Robot Comments

Edwin Kempin, Google
What are Robot Comments?

Robot Comments = Comments generated by automated systems.
What are Robot Comments?

E.g. created by static analysis tools that comment on
● Bugs
● Code Readability
● Code Style
What are Robot Comments?

E.g. created by static analysis tools that comment on
- Bugs
- Code Readability
- Code Style

⇒ Similar to Human Comments?
⇒ Why bother to add extra support for Robot Comments?
How are Robot Comments different?

Distinction to Human Comments:

- Visualization
- Filtering
- Suggested Fixes
- Retention
Users want to easily distinguish Human Comments and Robot Comments.

⇒ Different color for comment boxes.

⇒ Different decorations (buttons, links etc.)
Filtering

Users want to filter Robot Comments.

⇒ Show only Human Comments

⇒ Show only part of the Robot Comments
   (comments from some analyzers may be less important)
Suggested Fixes

For many problems fixes can be generated automatically.

⇒ Attach suggested fixes to Robot Comments

⇒ Allow users to apply them by a single click
Retention

Automated processes tend to generate large amounts of uninteresting data.

⇒ Might impact Gerrit performance

⇒ Robot Comments should be deletable
REST API - RobotCommentInfo

- Extends CommentInfo
  (fields: id, path, side, line, range, in_reply_to, updated, message, tag)

- Additional fields:
  - robot_id
  - robot_run_id
  - url (optional)
  - properties (optional)
  - fix_suggestions (optional)
Robot Comments REST Endpoints

- Post Robot Comments
- Retrieve Robot Comments
- Apply Fixes
REST API - Post Robot Comments

Use existing REST endpoint for posting a review:

POST /changes/{change-id}/revisions/{revision-id}/review
**REST API - Post Robot Comments**

ReviewInput has new field for Robot Comments:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The message to be added as review comment.</td>
</tr>
<tr>
<td>tag</td>
<td>Apply this tag to the review comment message, votes, and inline comments. Tags may be used by CI or other automated systems to distinguish them from human reviews. Comments with specific tag values can be filtered out in the web UI.</td>
</tr>
<tr>
<td>labels</td>
<td>The votes that should be added to the revision as a map that maps the label names to the voting values.</td>
</tr>
<tr>
<td>comments</td>
<td>The comments that should be added as a map that maps a file path to a list of CommentInput entities.</td>
</tr>
<tr>
<td>robot_comments</td>
<td>The robot comments that should be added as a map that maps a file path to a list of RobotCommentInput entities.</td>
</tr>
</tbody>
</table>
REST API - Retrieve Robot Comments

New REST endpoints analogous to REST endpoints for inline comments:

- List Robot Comments
  
  GET /changes/{change-id}/revisions/{revision-id}/robotcomments/

- Get Robot Comment
  
  GET /changes/{change-id}/revisions/{revision-id}/robotcomments/{comment-id}
REST API - Apply Fix

New REST endpoint:
PUT /changes/{change-id}/revisions/{revision-id}/fix/{fix-id}/apply

⇒ Creates a Change Edit
⇒ Multiple fixes can be applied to the same Change Edit.
⇒ Manual editing possible.
⇒ Change Edit is published as a new Patch Set
Storage Format (NoteDb)

- Robot Comments are stored in **NoteDb only**!
- Stored as notes on the patch set revision
  - Notes Branch:
    - `refs/changes/YY/XXXX/robot-comments`
  - Note Contents:
    - List of Robot Comments as JSON objects
- Similar to how Inline Comments are stored as Git Notes in
  `refs/changes/YY/XXXX/meta`
Storage Format (NoteDb)

Why `refs/changes/YY/XXXX/robot-comments`?

Why not store Robot Comments together with the Inline Comments in `refs/changes/YY/XXXXX/meta`?
Storage Format (NoteDb)

Why `refs/changes/YY/XXXX/robot-comments`?

Why not store Robot Comments together with the Inline Comments in `refs/changes/YY/XXXX/meta`?

⇒ Allows deletion of Robot Comments without rewriting the Change Meta Branch which serves as Audit Log.
Robot Comments UI (PolyGerrit)

Robot Comments are (fully) supported in **PolyGerrit only**

- Display of Robot Comments
- Filtering
- Viewing of Suggested Fixes
- Applying of Suggested Fix
- UI Extension Point
Display of Robot Comments

**clang-tidy**
clang-diagnostics-thread-safety-reference

Passing variable ‘mem_ceiling’ by reference requires holding mutex ‘lock_’
http://clang.llvm.org/extra/clang-tidy/checks/list.html

Please fix
Apply Suggested Fix

```
clang-tidy
clang-diagnostics-thread-safety-reference

Passing variable 'mem_ceiling' by reference requires holding mutex 'lock_
http://clang.llvm.org/extra/clang-tidy/checks/list.html
```

Please fix Show suggested fix (2)
View Suggested Fix
UI Extension Point

Allows adding additional functionality for Robot Comments, e.g.:

- Feedback Collection

---

Passing variable ‘mem_ceiling’ by reference requires holding mutex ‘lock’
http://clang.llvm.org/extra/clang-tidy/checks/list.html

Please fix

Not useful
How to integrate an analyzer?

1. Listen to Stream Events or Poll for new Changes
2. Run analyzer
3. Post Review: Vote + Post Robot Comments
Which analyzers will be integrated?

- **Tricium**: Analysis pipeline for Chromium  
  (based on Google’s internal static analysis pipeline, called Tricorder)

- **Shipshape (?)**: Open-source version of Tricorder  
  (built from scratch with a similar pipeline architecture and based on docker)
When is it available?

Planned timelines:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backend Support</td>
<td>Q1 2017</td>
</tr>
<tr>
<td>UI Support in PolyGerrit</td>
<td>Q1 2017</td>
</tr>
<tr>
<td>NoteDb Availability</td>
<td>Q1 2017</td>
</tr>
<tr>
<td>Tricium Integration</td>
<td>Q1/Q2 2017</td>
</tr>
<tr>
<td>Shipshape Integration</td>
<td>?</td>
</tr>
</tbody>
</table>

⇒ Available in Q2 2017
Thank You!

Questions?
Resources

- Design document by Emma Soederberg, Google: https://docs.google.com/document/d/1pLunr0YUvbFVjdgaiEM0IX-Cm0EbpLxfDP1QryGRXM/edit?usp=sharing