You ask me for a contribution
Well, you know
We’re all doing what we can
Mining Gerrit for Contentious Reviews & Community Evolution

Shane McIntosh
Assistant Professor

shane.mcintosh@mcgill.ca
@SoftwareREBELs
rebels.ece.mcgill.ca

McGill
Repository Excavation
Development data is mainly kept for archival purposes.
What if your software repositories could talk?
My students and I mine through historical development data in search of gold
My students and I mine through historical development data in search of gold.

In which modules are future defects likely to appear?

Baljinder Ghotra
Chakkrit (Kla) T.
My students and I mine through historical development data in search of gold.

In which modules are future defects likely to appear?

Baljinder Ghotra  Chakkrit (Kla) T.

Can risky code be predicted at creation time?

Takafumi Fukushima  Jacob G. Barnett
Software Rebel
Repository Excavation
Release Engineering: Supporting release pipelines
Release Engineering: Supporting release pipelines

1. Integration
Release Engineering: Supporting release pipelines

1. Integration
2. Build
Release Engineering: Supporting release pipelines

1. Integration

2. Build

3. Deployment

New!
Decision support for modern release pipelines
Release Engineering: Supporting release pipelines

1. Integration
2. Build
3. Deployment

New!
Release Engineering: Supporting release pipelines

1. Integration
2. Build
3. Deployment

Release Engineering:
Supporting release pipelines

Can we improve support for modern code review?

Toshiki Hirao  Ray Wen

New!
Release Engineering:
Supporting release pipelines

Can we improve support for modern code review?
Toshiki Hirao  Ray Wen

How can we optimize available CI/CD resources?
Keheliya Gallaba

How can we optimize available CI/CD resources?
We are looking to form industrial partnerships!
An overview of the modern code review process
An overview of the modern code review process
An overview of the modern code review process
An overview of the modern code review process

A code review tool (e.g., Gerrit)

Module A

Author
Reviewers may identify problems or suggest alternative approaches.
Reviewers may identify problems or suggest alternative approaches

A code review tool (e.g., Gerrit)

Qt Review #27977
Reviewers may identify problems or suggest alternative approaches

Author

Reviewer

A code review tool (e.g., Gerrit)

Qt Review #27977

Shouldn't console.log() call the toString() method (where appropriate) on objects?

Identifying a defect
Reviewers may identify problems or suggest alternative approaches

Shouldn’t `console.log()` call the `toString()` method (where appropriate) on objects?

Identifying a defect

A code review tool (e.g., Gerrit)

Qt Review #27977
Reviewers may identify problems or suggest alternative approaches

Reviewer

Shouldn't `console.log()` call the `toString()` method (where appropriate) on objects?

Reviewer

I think it’s better to do

```javascript
var s = "{}"
console.log(s)
```

Suggesting a solution

Author

A code review tool (e.g., Gerrit)

Qt Review #27977
Reviewers may identify problems or suggest alternative approaches.

Reviewer: Shouldn't `console.log()` call the `toString()` method (where appropriate) on objects?

Reviewer: I think it’s better to do `var s = "{}"` `console.log(s)`

Suggesting a solution

Author: Update

A code review tool (e.g., Gerrit)

Qt Review #27977

Reviewers may identify problems or suggest alternative approaches.
Some reviewers even provide potential solutions to the issues they identify.

Providing updates to the code change

Note that running `qutlook` will probably crash. I will push a patch set to fix the issue.

Author

Reviewer

A code review tool (e.g., Gerrit)

Qt Review #35360
Some reviewers even provide potential solutions to the issues they identify.

Note that running `qutlook` will probably crash. I will push a patch set to fix the issue.

Providing updates to the code change.

A code review tool (e.g., Gerrit)

Qt Review #35360
Modern code review:
Changes are guilty until proven innocent
Modern code review:
Changes are guilty until proven innocent

Legend
Pass →
Fail →

Version Control System

1. Upload change revision(s)
2. Execute sanity tests
3. Solicit peer feedback
4. Initiate integration request
5. Execute integration tests
6. Final integration
Code review takes a lot of developer time and is not always useful.
Code review takes a lot of developer time and is not always useful.

Devs spend 6 hours per week reviewing code.

Code review takes a lot of developer time and is not always useful

Devs spend 6 hours per week reviewing code

35% of review comments are “not useful”

[Bosu and Carver, ESEM 2013]

Characteristics of Useful Code Reviews: An Empirical Study at Microsoft
[Bosu et al., MSR 2015]
Developers do not like to spend time on process

Programming, Motherf
Do you speak it?

We are a community of motherf-programmers who have been humiliated by software development methodologies for years.

We are tired of XP, Scrum, Kanban, Waterfall, Software Craftsmanship (aka XP-Lite) and anything else getting in the way of...Programming, Motherf

We are tired of being told we're socially awkward idiots who need to be manipulated to work in a Forced Pair Programming chain gang without any time to be creative because none of the 10 managers on the project can do... Programming, Motherf

We must destroy these methodologies that get in the way of...Programming, Motherf
Mining Gerrit reviews to study reviewing processes
Mining Gerrit reviews to study reviewing processes

Contentious reviews

Toshiki Hirao
PhD Student
Mining Gerrit reviews to study reviewing processes

Contentious reviews

Toshiki Hirao
PhD Student

Community evolution

Ray Wen
Master’s Student
Mining Gerrit reviews to study reviewing processes

Contentious reviews

Toshiki Hirao
PhD Student

Community evolution

Ray Wen
Master's Student
And when the broken hearted people,
Living in the world agree,
There will be an answer,
Let it be
What happens when reviewers disagree?
Reviewer opinions about a patch may differ.
Reviewer opinions about a patch may differ.
Reviewer opinions about a patch may differ

Author

Reviewer 1

+2
Reviewer opinions about a patch may differ.
Reviewer opinions about a patch may differ

How do we proceed?

Author

Reviewer 1

+2

Reviewer 2

-2
Understanding code review in contentious patches
Understanding code review in contentious patches
Understanding code review in contentious patches

How often?
Understanding code review in contentious patches

How often?

Trend over time?
Understanding code review in contentious patches

How often?

Trend over time?

What drives abandonment?

Start
Understanding code review in contentious patches

How often?

Trend over time?

What drives abandonment?

How are concerns resolved?
Selecting subject systems for analysis or: why did we study OpenStack?

- **OpenStack**: 67% (Multi-reviewer patches)
- **Qt**: 21% (Multi-reviewer patches)
- **Android**: 26% (Multi-reviewer patches)
- **Eclipse**: 17% (Multi-reviewer patches)
- **LibreOffice**: 6% (Multi-reviewer patches)

Numbers represent the percentage of patches reviewed by multiple reviewers.
Selecting subject systems for analysis
or: why did we study OpenStack?

- openstack: 67%
- Qt: 21%
- Android: 26%
- Eclipse: 17%
- LibreOffice: 6%

Multi-reviewer patches
Solo-reviewer patches

0 22,500 45,000 67,500 90,000
Understanding code review in contentious patches

How often?

Trend over time?

What drives abandonment?

How are concerns resolved?
Understanding code review in contentious patches

How often?

Trend over time?

What drives abandonment?

How are concerns resolved?
Classifying contentious reviews

Include SP (+2)

- Yes
  - Include SN (-2)
    - Yes
      - SP-SN
        - +2
        - -2
        - +1
        - -1
    - No
      - Include SN (-2)
        - Yes
      - No
        - Include SN (-2)
          - Yes
          - No
          - Include SN (-2)

Handling reviews with several revisions

Contentious Pattern

Revision 1

+1

A

Revision 2

+2

A

Non-contentious Pattern

Revision 1

-2

A

Revision 2

+1

B

+1

C
Contentious reviews account for 26% of all OpenStack reviews.

Number (and percentage) of reviews in each category:

- +2, -2: 684 (1%)
- +2, -1: 7,753 (13%)
- +1, -2: 558 (1%)
- +1, -1: 6,378 (11%)
Contentious reviews with equal scores have a tendency to be integrated

Rate at which patches are eventually integrated

- +2, -2: 54%
- +2, -1: 88%
- +1, -2: 29%
- +1, -1: 76%
Understanding code review in contentious patches

How often?

Trend over time?

What drives abandonment?

How are concerns resolved?

Contentious reviews are not rare

Contention is not a show stopper
Understanding code review in contentious patches

How often?

Contentious reviews are not rare

Contention is not a show stopper

Trend over time?

What drives abandonment?

Start

How are concerns resolved?
Contentious reviews are growing over time in absolute terms.
Contentious reviews are also growing over time in comparative terms.
Integration rates of contentious reviews have roughly stabilized in recent periods.
Understanding code review in contentious patches

How often?
Contentious reviews are not rare
Contention is not a show stopper

Trend over time?
Contentious reviews have been growing over time
Integration rates of contentious reviews have stabilized

What drives abandonment?
Start

How are concerns resolved?
Handshake
Understanding code review in contentious patches

How often?

Contentious reviews are not rare
Contention is not a show stopper

Trend over time?

Contentious reviews have been growing over time
Integration rates of contentious reviews have stabilized

What drives abandonment?

Start

How are concerns resolved?

Handshake
Qualitative analysis: Selecting a representative sample

- 684 reviews with +2 and -2
- Sample of 362 reviews
Qualitative analysis:
Selecting a representative sample

684 reviews
with +2 and -2

Sample of 362 reviews
Qualitative analysis:
Manual classification of reviews

Sample of 362 reviews
Qualitative analysis: Manual classification of reviews

Sample of 362 reviews

Looks like a test coverage issue
Qualitative analysis: Manual classification of reviews

Sample of 362 reviews

Looks like a test coverage issue

Agreed!
Qualitative analysis: Manual classification of reviews

Sample of 362 reviews

Looks like a test coverage issue

Agreed!
Qualitative analysis:
Lifting tags to cohesive subhierarchies

Test coverage
Side effects
Alternative solution
Already fixed
Qualitative analysis: Lifting tags to cohesive subhierarchies

- QA
  - Test coverage
- Design
  - Side effects
  - Alternative solution
- Unnecessary contribution
  - Already fixed
Qualitative analysis: Lifting tags to cohesive subhierarchies

- Internal concerns
  - QA
    - Test coverage
  - Design
    - Side effects
    - Alternative solution
- External concerns
  - Unnecessary contribution
  - Already fixed
Abandonment is often due to external concerns!
Abandonment is often due to external concerns!

- **External Concerns** (59% (102/173))
  - Abandon
    - Integration Policy Compliance (173)
      - Integration Planning (25% (25/105))
        - Unnecessary Fix (58% (59/105))
          - Unclear Intention 10
            - Already Fixed 32
            - Not an Issue 17
          - Patch Dependency 5
            - Blueprint 6
            - Release Schedule 14
          - Lost by an author 6
            - Lost by a reviewer 1
          - Squashing Commits 6
            - Branch Placement 5
      - Lack of Interest (7% (7/105))
        - Alternative Solution 21
          - Flawed Changes 12
            - Shallow Fix 24
              - Side Effect 5
                - Patch Size 6
                  - Backward Compatibility 1
                    - Test Coverage 1
                      - Test Failure 1
        - Implementation (11% (11/105))
          - Design (87% (62/71))
            - Flawed Changes 12
              - Shallow Fix 24
                - Side Effect 5
                  - Patch Size 6
                    - Backward Compatibility 1
                      - Test Coverage 1
                        - Test Failure 1
          - Testing (3% (2/71))
            - Flawed Changes 12
              - Shallow Fix 24
                - Side Effect 5
                  - Patch Size 6
                    - Backward Compatibility 1
                      - Test Coverage 1
                        - Test Failure 1

- **Internal Concerns** (41% (71/173))
  - Abandon
    - Integration Policy Compliance (173)
Understanding code review in contentious patches

**How often?**

- Contentious reviews are not rare
- Contention is not a show stopper

**Trend over time?**

- Contentious reviews have been growing over time
- Integration rates of contentious reviews have stabilized

**What drives abandonment?**

- Concerns beyond patch scope are more strongly linked with abandonment than concerns within patch scope

**How are concerns resolved?**

- Start
Understanding code review in contentious patches

How often?
- Contentious reviews are not rare
- Contention is not a show stopper

Trend over time?
- Contentious reviews have been growing over time
- Integration rates of contentious reviews have stabilized

What drives abandonment?
- Concerns beyond patch scope are more strongly linked with abandonment than concerns within patch scope

How are concerns resolved?
- Start

Contention is more often resolved without altering the patch!

```
<table>
<thead>
<tr>
<th>%</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>44%</td>
<td>83/189</td>
</tr>
<tr>
<td>70%</td>
<td>58/83</td>
</tr>
<tr>
<td>12%</td>
<td>10/83</td>
</tr>
<tr>
<td>4%</td>
<td>3/83</td>
</tr>
<tr>
<td>5%</td>
<td>4/83</td>
</tr>
<tr>
<td>7%</td>
<td>6/83</td>
</tr>
<tr>
<td>2%</td>
<td>2/83</td>
</tr>
<tr>
<td>12%</td>
<td>10/83</td>
</tr>
<tr>
<td>38</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td></td>
</tr>
</tbody>
</table>
```

- Integrate
  - Directly Addressing
    - 44% (83/189)
      - Design: 70% (58/83)
        - Alternative Solution: 4
        - Flawed Changes: 38
      - Implementation: 4% (3/83)
        - Backward Compatibility: 3
      - Testing: 7% (6/83)
        - Test Coverage: 3
        - Test Failure: 3
      - Unnecessary Fix: 12% (10/83)
        - Unclear Intention: 10
      - Legal Problems: 2% (2/83)
        - Legal Issues: 2
      - Integration Policy Compliance: 5% (4/83)
        - Squashing Commits: 1
        - Branch Placement: 3
      - Withdrawal of Negative Score: 28% (30/106)
        - Self-change: 9
        - Persuasion: 21
        - Patch Dependency: 7
    - Indirectly Addressing
      - 56% (106/189)
      - Integration Planning: 72% (76/106)
        - Blueprint: 4
        - Release Schedule: 65
Contention is more often resolved without altering the patch!

- Directly Addressing
  - Design: 70% (58/83)
    - Alternative Solution: 4
    - Flawed Changes: 38
  - Implementation: 4% (3/83)
    - Backward Compatibility: 3
  - Testing: 7% (6/83)
    - Test Coverage: 3
    - Test Failure: 3
  - Unnecessary Fix: 12% (10/83)
    - Unclear Intention: 10
  - Legal Problems: 7% (6/83)
    - Legal Issues: 2
  - Integration Policy Compliance: 5% (4/83)
    - Squashing Commits: 1
    - Branch Placement: 3

- Indirectly Addressing
  - Withdrawal of Negative Score: 28% (30/106)
    - Self-change: 9
    - Persuasion: 21
  - Integration Planning: 72% (76/106)
    - Patch Dependency: 7
    - Blueprint: 4
    - Release Schedule: 65
Understanding code review in contentious patches

**How often?**
- Contentious reviews are not rare
- Contention is not a show stopper

**Trend over time?**
- Contentious reviews have been growing over time
- Integration rates of contentious reviews have stabilized

**What drives abandonment?**
- Concerns beyond patch scope are more strongly linked with abandonment than concerns within patch scope

**How are concerns resolved?**
- Concerns are often addressed through negotiation or scheduling
Life is very short, and there’s no time, for fussing and fighting, my friends.
Mining Gerrit reviews to study reviewing processes

Contentious reviews

Toshiki Hirao
PhD Student

Community evolution

Ray Wen
Master’s Student
Mining Gerrit reviews to study reviewing processes

Contentious reviews

Toshiki Hirao
PhD Student

Community evolution

Ray Wen
Master’s Student
And in the end
The love you take
Is equal to
The love you make
And in the end
The love you take
Is equal to
The love you make

You get what you put in(to code review)
You get what you put into code review
You get what you put into code review
You get what you put into code review

Looks like code to me! +2
You get what you put into code review

Looks like code to me!

Did you consider alternative designs?

Rookie Reviewer

Seasoned Reviewer

Author

+2

-1
How does reviewing feedback evolve with respect to...
How does reviewing feedback evolve with respect to...

Community aging?
How does reviewing feedback evolve with respect to...

- Community aging?
- Reviewer experience?
How does reviewing feedback evolve with respect to...

- Community aging?
- Reviewer experience?
- Reviewer workload?
How does reviewing feedback evolve with respect to...

- Community aging?
- Reviewer experience?
- Reviewer workload?
How does reviewing feedback evolve with respect to...

Community aging?

Reviewer experience?

Reviewer workload?
Automatically analyzing topics that are discussed in code reviews
Automatically analyzing topics that are discussed in code reviews

Get inline comments

506,950 inline comments
Automatically analyzing topics that are discussed in code reviews

Get inline comments

Apply topic analysis

506,950 inline comments

Topic 1

Topic 2

Topic 3
Automatically analyzing topics that are discussed in code reviews

1. Get inline comments
2. Apply topic analysis
3. Topic score analysis

506,950 inline comments
LDA automatically groups words into topics based on co-occurrence in comments.

### Table 1: The labelled topics with their corresponding words and topic share score

<table>
<thead>
<tr>
<th>Selected Words</th>
<th>Most Relevant Words</th>
<th>Topic Share (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>log, debug, warn, little, anywhere, bit, level, dashboard</code></td>
<td><code>comment, version, package, install, repo, option, config, copyright</code></td>
<td>4.8</td>
</tr>
<tr>
<td><code>big, core, timeout, performance, time, team, choice</code></td>
<td></td>
<td>4.4</td>
</tr>
<tr>
<td><code>test, miss, unit, sentence, character, functional, period</code></td>
<td><code>need, line, blank, pep, ref, alarm, insert, codeblock</code></td>
<td>5.5</td>
</tr>
<tr>
<td><code>ok, release, fix, yeah, liberty, thank, rid, mitaka, ah</code></td>
<td><code>ditto, token, context, label, auth, align, hint, keystone</code></td>
<td>4.1</td>
</tr>
<tr>
<td><code>function, period</code></td>
<td><code>need, line, blank, pep, ref, alarm, insert, codeblock</code></td>
<td>4.1</td>
</tr>
<tr>
<td><code>ok, release, fix, yeah, liberty, thank, rid, mitaka, ah</code></td>
<td><code>ditto, token, context, label, auth, align, hint, keystone</code></td>
<td>6.8</td>
</tr>
<tr>
<td><code>function, period</code></td>
<td><code>need, line, blank, pep, ref, alarm, insert, codeblock</code></td>
<td>2.1</td>
</tr>
</tbody>
</table>

3.1 Approach

We first identify the high-level concepts that the LDA-generated topics highlight. More specifically, we label them by reading the 20 terms and 20 review comments with the strongest association to each topic. We select the terms with the top 20 term weights for each topic. When ordering the terms that a topic is comprised of, we draw inspiration from the Term Frequency - Inverse Document Frequency (TF-IDF) concept. The TF score is mapped to the term weight within the topic. The IDF score is mapped to the Inverse Topic Frequency (ITF). We order terms by their TF-ITF score—terms with high term weight scores that appear in few other topics are considered first in our topic labelling process.
We read comments with high topic scores and name the topics

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>Selected Most Relevant Words</th>
<th>Topic Share (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logging</td>
<td>log, debug, warn, little, anywhere, bit, level, dashboard</td>
<td>4.8</td>
</tr>
<tr>
<td>Release Engineering</td>
<td>comment, version, package, install, repo, option, config, copyright</td>
<td>4.4</td>
</tr>
<tr>
<td>Performance</td>
<td>big, core, timeout, performance, time, team, choice</td>
<td>5.5</td>
</tr>
<tr>
<td>Testing</td>
<td>test, miss, unit, sentence, character, functional, period need, line, blank, pep, ref, alarm, insert, codeblock</td>
<td>4.1</td>
</tr>
<tr>
<td>Improvement Request</td>
<td></td>
<td>4.1</td>
</tr>
<tr>
<td>General Social Comm.</td>
<td>ok, release, fix, yeah, liberty, thank, rid, mitaka, ah</td>
<td>6.8</td>
</tr>
<tr>
<td>Confirmation</td>
<td>ditto, token, context, label, auth, align, hint, keystone</td>
<td>2.1</td>
</tr>
</tbody>
</table>

How to enable TRACE log level in config file? Enabling the trace would enable debug also?

Comment #136066, topic_score(logging) = 0.94
Then we group related topics into categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Topic Name</th>
<th>Selected Most Relevant Words</th>
<th>Topic Share (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>Logging</td>
<td>log, debug, warn, little, anywhere, bit, level, dashboard</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>Release Engineering</td>
<td>comment, version, package, install, repo, option, config, copyright</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>Performance</td>
<td>big, core, timeout, performance, time, team, choice</td>
<td>5.5</td>
</tr>
<tr>
<td>Testing</td>
<td>Testing</td>
<td>test, miss, unit, sentence, character, functional, period</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>Improvement Request</td>
<td>need, line, blank, pep, ref, alarm, insert, codeblock</td>
<td>4.1</td>
</tr>
<tr>
<td>Social Communication</td>
<td>General Social Comm.</td>
<td>ok, release, fix, yeah, liberty, thank, rid, mitaka, ah</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>Confirmation</td>
<td>ditto, token, context, label, auth, align, hint, keystone</td>
<td>2.1</td>
</tr>
</tbody>
</table>
As the community has aged, design-oriented feedback has declined

Software Design

Key Errors  ↓

Exception Handling  ↓

Object Names  —

Function Headers  ↓

Default and Parameters  —
On the other hand, discussion of networking and patch formatting have grown.

- Server-related Discussion
- Network Issues ↑
- Formatting ↑
How does reviewing feedback evolve with respect to...

Community aging?

Reviewer experience?

Reviewer workload?

As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting.
How does reviewing feedback evolve with respect to...

- Community aging?
- Reviewer experience?
- Reviewer workload?

As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting.
System-oriented topics increase as reviewers accrue experience with OpenStack

- **Operations**
  - Logging
  - Performance

- **Server-related Discussion**
  - Node Issues
  - Object Names

- **Software Design**
  - Function Headers

- **Social Communication**
  - General Social Comm.

*Other words, the topic impact metric measures the proportion of review comments.*
System-oriented topics increase as reviewers accrue experience with OpenStack.

- Operations
  - Logging
  - Performance

- Server-related Discussion
  - Node Issues

- Software Design
  - Object Names
  - Function Headers

- Social Communication
  - General Comm.
  - Social Comm.
Low ROI topics like Formatting tend to decrease as reviewers accrue experience.
How does reviewing feedback evolve with respect to...

**Community aging?**

As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting.

**Reviewer experience?**

As OpenStack reviewers accrue experience, they tend to comment more on system-oriented topics and less on low ROI topics.

**Reviewer workload?**
How does reviewing feedback evolve with respect to...

Community aging?

As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting.

Reviewer experience?

As OpenStack reviewers accrue experience, they tend to comment more on system-oriented topics and less on low ROI topics.

Reviewer workload?
Reviewers with high experience scores tend to also have a heavy workload score

= 0.68
How does reviewing feedback evolve with respect to...

**Community aging?**
As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting.

**Reviewer experience?**
As OpenStack reviewers accrue experience, they tend to comment more on system-oriented topics and less on low ROI topics.

**Reviewer workload?**
Similar observations as were reported for experience.
I wanna hold your hand!
Mining Gerrit reviews to study reviewing processes

Contentious reviews
Toshiki Hirao
PhD Student

Community evolution
Ray Wen
Master’s Student
Mining Gerrit reviews to study reviewing processes

Contentious reviews

Toshiki Hirao
PhD Student

Community evolution

Ray Wen
Master’s Student
Code review takes a lot of developer time and is not always useful

- Devs spend 6 hours per week reviewing code
- 35% of review comments are “not useful”

[Bosu and Carver, ESEM 2013]

Characteristics of Useful Code Reviews: An Empirical Study at Microsoft
[Bosu et al., MSR 2015]
Understanding code review in contentious patches

**How often?**
- Contentious reviews are not rare
- Contention is not a showstopper

**Trend over time?**
- Contentious reviews have been growing over time
- Integration rates of contentious reviews have stabilized

**What drives abandonment?**
- Concerns beyond patch scope are more strongly linked with abandonment than concerns within patch scope

**How are concerns resolved?**
- Concerns are often addressed through negotiation or scheduling
How does reviewing feedback evolve with respect to...

**Community aging?**
As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting.

**Reviewer experience?**
As OpenStack reviewers accrue experience, they tend to comment more on system-oriented topics and less on low ROI topics.

**Reviewer workload?**
Similar observations as were reported for experience.
Mining Gerrit for Contentious Reviews & Community Evolution

Shane McIntosh
Assistant Professor

shane.mcintosh@mcgill.ca
@SoftwareREBELs
rebels.ece.mcgill.ca

McGill
Code review takes a lot of developer time and is not always useful

- Devs spend 6 hours per week reviewing code
- 35% of review comments are “not useful”

[Bosu and Carver, ESEM 2013]

Characteristics of Useful Code Reviews: An Empirical Study at Microsoft
[Bosu et al., MSR 2015]

Understanding code review in contentious patches

- How often?
- Trend over time?
- What drives abandonment?
- How are concerns resolved?

- Contentious reviews are not rare
- Integration rates of contentious reviews have stabilized
- Concerns beyond patch scope are more strongly linked with abandonment than concerns within patch scope
- Concerns are often addressed through negotiation or scheduling

How does reviewing feedback evolve with respect to...

- Community aging?
- Reviewer experience?
- Reviewer workload?

As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting

As OpenStack reviewers accrue experience, they tend to comment more on system-oriented topics and less on low ROI topics

Similar observations as were reported for experience

Mining Gerrit for Contentious Reviews & Community Evolution

Shane McIntosh
Assistant Professor

shane.mcintosh@mcgill.ca
@SoftwareREBELs
rebels.ece.mcgill.ca
Code review takes a lot of developer time and is not always useful

Devs spend 6 hours per week reviewing code
35% of review comments are “not useful”.

[Bosu and Carver, ESEM 2013]

Characteristics of Useful Code Reviews: An Empirical Study at Microsoft
[Bosu et al., MSR 2013]

Understanding code review in contentious patches

How often?

Contentious reviews are not rare
Contention is not a show stopper

Trend over time?

Contentious reviews have been growing over time
Integration rates of contentious reviews have stabilized

What drives abandonment?

Concerns beyond patch scope are more strongly linked with abandonment than concerns within patch scope

How are concerns resolved?

Concerns are often addressed through negotiation or scheduling

How does reviewing feedback evolve with respect to...

Community aging?

As OpenStack has aged, the community has focused less on patch design and more on networking and patch formatting

Reviewer experience?

As OpenStack reviewers accrue experience, they tend to comment more on system-oriented topics and less on low ROI topics

Reviewer workload?

Similar observations as were reported for experience

Mining Gerrit for Contentious Reviews & Community Evolution

Shane McIntosh
Assistant Professor

@shane.mcintosh@mcmill.ca
@SoftwareREBELs
rebels.ece.mcgill.ca