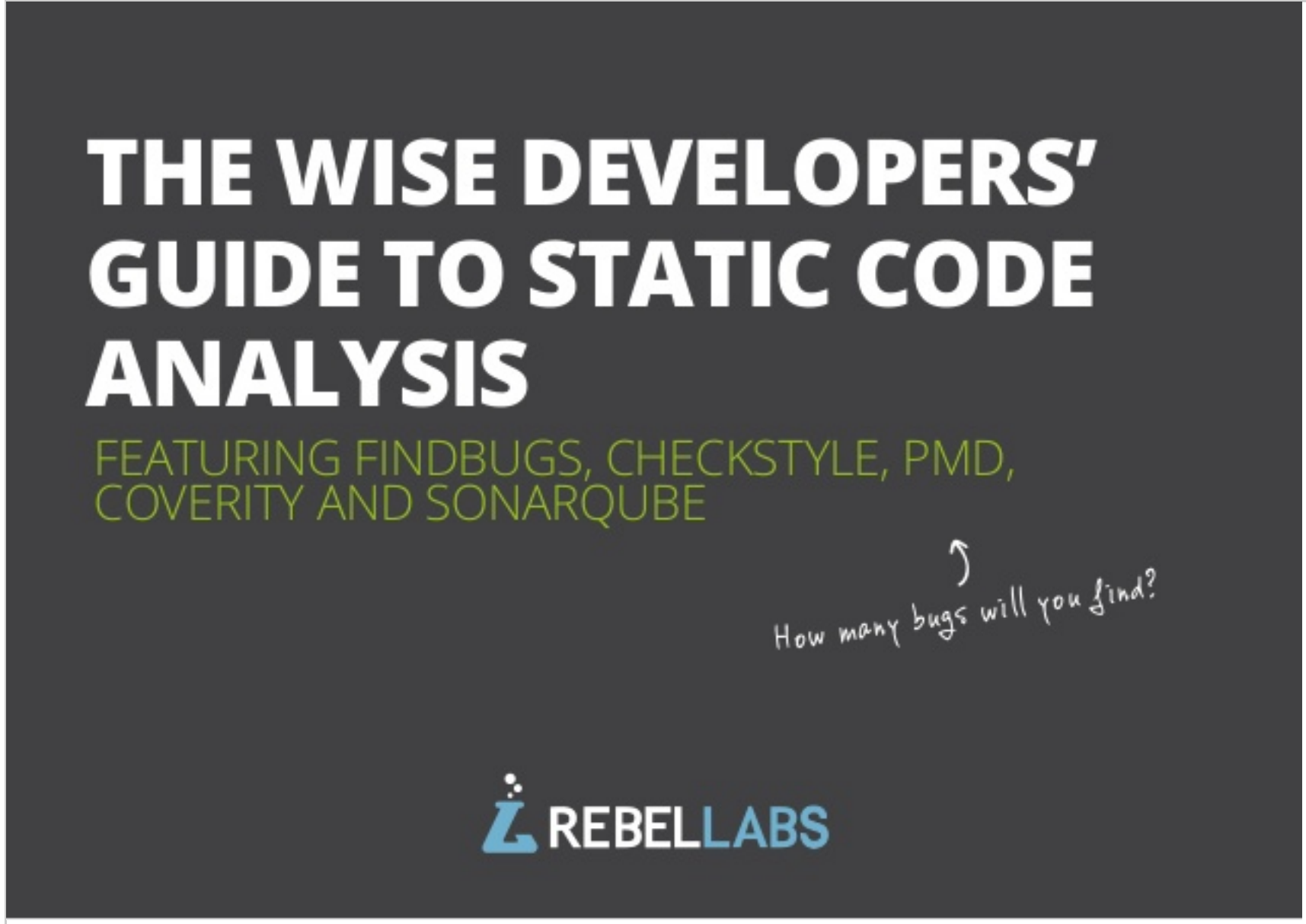
The Wise Developers' Guide to Static Code Analysis featuring FindBugs, Checkstyle, PMD, Coverity and SonarQube

April 17, 2014 | Oleg Shelajev | 4 comments

REPORTS POSTS

The Ultimate Java Web



Chapter I: Welcome to static code analysis, that thing you aren't doing

> *"The quality of your code is a weak spot in almost every software project you'll ever touch. This is because ongoing development ensures that even the bits you were once proud of become, over time, first less elegant, then rough, and finally incomprehensible."*
>
> — **Oleg Shelajev**, Java Developer/Author

Read it later! (pdf download)

Why should we monitor and fix code quality issues?

If we start at the very beginning, it would be with what we know about developers and their use of the tools and practices used to analyze code quality. There are a few things we found out about how developers think about code quality analysis, from Developer Productivity Report 2013 – How Engineering Tools & Practices Impact Software Quality & Delivery, which surveyed just over 1000 developers. Here's what we saw:

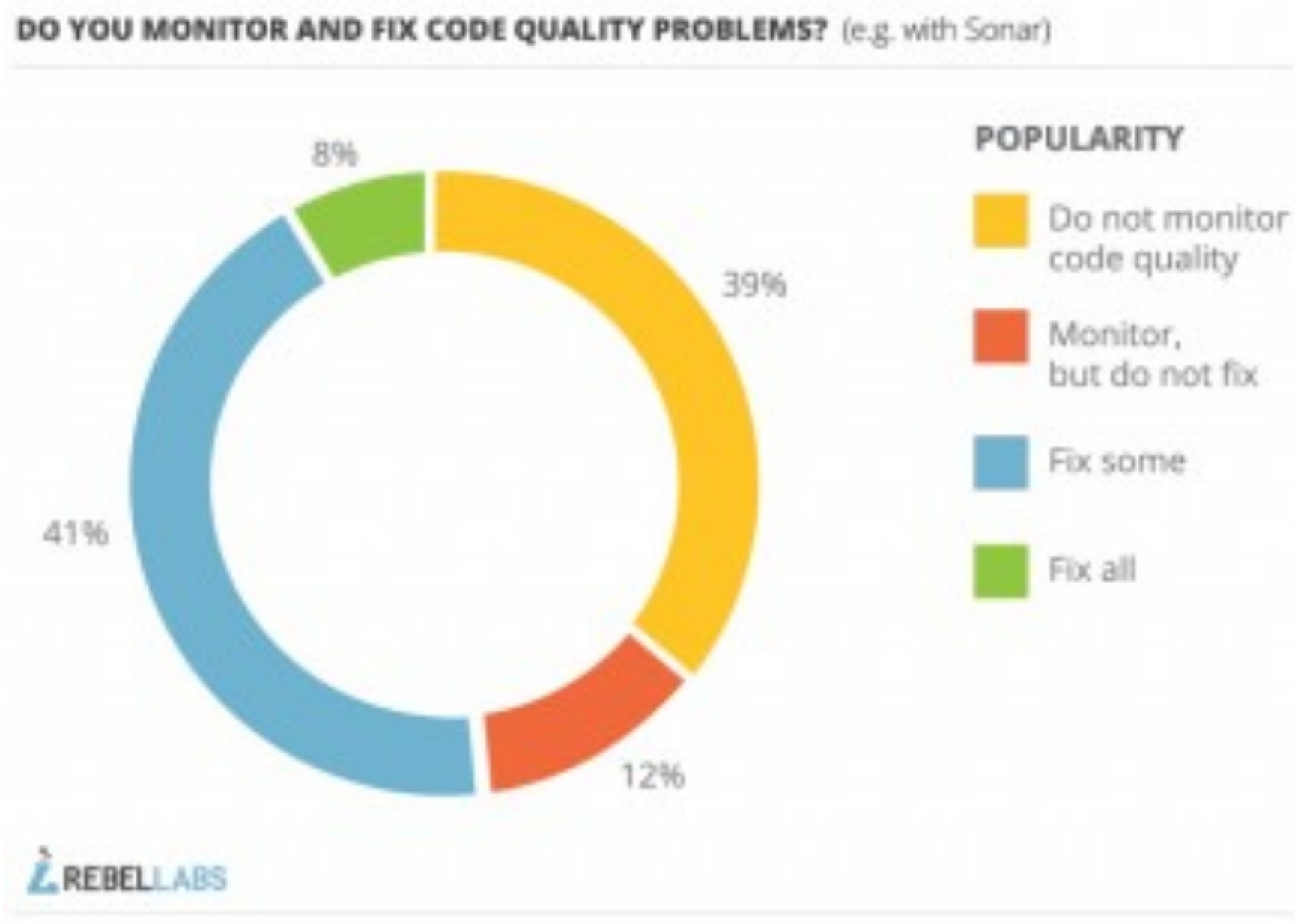
Frameworks Comparison: Spring MVC, Grails, Vaadin, GWT, Wicket, Play, Struts and JSF
30 July 2013

The Great Java Application Server Debate with Tomcat, JBoss, GlassFish, Jetty and Liberty Profile
21 May 2013

Java 8 Revealed: Lambdas, Default Methods and Bulk Data Operations
25 June 2013

Tags

continuous delivery 23
developers 37
development 49
DevOps 28
eclipse 33
java 178
java 8 44
Java EE 43
JBoss 31
jenkins 23
JRebel 135
JVM 40
LiveRebel 51
productivity 56
RebelLabs 404
spring 46
Tools 37
vJUG 42
xrebel 39
ZeroTurnaround 432

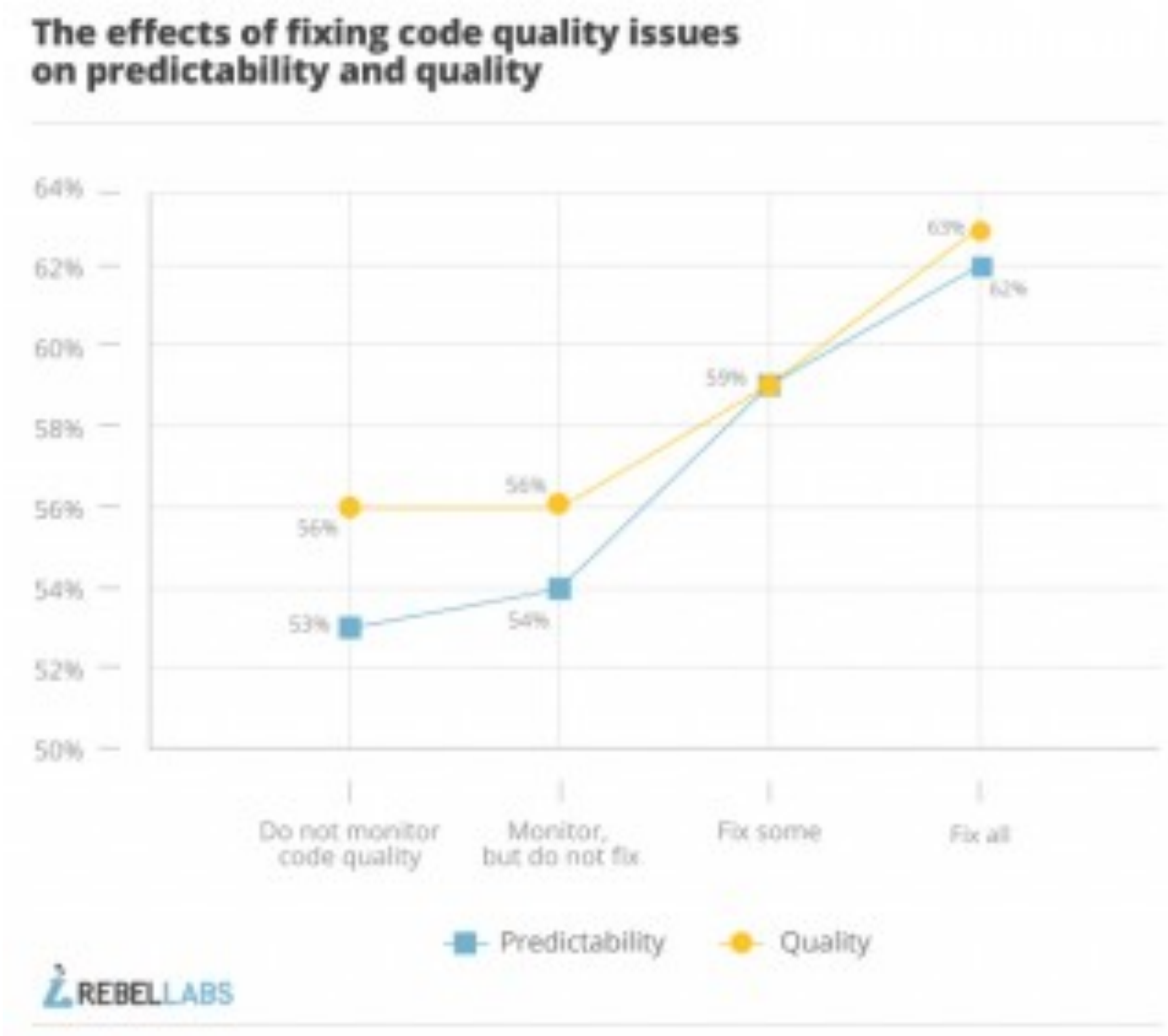


- **39% of developers don't monitor code quality at all**
- **12% of developers monitor, but don't fix issues**
- **41% of developers fix some code quality issues**
- **8% of developers fix all code quality issues**

So, code quality analysis is not a terrible popular category to start with–which is probably one reason that most apps, even the best of them, suffer from creeping bugs and errors at some point. And do you know what else we found when we correlated these answers with others in the report?

Fixing code quality issues has a significant effect on, well, the overall quality of your code, as well as your ability to accurately predict when the software can be delivered to end users. Boom!

If the choice is between doing nothing and fixing all code quality issues that are identified, this is the analysis that emerged from the responses provided by the sample population:



- **Developers reported up to 9% better predictability for app delivery**
- **Developers reported up to 7% better app quality**

The point here is, monitoring and fixing code quality issues is something that is **proven to raise the quality of your application AND your ability to deliver that application to stakeholders on time**. But it's clear that the vast majority of developers aren't taking full advantage of tools designed to improve app quality.

Perhaps most developers don't know where to start. For developers, the main point can be summed up in one sentence:

> How are you supposed to integrate your tool of choice into your development cycle so it can find relevant issues and allow the team to fix them?

There are many aspects of "code quality" that we can sink our teeth into, but we've decided that Static Code Analysis is an essential building block in your pyramid of tools that help improve the quality of your code. However, developers are using tools that fit into other categories as well, such as:

Dynamic Code analysis

The simplest difference between Static and Dynamic analysis tools is that the former runs in the development environment and the latter needs to be active during the runtime of the application under analysis. Typical dynamic code analyzers profile your system and monitor its health. Both execution time and memory usage profilers, figuring out number of database transactions per request, the average size of an user session object, etc. require the system to be under a load comparable with the intended in production environment. Dynamic analysis tools often

instrument the code to add tracing of method calls, catching and notifying about exceptions, and any other statistics they collect.

Profilers

Performance is a magical term that never fails to generate interest. Figuring out why your system is slow and how to make it faster is a rewarding exercise. Combine this with the fact that you can continue optimizing forever (as something will always be a bottleneck), performance-related tasks are always picked first by developers. It just sounds so cool, and it's also measurable too.

Memory tools

Most of existing tools that deal with memory management either provide some high level statistics in real-time, like telling you the size of the heap and the number of classes loaded into the JVM, or work in an offline mode feeding on some traces produced during a run. Garbage collector's logs, object allocation rates, ability or inability to refresh the memory taken by the classloader of your web-application, these are questions usually attacked with a tool analyzing your application's memory behavioral patterns.

Monitoring tools

Monitoring tools are known to everybody, often these are the last man standing before a service goes offline because of some resource limits.

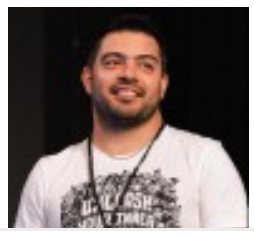
Naturally, there are a lot of questions to ask before you start to use any of the tools we discuss later, so in this report we show you what aspects are important to consider when getting started.

Download the PDF

1 2 3 4 5 6 7 8 9 10 Next »

Authors






OLEG ŠHELAJEV

Developer Advocate

Oleg Šelajev is an engineer, author, speaker, lecturer and advocate at ZeroTurnaround. He spends his time testing, coding, writing, giving conference talks, crafting blogposts and reports. He is pursuing a PhD on Dynamic System updates and code evolution. Oleg is a part-time lecturer at the University of Tartu and

:us

checkstyle, code quality, coverity, findbugs, Maven, pmd, RebelLabs, sonaqube, sonar, static code analysis

Company

- Careers
- Contact Us
- Our story

Resources

- JRebel White Paper (PDF)
- Devs Productivity Report
- IT Ops & DevOps Productivity Report
- Rocket powered Java development
- XRebel ROI White Paper