DON’T ANALYZE AND IGNORE, AUTOMATE AND ACT!

Agile 2015 Conference
Washington D.C. USA

Scott Dillman
Senior Director Analysis & Design
Travelers Insurance
August 6, 2015

DISCLAIMER: The opinions and views expressed in this presentation are those of the author and do not necessarily reflect the position of Travelers.
Agenda

**Take back control**
How to analyze and act

**Let’s review**
Automatically trigger code reviews

**Too close for comfort**
Monitor for changes to sensitive code

**Say what?**
Monitor for keywords

**Take note**
Create automated release notes

**Reach for the stars**
More ways to analyze and act
Take back control

- Acknowledge your limitations
- Narrow your focus
- Decide what is most important to you
- Automate and act with precision
Let’s review

• If code reviews are a good thing…
• Review everything?
• What about your limitations?
• How do you decide what you review?
• Automate the review-creation process, and review what matters most
Automatically trigger code reviews

1. Automated Build performs some type of analysis that you wish to act on.
2. Automated Build evaluates the results of the analysis, and determines if the type of change warrants a code review.
3. If appropriate, Automated Build triggers the creation of a code review either by invoking a custom application, or contacting the code-review system directly.
4. A custom application can take in command-line arguments from the Automated Build, and contact the code-review system using technologies that may be more difficult to leverage from the Automated Build (ex. SOAP/REST/SQL).
5. Most modern code-review tools expose APIs to allow code reviews to be created in an automated fashion.
Too close for comfort

• If you can’t monitor and review everything...
• Identify which areas of your code base are most “sensitive”
• Some might contain critical modules or algorithms
• Others might be fragile with frequent defects
• Set up your automated build to monitor for changes to these sensitive areas of code
• Automatically create a code review when sensitive code changes
Monitoring Sensitive Code

**Local Build**

1. Automated Build leverages source-code repository to determine which files have been modified.

2. Features of the repository can be used to identify the modified files. For example, with Subversion, you can use ‘svn status’. Check in the list of modified files.

3. Automated Build invokes a custom application to evaluate what has changed.

4. Custom application compares the list of modified files against the list of files which have been predetermined to contain sensitive code.

5. If any sensitive files have changed, then the Automated Build notifies the developer.

**Build Server**

1. Automated Build leverages source-code repository to determine the revision information for the check in.

2. Features of the repository can be used to identify the revision. For example, with Subversion, you can use ‘svn info’.

3. Automated Build invokes a custom application to evaluate what has changed.

4. Custom application compares the list of modified files against the list of files which have been predetermined to contain sensitive code. If any sensitive files have changed, then the application creates a code review, associating the modified files and revision information with the review.
Say what?

• If you can’t monitor and review everything...
• Identify keywords or phrases from your code that might indicate something you should look at
• For example, the use of a certain method or class that has caused problems in the past
• Set up your automated build to monitor for code changes that include any of these keywords
• Automatically create a code review on keyword matches
Automated Keyword Monitoring

**.NET**

1. Automated Build launches StyleCopRun application.
2. StyleCopRun application runs StyleCop on source code. Custom rule checks for instances of specified keywords defined in Keywords.xml. All StyleCop violations are output to KeywordViolations.xml.
3. Automated Build invokes a custom application to evaluate the results.
4. Custom application compares KeywordViolations.xml to IgnoredKeywordViolations.xml to see if one or more keyword violations were found that are not ignored. In a local build, the developer is notified of results. In a server build, a code review is created for any violations.

**JAVA**

1. Automated Build executes Ant Checkstyle task.
2. Checkstyle task runs Checkstyle on source code. Custom rule checks for instances of specified keywords defined in Keywords.xml. All Checkstyle violations are output to KeywordViolations.xml.
3. Automated Build invokes a custom application to evaluate the results.
4. Custom application compares KeywordViolations.xml to IgnoredKeywordViolations.xml to see if one or more keyword violations were found that are not ignored. In a local build, the developer is notified of results. In a server build, a code review is created for any violations.
Take note

- Once you start thinking differently, new opportunities present themselves
- Why create release notes manually when your automated build can do it for you?
- If your build is already monitoring what changed, simply associate a business context to those changes
- Your automated release notes will save you time, and increase accuracy and completeness
- QA will thank you!
Automated Build leverages source-code repository to determine which files have been modified.

2 Features of the source-code repository can be used to identify the modified files. For example, with Subversion, you can use ‘svn status’. Save the list of modified files.

3 Automated Build invokes a custom application to evaluate what has changed, and determine which business features/capabilities are affected.

4 Custom application uses a mapping file to map code modules to business capabilities, allowing it to determine which features/capabilities have been affected by the changes. Using the results of the processing, it generates release notes highlighting which business features/capabilities have been impacted by changes, and therefore may require testing.
Reach for the stars

There are many other ways to automate and act:

• Monitor specific thresholds from code coverage or static-code analysis tools, and either fail the build or trigger code reviews in response
• Create/maintain service catalog
• Connect any type of analysis tool to your build, and let your imagination flow…
Conclusion

Remember:
• Don’t analyze everything and ignore the results
• Determine a small number of things you care most about
• Analyze those, and set your build up to automatically act on them

“You are the embodiment of the information you choose to accept and act upon. To change your circumstances you need to change your thinking and subsequent actions.”

Adlin Sinclair¹

References