

Gerrit Code Review – Improvements to submit rules

Han-Wen Nienhuys
Google Munich
hanwen@google.com

Thanks to Maxime Guerreiro



Maxime Guerreiro
maxime.guerreiro@gmail.com

Origins of Gerrit at Google

- Started as standalone server for Android project
 - <https://review.source.android.com>
- Standalone server for Linux kernel
- Gerrit
- etc.

Not scalable



virtual hosting since early 2012 (v2.2)

Gerrit at Google today

- Standard Git hosting/codereview for Google projects
- Largely automated:

```
gerrit-control --ticket=12345 --name sapphire-skunk \  
  --owner sapphire-skunk-team@google.com \  
  --require-trusted-device=false \  
  --enable-gerrit=true
```

Result:

<https://sapphire-skunk-review.google.com/>

Congratulations!



Congratulations?

How to fasten seat belts

Warning



For an effective restraining action, the seat belt must be fastened correctly with the seat backrest in the upright position.

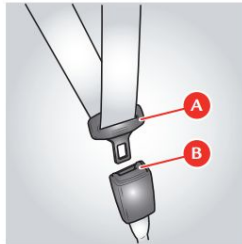
The seat belt is fastened correctly when the upper part of the belt crosses the centre of the shoulder (not the neck) and the abdominal section is fitted over the hips (not the abdomen).

Make sure it is not twisted and that it passes closely over your body; if not, in the event of a head-on collision, it may move and cause injury to the abdomen.

Avoid wearing bulky clothing that may interfere with the proper operation of the seat belts.

Once you have adjusted the seat correctly (see page 162):

- Grip the latch plate **A**, slowly pull the belt and insert the latch plate into the buckle **B** (if the belt locks while you are pulling it



out, let it wind back a little and pull it out again without jerking it).

- Make sure that it has clicked into the locked position: hold the belt and pull it to check that the latch plate has been inserted correctly.
- Position the seat belt correctly.

If the driver's seat belt is not fastened, when you turn the ignition key to position **II**, the warning light **D** on the instrument panel lights up and remains lit until the seat belt is fastened.

55 seconds after a speed of 10 km/h (6 mph) is exceeded, a buzzer sounds warning the driver that the seat belt is not fastened.

When a speed of 20 km/h (12 mph) is exceeded, the buzzer activates immediately and stops after 90 seconds.

This acoustic signal is emitted only once, even if the vehicle speed goes above and below the above mentioned limits. It is repeated (when the vehicle speed is in the indicated ranges) only if the seat belt is fastened and unfastened again or, in any case, every time the engine is turned off and then on.



Administering Gerrit

- Accounts/Groups/Authentication
- Permissions
- ACLs, ACL inheritance
- `workInProgressByDefault`, `matchAuthorToCommitterDate`, `rejectImplicitMerges`,
`createNewChangeForAllNotInTarget`, `headingMaxObjectSizeLimit`, `enableSignedPush`, `requireChangeID`, `useSignedOffBy`
- Submit strategies: cherry-pick, rebase-always, ... ?
- Workflows



Must **simplify** to decrease support load

Objective of Code Review

- Facilitate human review
- Enforce policy

Example: When can a Gerrit change be submitted?

- team “maintainers” voted “Review: +2”
- CI voted “Verified: +1”
- If changing file lib/*: team “gerrit googlers” voted “Library compliance: +1”
- All comments must be resolved

Submittability

Given

- project configuration (“rules”),
- change C with certain properties (“facts”),

can C be submitted?

Answer:

- YES
- NO, because X, Y, Z

This is **logic programming**, so use **Prolog**

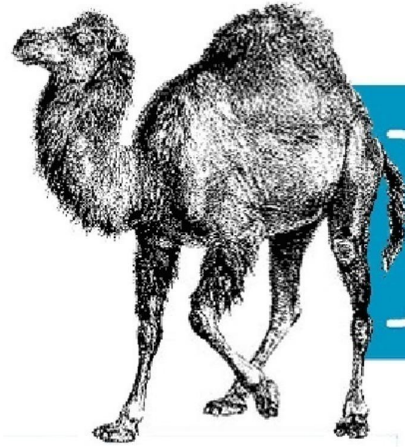
Prolog example

sample.pl:

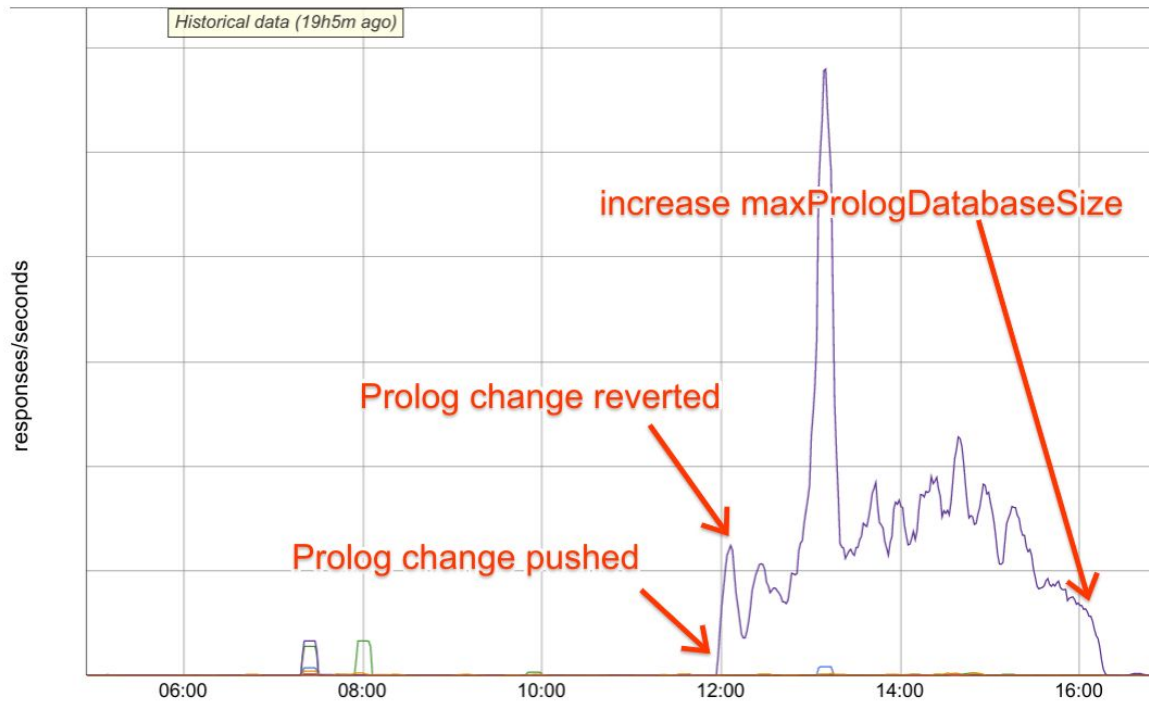
```
submit_rule(S) :-  
    gerrit:default_submit(X),  
    X =.. [submit | Ls],  
    add_non_author_approval(Ls, R),  
    S =.. [submit | R].
```

```
add_non_author_approval(S1, S2) :-  
    gerrit:commit_author(A),  
    gerrit:commit_label(label('Code-Review', 2), R),  
    R \= A, !,  
    S2 = [label('Non-Author-Code-Review', ok(R)) | S1].
```

```
add_non_author_approval(S1, [label('Non-Author-Code-Review', need(_)) | S1]).
```



Hackathon panic (Nov 14, 2018)



Prolog

The good

- Allows write complex logic
- Hermetic evaluation

The bad

- Nobody knows prolog
- Any requirement appears as label

The ugly

- Admins: cut & paste from cookbook
- My team: learn, debug & forget

Conclusion: drop Prolog

Verified

Label Status

Needs label:

- **Bug** ID must be specified for changes in this branch, b/35267372, b/69120821

Links

[builds](#)

The screenshot shows a Stack Overflow page with the following elements:

- Header:** Stack Overflow logo and a search bar.
- Navigation:** Home, PUBLIC, Stack Overflow (selected), Tags, Users, Jobs.
- Teams:** A section for "Teams" with a "Learn More" button.
- Question Title:** "Exclude author from gerrit review".
- Question Body:** "I want to disallow the author of a change to review his/her own changes in `gerrit`. I'm aware of [this suggested hack](#), but that doesn't really solve the issue. Now I [learned from the gerrit issues](#) that gerrit's hardcoded rules can be modified by [custom prolog code](#), so it should potentially be possible to modify the workflow as I want. However, I have never modified gerrit's workflow before and I don't know much `prolog`." (Note: the original image has some text obscured by a blue box).
- Metadata:** 15 votes, 4 answers, "prolog" and "gerrit" tags.
- Footer:** "share improve this question", "edited May 23 '17 at 12:00", "Community", and a user profile for "Bananeweizen" (asked Jul 19 '12 at 12:22, 18.5k reputation, 7 answers, 55 comments, 79 votes).

Incrementally defenestrating Prolog

Add an extension point. The SubmitRule contract:

```
interface SubmitRule {  
    public Collection<SubmitRecord> evaluate(  
        ChangeData cd, SubmitRuleOptions options);  
}  
  
...  
  
class PrologRule implements SubmitRule {  
    ...  
}
```



Submit API elements

SubmitRecord contains:

- A status (OK, NOT_READY, RULE_ERROR)
- A list of labels and their status (OK, REJECT, NEED, MAY)
- A list of **Requirements**
 - Human-readable description (ie: *Travis detected failing tests*)
 - Type (ie: *travis_status*)
 - Additional data (ie: *duration = 60s, tests_failed = 17*)

UI to display requirements

Submit Status Needs labels:


- Verified
- Code-Review
- Code-Style

Other Requirements:

- Resolve all comment threads

Submit Status

- Label: **Verified**
- Label: **Code-Review**
- Label: **Code-Style**
- Resolve all comments

 **Resolve all comments**

Verified +1  Maxime Guerreiro

Code-Review +2  Wyatt Allen

Sample code - Reject self approval

```

1 public class NoSelfApprovalRule implements SubmitRule {
2
3     private static final SubmitRequirement REQUIREMENT =
4         SubmitRequirement.builder()
5             .setType("no_self_approval")
6             .setFallbackText("You can't self approve a change")
7             .build();
8
9     @Override
10    public Collection<SubmitRecord> evaluate(ChangeData cd, SubmitRuleOptions options) {
11        boolean hasNonOwnerApproval = false;
12
13        try {
14            Account.Id owner = cd.change().getOwner();
15            for (PatchSetApproval approval : cd.currentApprovals()) {
16                if (!"Verified".equalsIgnoreCase(approval.getLabel())) continue;
17                if (approval.getValue() != +1) continue;
18
19                if (!approval.getAccountId().equals(owner)) {
20                    hasNonOwnerApproval = true;
21                }
22            }
23        } catch (OrmException e) {
24            return ImmutableList.of();
25        }
26
27        SubmitRecord sr = new SubmitRecord();
28        sr.requirements = Collections.singletonList(REQUIREMENT);
29        sr.status = hasNonOwnerApproval ? SubmitRecord.Status.NOT_READY : SubmitRecord.Status.OK;
30
31        return ImmutableList.of(sr);
32    }
}

```


Simple-submit, a plugin for basic rules

What are basic rules?

- Require resolved comments
- Configure existing labels:
 - Disable self-approval
 - Vote copying rules

Implemented in an open-source plugin

Simple Submit Rules

Allow submission with
unresolved comments

No ▾

Label Verified

Should a vote with the
maximum value be required?

Should votes with the lowest
value block submission?

(Expert users) Function name

Allow approval by the change
owner

When a new patchset is
uploaded, Gerrit should copy
votes ...

with minimal value ⓘ

with maximal value ⓘ

on trivial rebase ⓘ

when only the commit message
is modified ⓘ

when only commit metadata are
modified ⓘ

on rebased merge commits ⓘ

Submit rules as plugins

- It works!
- Don't have to debug other people's Prolog code
- Written in Java:
 - Not hermetic
 - Not flexible
- Can't link other everyone's Java rules into our Gerrit



Conclusion

- Simplicity is key to scaling support load
- New submit rule infrastructure:
 - Abstracted away Prolog interpreter
 - Use it to build simple submit plugin
 - Thanks to Maxime Guerreiro

<https://gerrit.googlesource.com/plugins/simple-submit-rules>

- Future work:
 - Move Prolog interpreter into plugin
 - Experiment with scripting engines (JS? Starlark?)

JS engine

What if we could write submit rules using Javascript?

The Requirement class takes two arguments:

- Is it met? (boolean)
- A description (string)

Plugin: gerrit.googlesource.com/plugins/scripting-rules

Design doc: go/gerrit-rules-js

JS engine

WIP

```
1 function submit_rule(change, requirements) {  
2   let verifiedVotes = change.findVotes("Verified", +1);  
3   if (verifiedVotes.length == 0) {  
4     return;  
5   }  
6  
7   for (var vote of verifiedVotes) {  
8     if (!vote.account.hasEmail(change.committer().email)) {  
9       return;  
10    }  
11  }  
12  requirements.push(new Requirement(false, "You can't self-verify."));  
13 }
```