save settings" to activate your changes.