Code Reviews Do Not Find Bugs
How the Current Code Review Best Practice Slows Us Down

Jacek Czerwonka, Michaela Greiler, Jack Tilford
{jacekcz, mgreiler, jtilford} @ microsoft.com
Tools for Software Engineers team, Microsoft Corp.

(in collaboration with Christian Bird, Microsoft Research)
You must publish your review before other reviewers will be able to see it.

```xml
  <Build Include="$(SrcRoot)\Common\Database\Schemas\loader\Stored Procedures\GetInvalidConstraints.proc.sql">
    <link>Schemas\loader\Stored Procedures\GetInvalidConstraints.proc.sql</link>
  </Build>

  <Build Include="$(SrcRoot)\Common\Database\Schemas\loader\Tables\Constraints.table.sql">
    <link>Schemas\loader\Tables\Constraints.table.sql</link>
  </Build>

  <Build Include="$(SrcRoot)\Common\Database\Schemas\loader\Tables\Keys\Constraints.PK_KeysConstraints.key.sql">
    <link>Schemas\loader\Tables\Keys\Constraints.PK_KeysConstraints.key.sql</link>
  </Build>

  <Build Include="Schemas\dbo\Tables\Builds\BuildsBuilds.table.sql" />

  <Build Include="Schemas\loader\Tables\ProcessingStatus\BuildBuildsBuilds.table.sql" />

  <ItemGroup>
    <Build Include="$(SrcRoot)\Common\Database\Schemas\Configuration\Stored Procedures\RebuildIndexes.proc.sql">
      <link>Schemas\Configuration\Stored Procedures\RebuildIndexes.proc.sql</link>
    </Build>

    <Build Include="$(SrcRoot)\Common\Database\Schemas\Configuration\Stored Procedures\SetBulkLoadMode.proc.sql">
      <link>Schemas\Configuration\Stored Procedures\SetBulkLoadMode.proc.sql</link>
    </Build>

    <Build Include="$(SrcRoot)\Common\Database\Schemas\Maintenance\Tables\Updates.table.sql">
      <link>Schemas\Maintenance\Tables\Updates.table.sql</link>
    </Build>

    <Build Include="$(SrcRoot)\Common\Database\Schemas\Maintenance\Tables\Keys\Updates.PK_Updates.key.sql">
      <link>Schemas\Maintenance\Tables\Keys\Updates.PK_Updates.key.sql</link>
    </Build>
  </ItemGroup>

  <ItemGroup>
    <None Include="Schema Comparisons\LiveChanges.scmp" />
    <None Include="Post-Deployment\Data\dbo.DepotsMSG.data.sql" />
    <None Include="Post-Deployment\Data\dbo.DepotsWindows.data.sql" />
    <None Include="Post-Deployment\Data\dbo.DepotsWindowsPhone.data.sql" />
    <None Include="Post-Deployment\Data\dbo.DepotsWindowsServices.data.sql" />
    <None Include="Post-Deployment\Data\dbo.DepotsOSD.data.sql" />
    <None Include="Post-Deployment\Data\dbo.DepotsExchange.data.sql" />
    <None Include="Post-Deployment\Data\dbo.DepotsOffice.data.sql" />
  </ItemGroup>
```

Review Properties

Predicted risk of defects: Low

100 %
<table>
<thead>
<tr>
<th>Review Title</th>
<th>Required Reviewers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding support for new types of builds</td>
<td>Michaela Grelier, Jack Tillord</td>
</tr>
</tbody>
</table>

**New feature enabling saving data from new types of builds**

```xml
<Build Include="$(SrcRoot)\Common\Database\Schemas\loader\Stored Procedures\GetInvalidConstraints.proc.sql">
  <Link>Schemas\loader\Stored Procedures\GetInvalidConstraints.proc.sql</Link>
</Build>

<Build Include="$(SrcRoot)\Common\Database\Schemas\loader\Tables\Constraints.table.sql">
  <Link>Schemas\loader\Tables\Constraints.table.sql</Link>
</Build>

<Build Include="$(SrcRoot)\Common\Database\Schemas\loader\Tables\Keys\Constraints.PK_Constraints.pkey.sql">
  <Link>Schemas\loader\Tables\Keys\Constraints.PK_Constraints.pkey.sql</Link>
</Build>
```

```xml
<Build Include="Schemas\dao\Tables\QBuilds.table.sql" />
<Build Include="Schemas\loader\Tables\ProcessingStatus_QBuild.table.sql" />
```

```xml
<ItemGroup>
  <ItemGroup>
    <Build Include="$(SrcRoot)\Common\Database\Schemas\Configuration\Stored Procedures\RebuildIndexes.proc.sql">
      <Link>Schemas\Configuration\Stored Procedures\RebuildIndexes.proc.sql</Link>
    </Build>
    <Build Include="$(SrcRoot)\Common\Database\Schemas\Configuration\Stored Procedures\SetBulkLoadMode.proc.sql">
      <Link>Schemas\Configuration\Stored Procedures\SetBulkLoadMode.proc.sql</Link>
    </Build>
    <Build Include="$(SrcRoot)\Common\Database\Schemas\Maintenance\Tables\Updates.table.sql">
      <Link>Schemas\Maintenance\Tables\Updates.table.sql</Link>
    </Build>
  </ItemGroup>
</ItemGroup>
```
This seems to refer to the wrong location. Should it be "Schemas\Loader\Tables\QBuilds\table.sql"?
This seems to refer to the wrong location. Shouldn't it be "Schemas\Loader\Tables\QBuilds\table.sql"? Makes sense. I made the change.
Changing Task to Definition #2

jacekczerwonka wants to merge 1 commit into master from refactoring

jacekczerwonka commented 2 minutes ago

Changing "Task" to "Definition"
 Hard to read when type name and variable name are the same. Change?
The goals of code reviewing...
Why Code Review?

Find defects
Improve maintainability
Share knowledge
Broadcast progress

⚠️ It is all about the conversation

Figure 3. Developers’ motivations for code review.

Alberto Bacchelli, Christian Bird. Expectations, Outcomes, and Challenges Of Modern Code Review
What do we achieve in practice?
Engineering Data Collection

## Study: Comments Classification

<table>
<thead>
<tr>
<th>Category</th>
<th>Types of issues included</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>Comments, naming, style</td>
<td>22.3%</td>
</tr>
<tr>
<td>Organization of code</td>
<td>Modularity, location of artifacts, new class, duplicate code, size of methods</td>
<td>15.9%</td>
</tr>
<tr>
<td>Solution approach</td>
<td>Alternate algorithm or data structure</td>
<td>8.5%</td>
</tr>
<tr>
<td>Validation</td>
<td>Lack of or improper validation</td>
<td>7.2%</td>
</tr>
<tr>
<td>Visual representation</td>
<td>Beautification, indentation, blank links</td>
<td>6.4%</td>
</tr>
<tr>
<td>False positive</td>
<td>Not an real issue</td>
<td>4.6%</td>
</tr>
<tr>
<td>Defect</td>
<td>Incorrect implementation or missing functionality</td>
<td>2.6%</td>
</tr>
<tr>
<td>Logical</td>
<td>Control flow or logic issues</td>
<td>2.3%</td>
</tr>
<tr>
<td>Support</td>
<td>Configuration support systems or libraries</td>
<td>2.0%</td>
</tr>
<tr>
<td>Interface</td>
<td>Interactions with other components</td>
<td>1.5%</td>
</tr>
<tr>
<td>Resources</td>
<td>Resource initialization, manipulation, and release</td>
<td>1.3%</td>
</tr>
<tr>
<td>Timing</td>
<td>Thread synchronization, races</td>
<td>0.3%</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>25.1%</td>
</tr>
</tbody>
</table>

~50% of all

15% of all


By: Amiangshu Bosu (U of Alabama), Michaela Greiler (TSE), Christian Bird (Microsoft Research Redmond), Characteristics of Useful Code Reviews: An Empirical Study at Microsoft (MSR 2015)
Study: Code Review Usefulness

<table>
<thead>
<tr>
<th>Category</th>
<th>Types of issues included</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>Comments, naming, style</td>
<td>22.3%</td>
</tr>
<tr>
<td>Organization of code</td>
<td>Modularity, location of artifacts, new class, duplicate code, size of methods</td>
<td>15.9%</td>
</tr>
<tr>
<td>Solution approach</td>
<td>Alternate algorithm or data structure</td>
<td>8.5%</td>
</tr>
<tr>
<td>Validation</td>
<td>Lack of or improper validation</td>
<td>7.2%</td>
</tr>
<tr>
<td>Visual representation</td>
<td>Beautification, indentation, blank linkes</td>
<td>6.4%</td>
</tr>
<tr>
<td>False positive</td>
<td>Not an real issue</td>
<td>4.6%</td>
</tr>
<tr>
<td>Defect</td>
<td>Incorrect implementation or missing functionality</td>
<td>2.6%</td>
</tr>
<tr>
<td>Logical</td>
<td>Control flow or logic issues</td>
<td>2.3%</td>
</tr>
<tr>
<td>Support</td>
<td>Configuration support systems or libraries</td>
<td>2.0%</td>
</tr>
<tr>
<td>Interface</td>
<td>Interactions with other components</td>
<td>1.5%</td>
</tr>
<tr>
<td>Resources</td>
<td>Resource initialization, manipulation, and release</td>
<td>1.3%</td>
</tr>
<tr>
<td>Timing</td>
<td>Thread synchronization, races</td>
<td>0.3%</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>25.1%</td>
</tr>
</tbody>
</table>


By: Amiangshu Bosu (U of Alabama), Michaela Greiler (TSE), Christian Bird (Microsoft Research Redmond), Characteristics of Useful Code Reviews: An Empirical Study at Microsoft (MSR 2015)
Smaller Reviews Are Better

Anecdotally: smaller reviews are “better”
From data: <=20 files implies usefulness stability and predictability

Absolute number of useful comments grows with size of review until 25-30 files, steady until 55-65 and then starts going down

By: Amiangshu Bosu (U of Alabama), Michaela Greiler (TSE), Christian Bird (Microsoft Research Redmond)
Relevant Experience Makes for Better Reviewers

Reviewers with prior experience with the changed file produce much more useful feedback.

New reviewers learn fast but need 6-12 months to be as productive as the rest of the team.

By: Amiangshu Bosu (U of Alabama), Michaela Greiler (TSE), Christian Bird (Microsoft Research Redmond)
Risk of Defects In a Change Can Be Predicted

Prior success with large-scale defect prediction
Expose risk prediction in code review to change the reviewer behavior
Improving the tools
Feature #1: Reviewer Recommendations

- Find potential reviewers based on their previous history with the code
- Consider number of changes and time since last activity
- Default is two reviewers based on most common practice and usefulness data

By: Christian Bird, Birendra Acharya, Michaela Greiler, Trevor Carnahan (Microsoft Research Redmond and TSE)
Feature #2: Change Decomposition

By: Shuvendu Lahiri, Mike Barnett, Christian Bird, Jack Tilford (Microsoft Research Redmond and TSE)
Feature #3: Change Risk Prediction

By: Nachi Nagappan, Jacek Czerwonka, Birendra Acharya (Microsoft Research Redmond and TSE)

By: Kim Herzig (Microsoft Research Cambridge)
Improving the workflow
Code Reviews in Engineering Workflow

**INNER LOOP**

- **Pre-checkin Validation**
  - Private test env
  - Functional tests
  - Static analysis

- **Coding**
  - Version control
  - Branch/Packages
  - IDE & Dev Tools

- **Local/Buddy Build**
  - Unit testing

- **Code Review**
  - Select reviewers
  - Review
  - Check-in

**OUTER LOOP**

- **Rolling/Official Build**
  - Build verification
  - Unit testing

- **Production Validation**
  - Watchdogs
  - Flighting

- **Deploy to Production**
  - Deployment targets

- **E2E Validation**
  - Pre-prod deploy
  - Scenario tests
  - Build drop

**Check-in**
### Study: Comments Classification

<table>
<thead>
<tr>
<th>Category</th>
<th>Types of issues included</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>Comments, naming, style</td>
<td>22.3%</td>
</tr>
<tr>
<td>Organization of code</td>
<td>Modularity, location of artifacts, new class, duplicate code, size of methods</td>
<td>15.9%</td>
</tr>
<tr>
<td>Solution approach</td>
<td>Alternate algorithm or data structure</td>
<td>8.5%</td>
</tr>
<tr>
<td>Validation</td>
<td>Lack of or improper validation</td>
<td>7.2%</td>
</tr>
<tr>
<td>Visual representation</td>
<td>Beautification, indentation, blank linkes</td>
<td>6.4%</td>
</tr>
<tr>
<td>False positive</td>
<td>Not an real issue</td>
<td>4.6%</td>
</tr>
<tr>
<td>Defect</td>
<td>Incorrect implementation or missing functionality</td>
<td>2.6%</td>
</tr>
<tr>
<td>Logical</td>
<td>Control flow or logic issues</td>
<td>2.3%</td>
</tr>
<tr>
<td>Support</td>
<td>Configuration support systems or libraries</td>
<td>2.0%</td>
</tr>
<tr>
<td>Interface</td>
<td>Interactions with other components</td>
<td>1.5%</td>
</tr>
<tr>
<td>Resources</td>
<td>Resource initialization, manipulation, and release</td>
<td>1.3%</td>
</tr>
<tr>
<td>Timing</td>
<td>Thread synchronization, races</td>
<td>0.3%</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>25.1%</td>
</tr>
</tbody>
</table>


Code Reviewing: It Takes Time and Effort

*Figure 5. Developers’ responses in surveys of the amount of code understanding for code review outcomes.*

*Figure 1: First Response on left (we do not have first response data for AMD) and Full interval on right*


The Importance of Speed

**Topic:** General

**Feedback:** Please implement a "virtual whip" feature to ping people who are dragging their feet on a review.

**ReviewID:** xyz-a723ea9f0dc14440586656f9

**Project:** test

**SourceControl:** 123456

**Effects of delaying:**

- Process stalls; impact on dependents
- Costly context switches for engineers
Code Reviewing is a Social Process

• Waiting is not just due to lack of time:
  I’m expected to participate but I’m not quite sure how. I’ll wait until someone else starts.
  There are a lot of outstanding comments already. I’ll wait until the next version.

• Code reviewing in social context
  Reviewing can be uncomfortable for authors and reviewers
  Team's hierarchy influence the outcome

• How many engineers does it take to do a proper code review?
  Sometimes reviewers are added as a courtesy
### Code Reviewing is Social

**Review completed**  
Submitted in changelist:3561857

**Open in:**  
[CodeFlow] [Dogfood] [Browser] [Visual Studio]

**Author:**

<table>
<thead>
<tr>
<th>Reviewer</th>
<th>Type</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton</td>
<td>Required</td>
<td>SignedOff</td>
</tr>
<tr>
<td>Vinod</td>
<td>Required</td>
<td>SignedOff</td>
</tr>
<tr>
<td>Erik</td>
<td>Required</td>
<td>SignedOff</td>
</tr>
<tr>
<td>Yancho</td>
<td>Required</td>
<td>Started</td>
</tr>
<tr>
<td>Hamed E.</td>
<td>Required</td>
<td>NotStarted</td>
</tr>
<tr>
<td>CodeFlow Listener</td>
<td>Optional</td>
<td>SignedOff</td>
</tr>
<tr>
<td>Q Build Team - Internal</td>
<td>Optional</td>
<td>InviteOnly</td>
</tr>
</tbody>
</table>

**Description:**  
Fix Bug 168324. Zero build outputs get Cached and persist through sessions causing build failures
Code Reviews Are Not Free

Potential Benefits
• More maintainable designs
• More consistent code base
• Knowledge sharing
• Better awareness of changes
• Additional defects found

Potential Costs
• Time spent by reviewers
• Time spent by author addressing feedback
• Time spent by change waiting in process

Do we need to apply the same verification criteria to all changes?
Opportunities for Optimization

Automate tasks
• Syntax, style conformance, static analysis

Make better use of people and resources
• When is it required to have a senior engineer?
• Since we mostly find maintainability issues, who is best at that?
• How to best prioritize work for reviewers?

Reorder steps
• Code review after submission not before?

Eliminate steps
• Auto-submit after required reviewers signed off?

Make reviewing optional
• Does every change need the same level of reviewing?
• Which types of changes do not benefit little from code reviewing?

Significant workflow changes require better understanding of how and when code reviews provide value
Summary

Code reviews do not find as many bugs as you may think (they are still very useful)
There is a discrepancy between what developers aim for and what the process does
You can’t use code reviews as a replacement for other verification techniques
Code reviewing is costly as often the longest and most variable part of code integration
Code reviewing is a skill that needs time to hone
Social aspects of code reviewing make it a complex process
Data can influence changes to tools but verify the outcome
Applying the same process to all commits is wasteful but to early to make changes
To improve further we need to understand the process even more
Mission: “Enable Microsoft to accelerate its software development”
TSE contributes to and innovates on major parts of the engineering system
TSE runs many of the engineering services and works with all engineering teams
TSE collaborates closely with Microsoft Research teams
Collaboration with academia: visiting researchers and PhD interns (10 this summer)

http://research.microsoft.com/tse