Code Reviews

Rabea Gransberger

@rgransberger

rgra.github.io

About Me

Rabea Gransberger

- Computer Science Diploma 2008
- Developer, Project Lead at MEKO-S GmbH, Bremen
  - Code Review supported by tools in all projects
- Co-Organizer JUG Bremen

Agenda

- Why do Code Reviews?
- How to do Reviews?
- Which Tools are available?
- Tips for Developers and Reviewers
- Which social problems can occur?

INTRODUCTION

Why Code Reviews?

- Find errors
- Increase customer satisfaction
- Pareto principle (80/20 rule)
- Quality of code
- Education for whole team
- Less stress
- Lower overall project cost

Project cost

	Bugs	Cost
After Development	(463)	
QA/Test	142	200$ / fix
Customer (within 6 month)	127	1000$ / fix
Cost of fixing bugs		155k $
+ Cost of 194 latent bugs		194k $
Total		**349k $**

[10]

Project cost

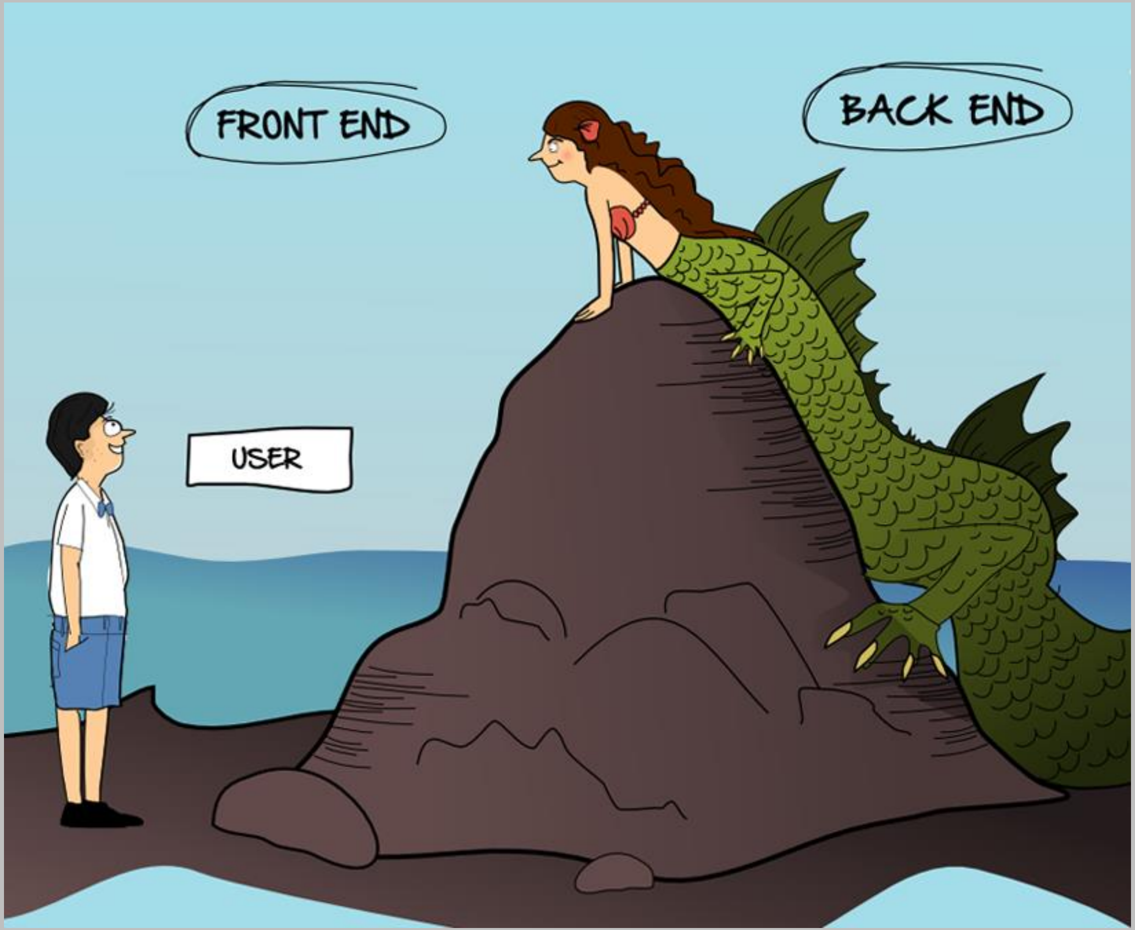
	Bugs	Cost
After Development	(463)	
Code Review	283	25$ / fix
After QA/Test	67	200$ / fix
After Customer	81	1000$ / fix
Cost of fixing bugs		101k $
+ Cost of 32 latent bugs		32k $
Total		**133k $** (349 k $)

We already do TDD…

- Readable code?

- Errors are not only found in code:
  - Requirements
  - Design
  - Documentation
  - Test cases

[3]



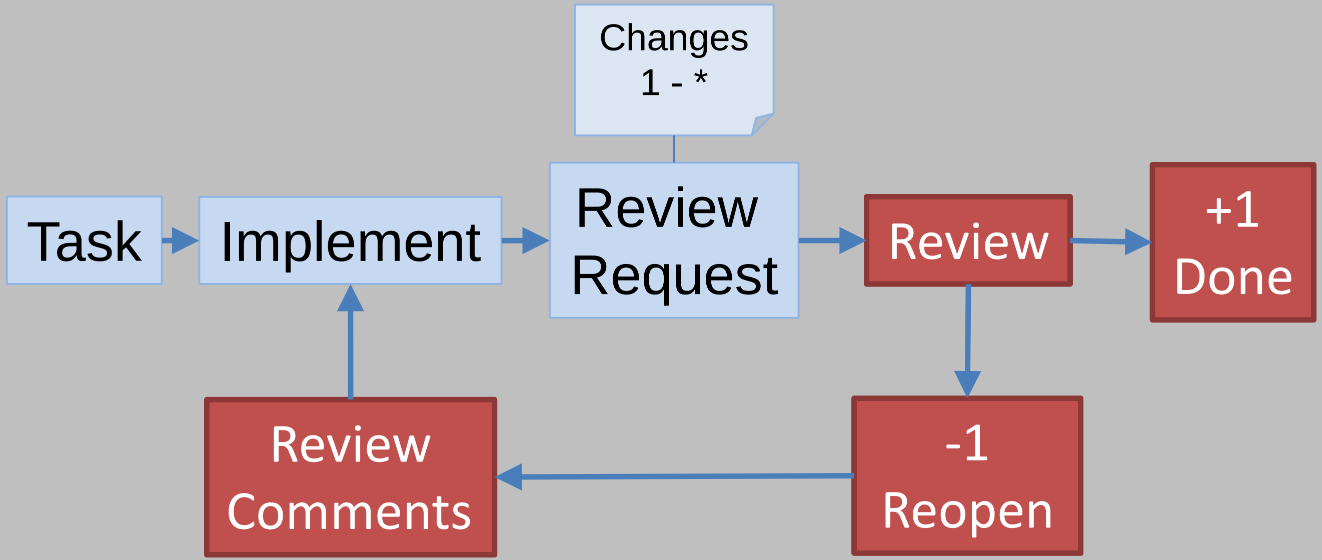
https://lol.browserling.com/full-stack-hires.png

PROCESS & TECHNIQUES

Process Types

- Formal
  - Inspection: formal meeting with whole team
  - Audit: by external company
- Informal / Lightweight
  - Pair Programming: 2 developers, 1 keyboard
  - Walkthrough: Author shows code to Reviewer
  - Tool-supported Review
- 20 % time, same number issues

Example: Task based review process



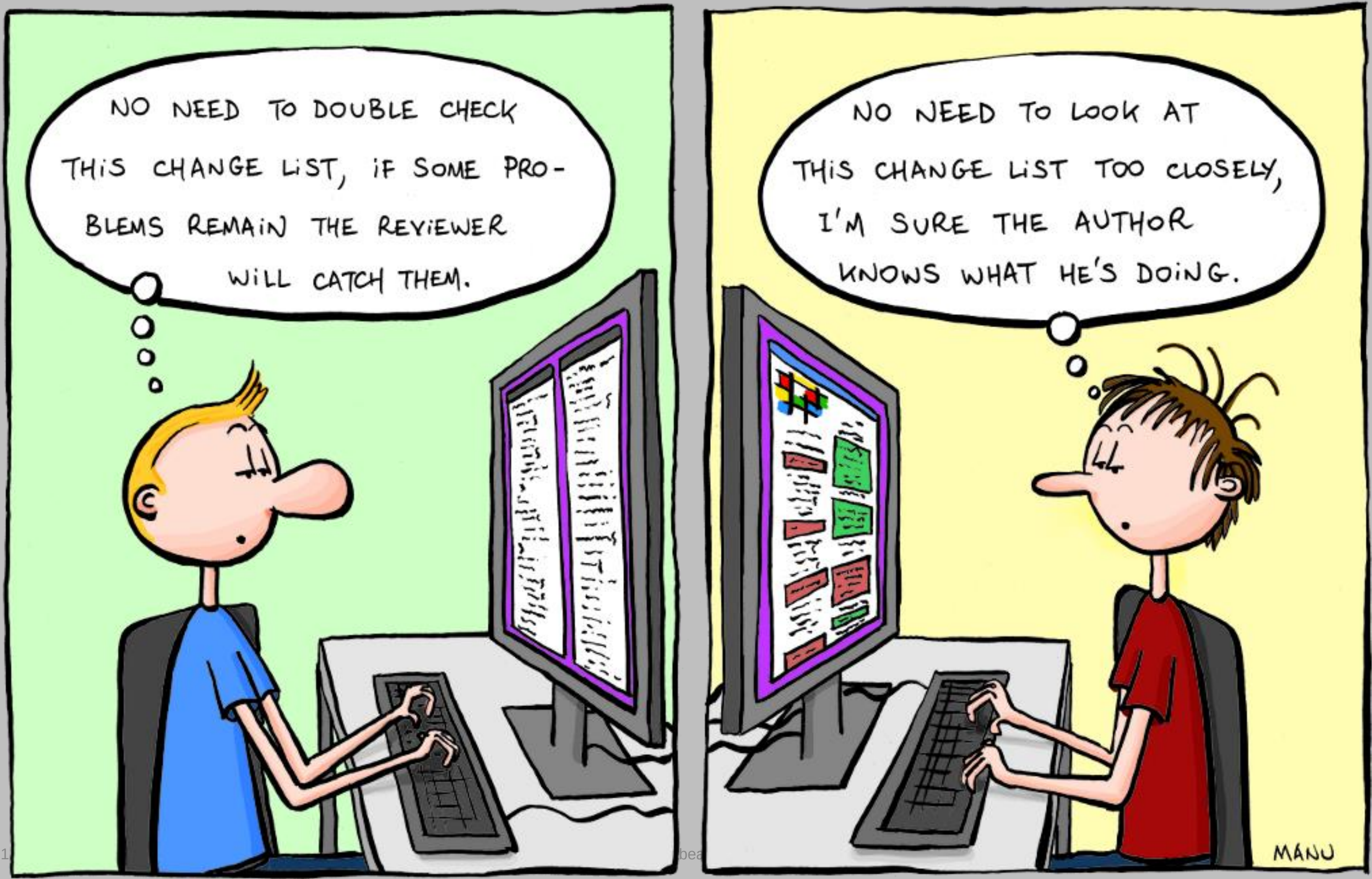
Roles: Author / Reviewer 1-*

Pre-Requisites

- Process backed by Team and Management
- Deal with criticism: Code quality is important
- Define standards: Syntax, naming, frameworks
- Comprehensible tasks
- Developers review own code before commit
- Define goals

A CODE REVIEW EXAMPLE

```java
public class ReviewCodeExample {

  public static BigDecimal FAC = new BigDecimal(0.1);

  public Collection<String> getCarNames() {
    List<Car> cars = getCarsFromDatabase();
    List<String> carNames = new ArrayList<>();
    for (Car car : cars) {
      if (!carNames.contains(car))
        carNames.add(car.getName());
    }
    return carNames;
  }
}
```


NO NEED TO DOUBLE CHECK THIS CHANGE LIST, IF SOME PRO- BLEMS REMAIN THE REVIEWER WILL CATCH THEM.
NO NEED TO LOOK AT THIS CHANGE LIST TOO CLOSELY, I'M SURE THE AUTHOR KNOWS WHAT HE'S DOING.
MANU

Who?

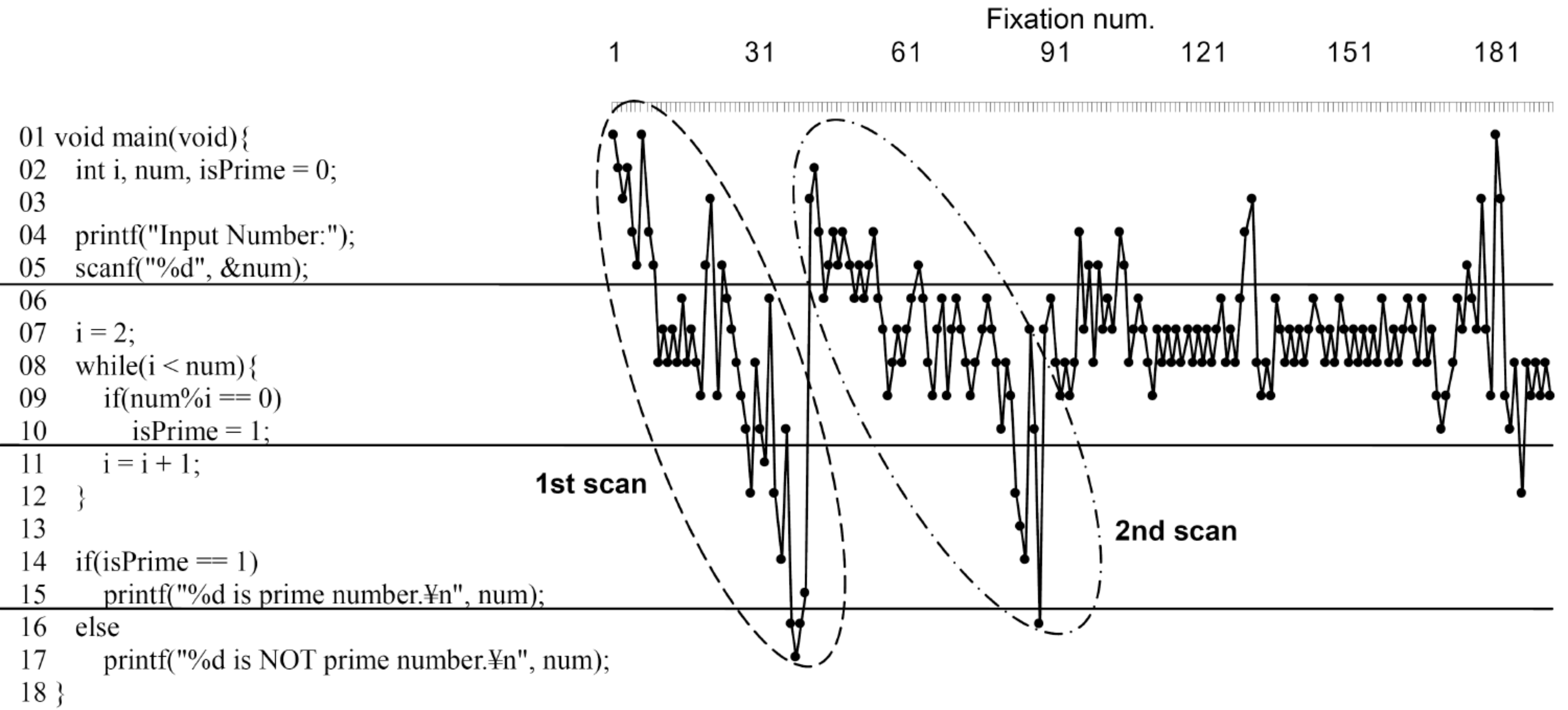
- Recommended: Every developer
- How many reviewers per request?
  - min. 2 with different focus
  - Recommended: Expert in domain of review

[1]

How?

- Read task and extract requirements
- Overview: What has changed?
- Have requirements been met?
- Check if code works by testing
- Inspect code line by line
- Identify issues, write comment and give priority
- Difficult: Identify missing parts
- Go slowly: 1 line changes at least 5min review

How: Eye Tracking



```
01 void main(void){
02     int i, num, isPrime = 0;
03
04     printf("Input Number:");
05     scanf("%d", &num);
06
07     i = 2;
08     while(i < num){
09         if(num%i == 0)
10             isPrime = 1;
11         i = i + 1;
12     }
13
14     if(isPrime == 1)
15         printf("%d is prime number.¥n", num);
16     else
17         printf("%d is NOT prime number.¥n", num);
18 }
```

When?

- Shortly after development/request
- Pre-Commit or Post-Commit
- Don’t postpone to day before release
- Maximum 90 min per review

What?

- Deviation from standard/requirements/code guidelines
- Code has to be readable. Prefer refactoring to comment
- Use small Checklists

Book (Java): T. Gee: What to Look for in a Code Review (2016)

What?

- Deviation from standard/requirements/code guidelines
- Code has to be readable. Prefer refactoring to comment
- Check coverage of new constants in if/switch
- if without else
- Correctness of exception handling
- Prefer immutable objects
- Spell check messages shown to users
- synchronized/transactions for atomic operations
- Watch out for Strings/Magic Numbers. Prefer value objects

Book (Java): T. Gee: What to Look for in a Code Review (2016)

Example Checklist

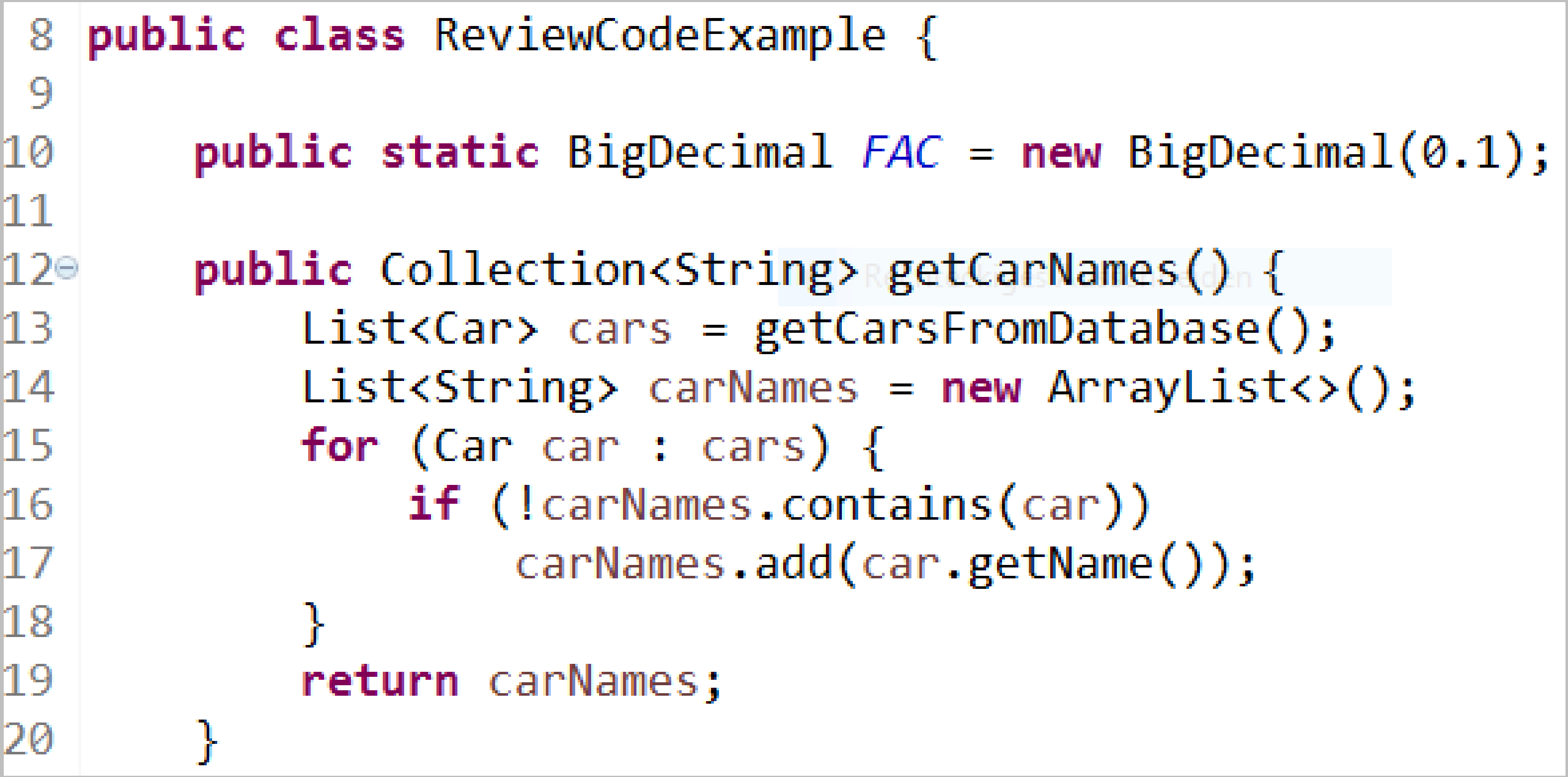
1. Documentation: All subroutines are commented in clear language.
2. Documentation: Describe what happens with corner-case input.
3. Documentation: Complex algorithms are explained and justified.
4. Documentation: Code that depends on non-obvious behavior in external libraries is documented with reference to external documentation.
5. Documentation: Units of measurement are documented for numeric values.
6. Documentation: Incomplete code is indicated with appropriate distinctive markers (e.g. “TODO” or “FIXME”).
7. Documentation: User-facing documentation is updated (online help, contextual help, tool-tips, version history).
8. Testing: Unit tests are added for new code paths or behaviors.
9. Testing: Unit tests cover errors and invalid parameter cases.
10. Testing: Unit tests demonstrate the algorithm is performing as documented.
11. Testing: Possible null pointers always checked before use.
12. Testing: Array indexes checked to avoid out-of-bound errors.
13. Testing: Don’t write new code that is already implemented in an existing, tested API.
14. Testing: New code fixes/implements the issue in question.
15. Error Handling: Invalid parameter values are handled properly early in the subroutine.
16. Error Handling: Error values of null pointers from subroutine invocations are checked.
17. Error Handling: Error handlers clean up state and resources no matter where an error occurs.
18. Error Handling: Memory is released, resources are closed, and reference counters are managed under both error and nonerror conditions.
19. Thread Safety: Global variables are protected by locks or locking subroutines.
20. Thread Safety: Objects accessed by multiple threads are accessed only through a lock.
21. Thread Safety: Locks must be acquired and released in the right order to prevent deadlocks, even in error-handling code.
22. Performance: Objects are duplicated only when necessary.
23. Performance: No busy-wait loops instead of proper thread synchronization methods.
24. Performance: Memory usage is acceptable even with large inputs.
25. Performance: Optimization that makes code harder to read should only be implemented if a profiler or other tool has indicated that the routine stands to gain from optimization. [10]

Everything?

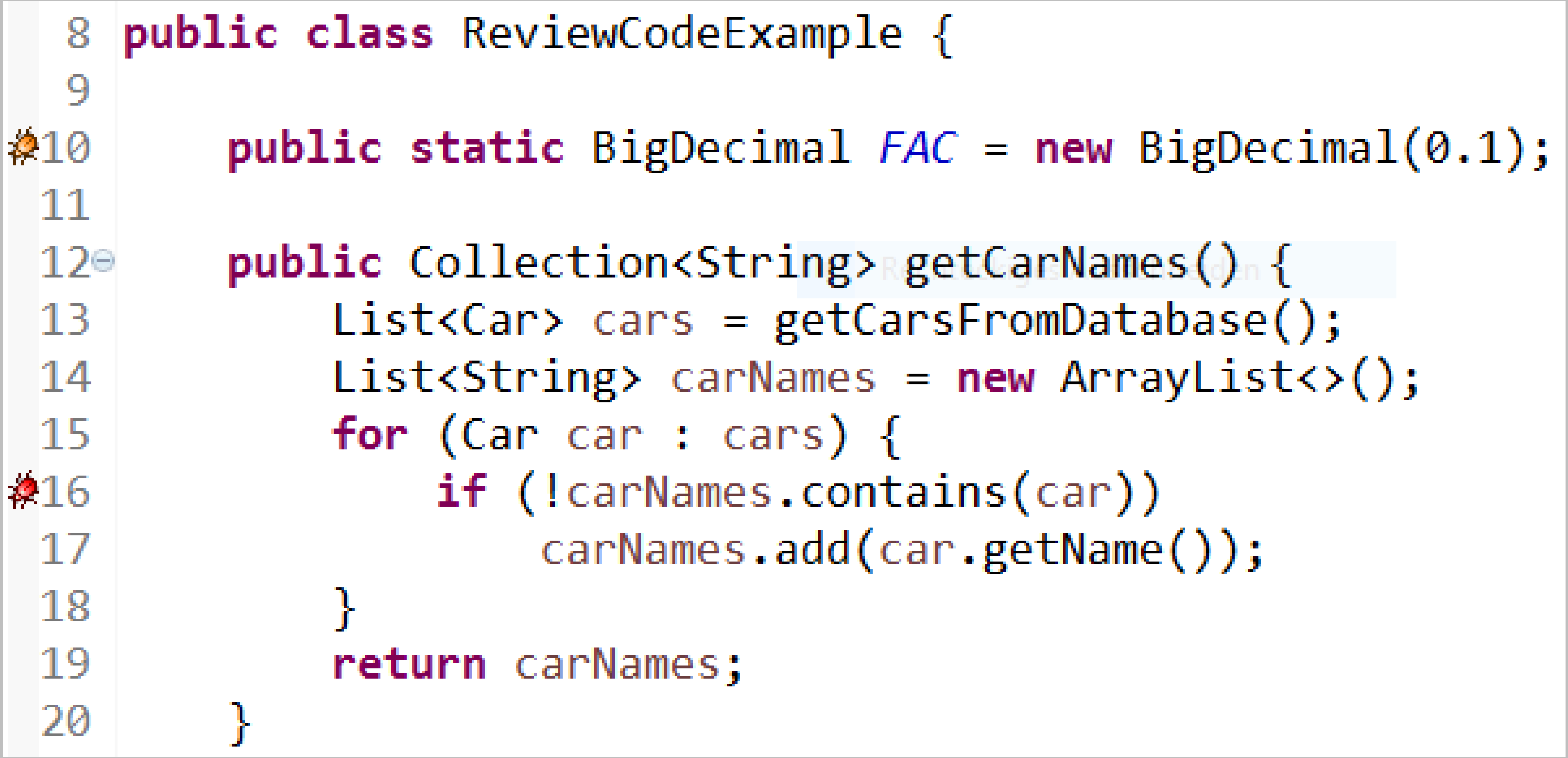
- Just get started, every review helps

- Start with high risk changes:
  - Change in important calculations
  - Safety critical code, e.g. authentication
  - Code without test coverage
  - Code of new team members
  - Change sets with high number of files touched

SMALL TOOLS



```java
public class ReviewCodeExample {

    public static BigDecimal FAC = new BigDecimal(0.1);

    public Collection<String> getCarNames() {
        List<Car> cars = getCarsFromDatabase();
        List<String> carNames = new ArrayList<>();
        for (Car car : cars) {
            if (!carNames.contains(car))
                carNames.add(car.getName());
        }
        return carNames;
    }
```



```java
public class ReviewCodeExample {

    public static BigDecimal FAC = new BigDecimal(0.1);

    public Collection<String> getCarNames() {
        List<Car> cars = getCarsFromDatabase();
        List<String> carNames = new ArrayList<>();
        for (Car car : cars) {
            if (!carNames.contains(car))
                carNames.add(car.getName());
        }
        return carNames;
    }
```

FindBugs

- Static code analysis
- Explanation with possible solution
  - Bug: Method ReviewCodeExample.getFactor() **passes double value to BigDecimal Constructor**
  - This method calls the BigDecimal constructor that takes a double, and passes a literal double constant value. Since the use of BigDecimal is to get better precision than double, by passing a double, you only get the precision of double number space. To take advantage of the BigDecimal space, **pass the number as a string**.

Automated Review

- Errors which can easily get overlooked
  - Naming and formatting
  - Wrong API usage (BigDecimal example)
- Run before manual review
  - Developer before commit
  - Build-System/Continuous Integration
- Important: Handling of False-Positives
  - FindBugs @SuppressFBWarnings

Free to use:

- FindBugs
- Checkstyle
- PMD
- JQAssistant
- SonarQube

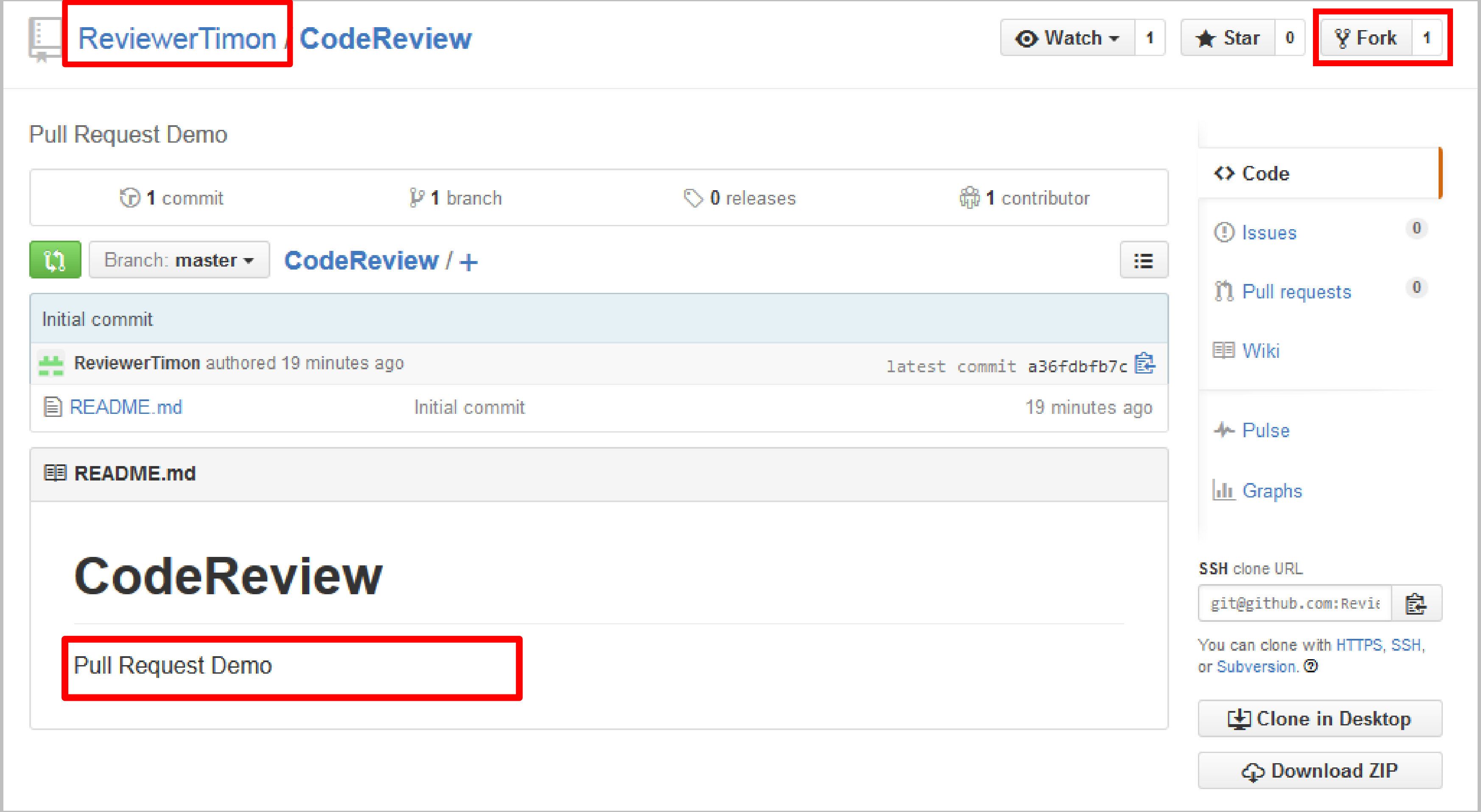
Payed:

- Coverity

TOOL-BASED REVIEW

Example: GitHub Pull Request

- Web-based Review
- Commit/Branch/Task-based Review
- Fork project / create branch / edit file on master
- Create Pull Request
- Notification for Repo Owners
- Can add (line based) review comments on files
- Close or accept pull request



Pull Request Demo

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master   CodeReview / +

Initial commit

ReviewerTimon authored 19 minutes ago   latest commit a36fdbfb7c

README.md   Initial commit   19 minutes ago

README.md

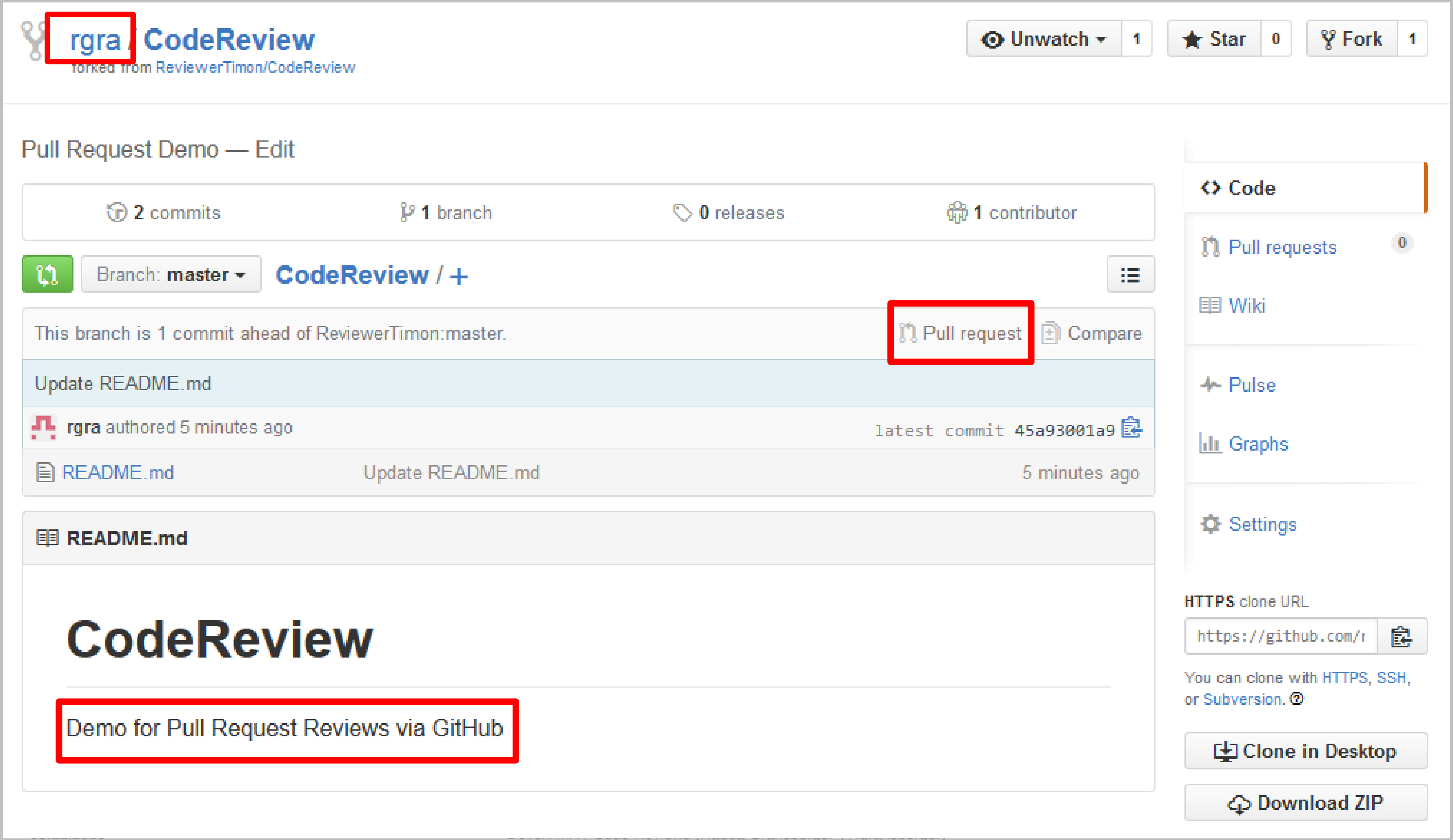
CodeReview

Pull Request Demo

SSH clone URL

`git@github.com:Revie`

You can clone with HTTPS, SSH, or Subversion.



rgra / CodeReview
forked from ReviewerTimon/CodeReview

Unwatch 1 | Star 0 | Fork 1

Pull Request Demo — Edit

2 commits | 1 branch | 0 releases | 1 contributor

Branch: master | CodeReview / +

This branch is 1 commit ahead of ReviewerTimon:master. Pull request | Compare

Update README.md

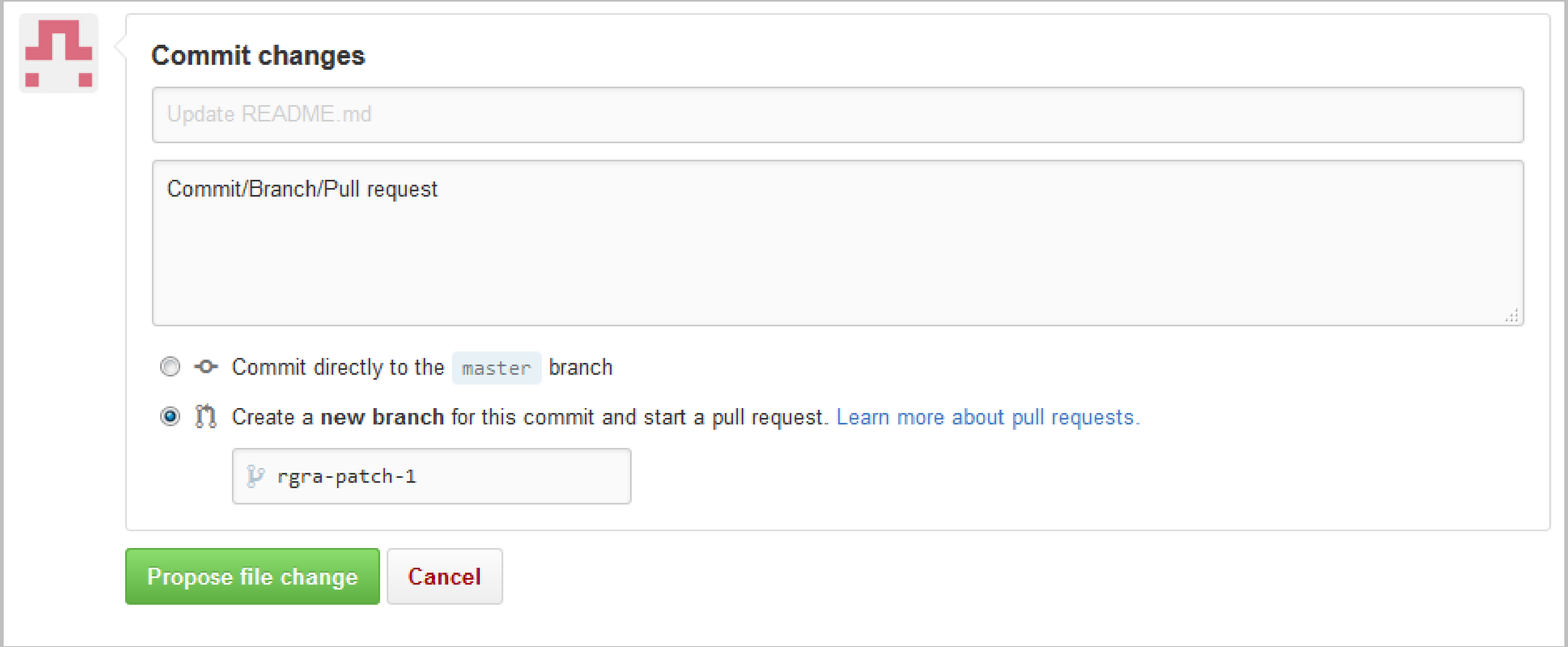
rgra authored 5 minutes ago   latest commit 45a93001a9

README.md	Update README.md	5 minutes ago

README.md

CodeReview

Demo for Pull Request Reviews via GitHub



Commit changes

Update README.md

Commit/Branch/Pull request

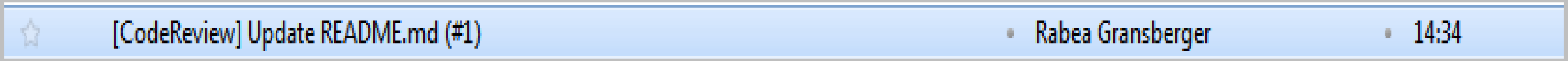
( ) Commit directly to the `master` branch

(●) Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

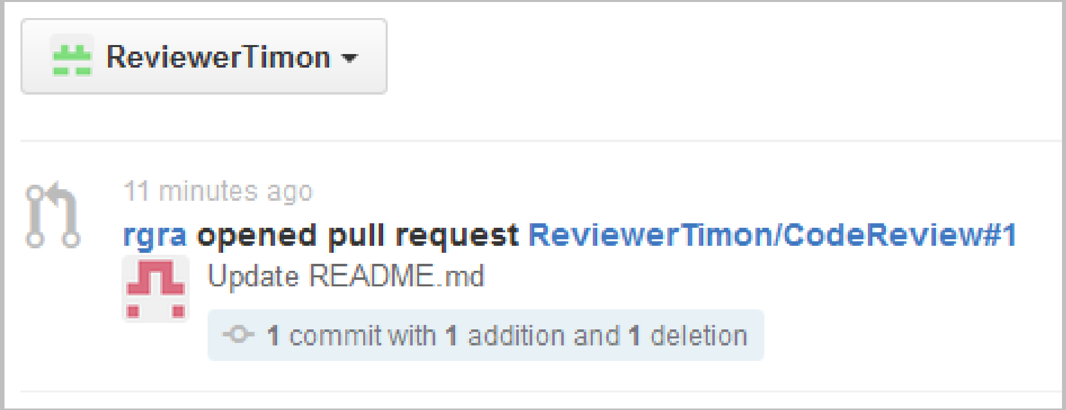
rgra-patch-1

Pull Request Review

- Mail: Notification Pull Request / Review


[CodeReview] Update README.md (#1) • Rabea Gransberger • 14:34

- Web: Pending Pull Request / Review

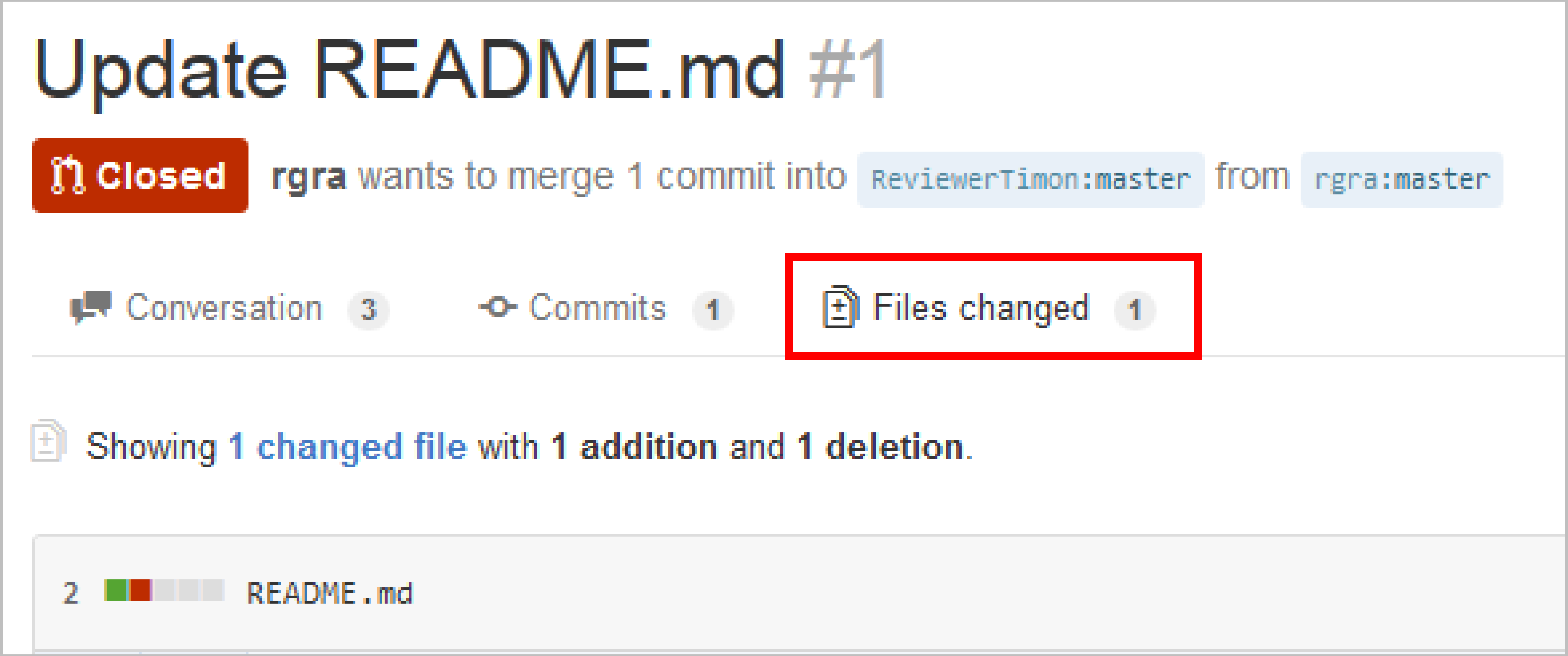

ReviewerTimon

11 minutes ago

rgra opened pull request ReviewerTimon/CodeReview#1

Update README.md

1 commit with 1 addition and 1 deletion



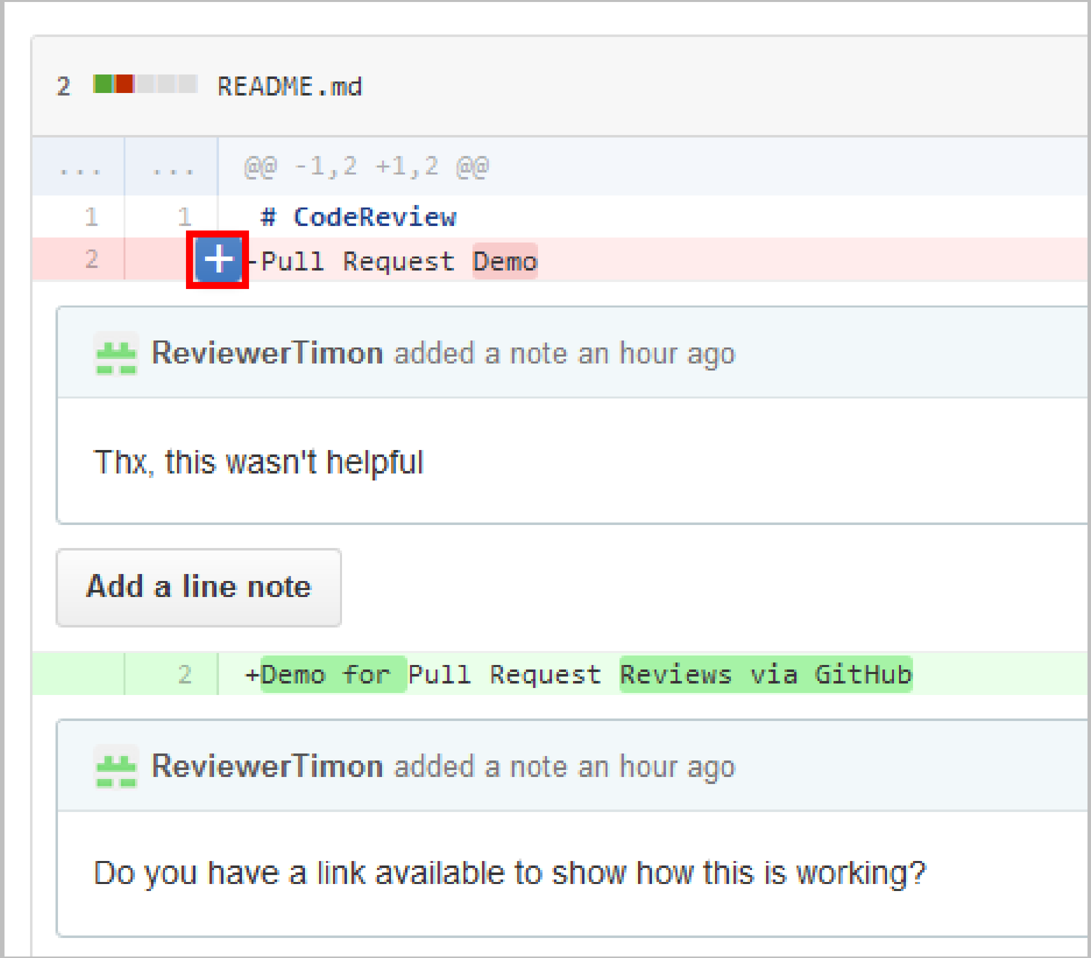
Update README.md #1

Closed rgra wants to merge 1 commit into ReviewerTimon:master from rgra:master

Conversation 3 Commits 1 Files changed 1

Showing 1 changed file with 1 addition and 1 deletion.

2 README.md



2 README.md

```
...  ...  @@ -1,2 +1,2 @@
1    1     # CodeReview
2         -Pull Request Demo
```

ReviewerTimon added a note an hour ago

Thx, this wasn't helpful

```
     2    +Demo for Pull Request Reviews via GitHub
```

ReviewerTimon added a note an hour ago

Do you have a link available to show how this is working?

Review Tools

- Reviews in pull requests Github
- JetBrains Upsource *
- Atlassian Crucible
- Gerrit
- Review Board
- Phabricator Differential
- SmartBear Collaborator / CodeReviewer*
- ReviewClipse*

(* with IDE integration)

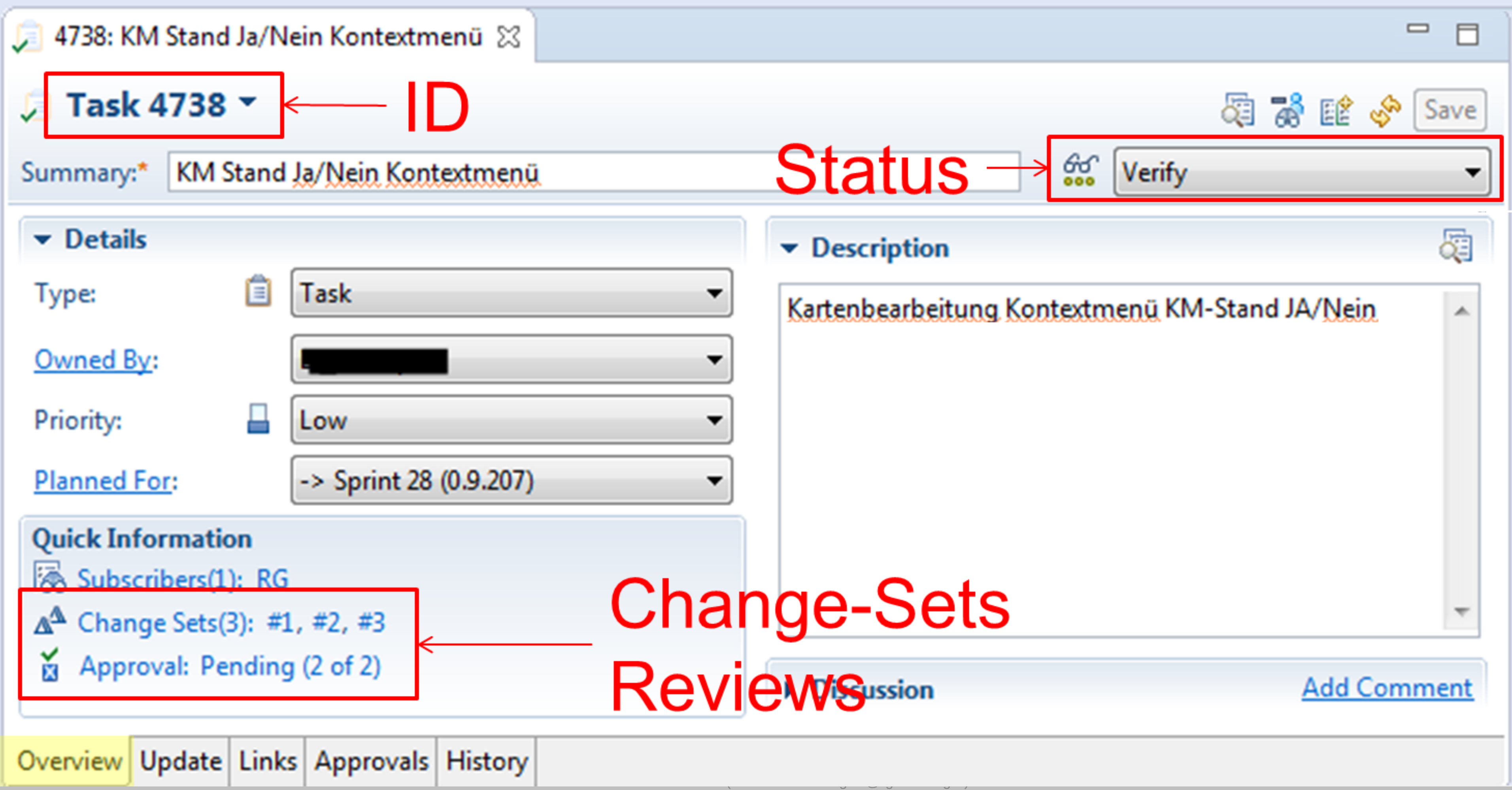
Tools Checklist

- Automatic Review creation via hooks from SCM
- Adding new changes to existing review
- Pre-/Post-Commit Review Support
- Patch/Live-Code
- Where are comments saved? Embedded in code, separate XML/Database?
- Overview with all pending reviews
- Tracking which code still needs review
- Comments and priorities and possibility to mark comment as closed
- Webpage / IDE Integration
- Notifications by mail
- Review by task / whole code base supported
- Statistics to check effects of review / improve process

Example: Tools in your IDE

- IDE provides sufficient support for reviews
- SCM Integration
- Issue Tracker Integration
- Task Tags

- Example: Review at MEKOS with Eclipse/RTC



4738: KM Stand Ja/Nein Kontextmenü

Task 4738

Summary:* KM Stand Ja/Nein Kontextmenü

Verify

Details

Type: Task

Owned By: [redacted]

Priority: Low

Planned For: -> Sprint 28 (0.9.207)

Quick Information

Subscribers(1): RG

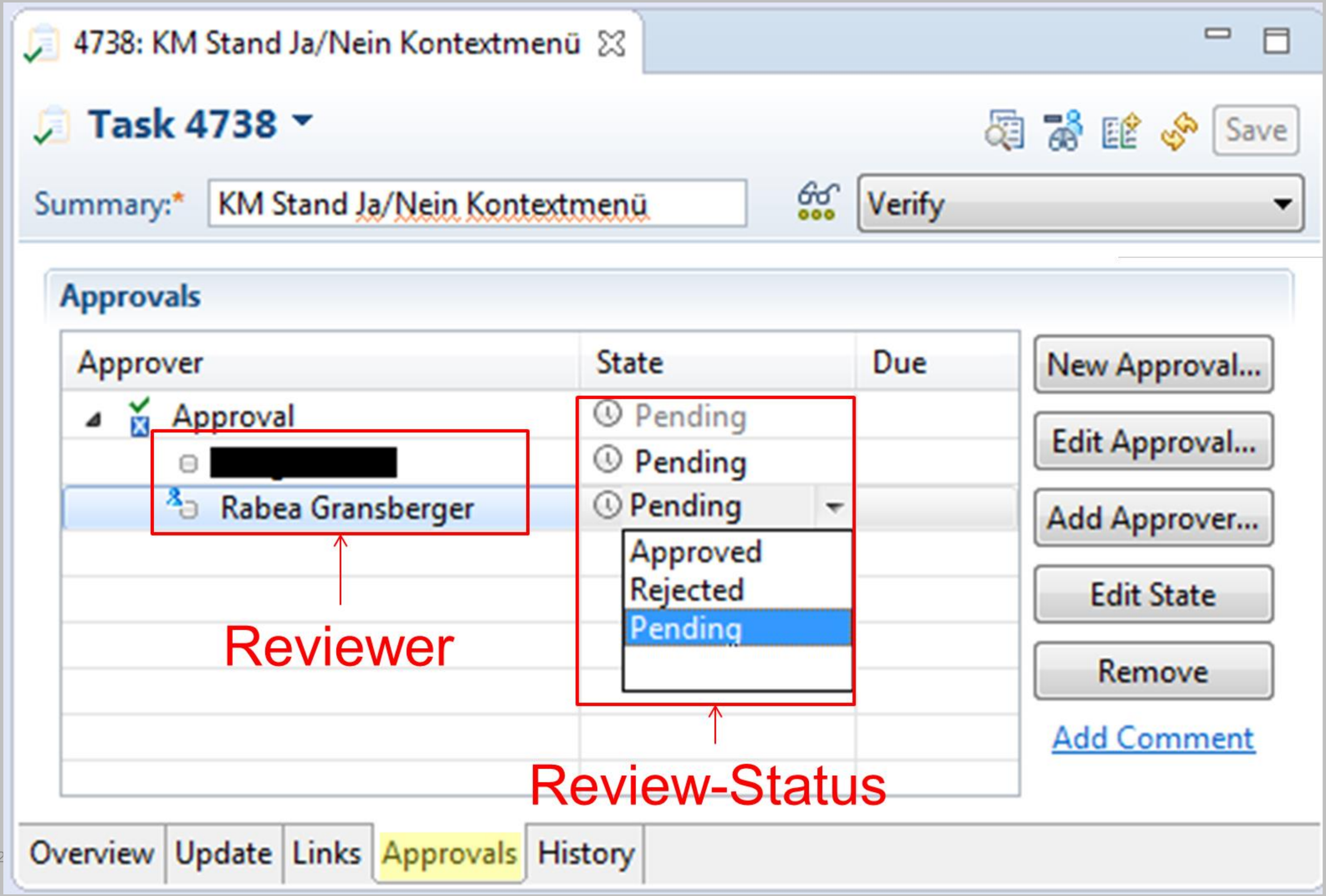
Change Sets(3): #1, #2, #3

Approval: Pending (2 of 2)

Description

Kartenbearbeitung Kontextmenü KM-Stand JA/Nein

Discussion



4738: KM Stand Ja/Nein Kontextmenü

Task 4738

Summary:* KM Stand Ja/Nein Kontextmenü

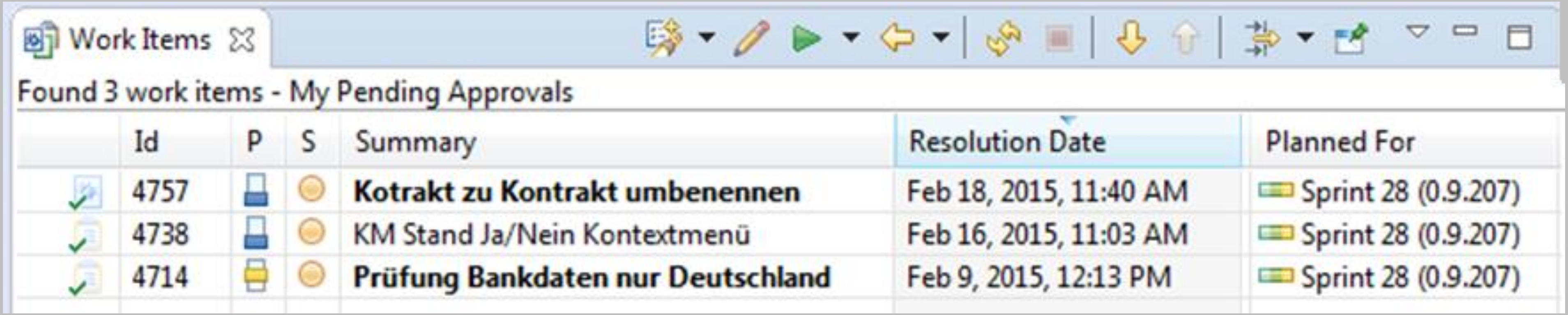
Verify

Approvals

Approver	State	Due
Approval	Pending	
[redacted]	Pending	
Rabea Gransberger	Pending	

Rational Team Concert

- Reviewer can query pending reviews

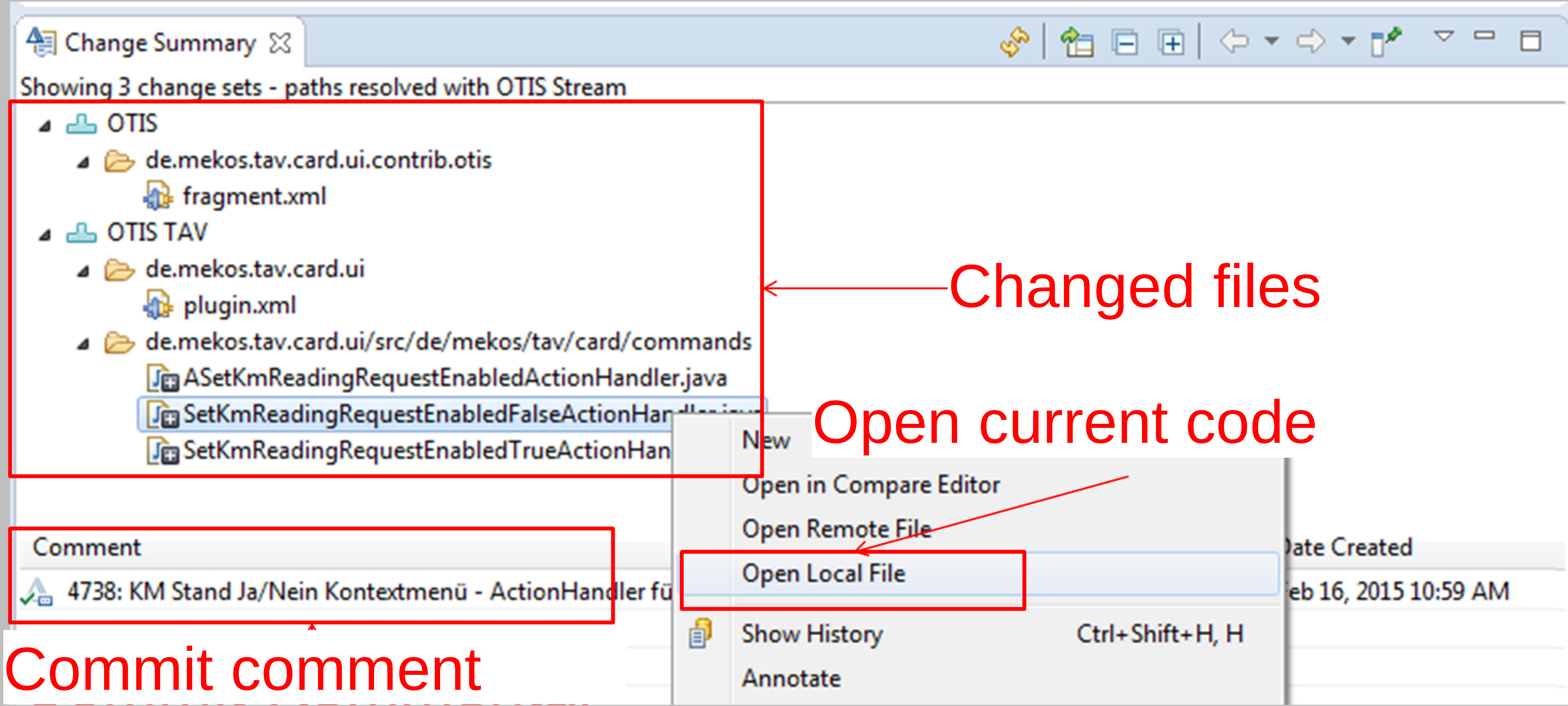


Work Items

Found 3 work items - My Pending Approvals

	Id	P	S	Summary	Resolution Date	Planned For
✓	4757		●	**Kotrakt zu Kontrakt umbenennen**	Feb 18, 2015, 11:40 AM	Sprint 28 (0.9.207)
✓	4738		●	KM Stand Ja/Nein Kontextmenü	Feb 16, 2015, 11:03 AM	Sprint 28 (0.9.207)
✓	4714		●	**Prüfung Bankdaten nur Deutschland**	Feb 9, 2015, 12:13 PM	Sprint 28 (0.9.207)

- Select Work-Item with double click
- Open attached Change-Sets to review code



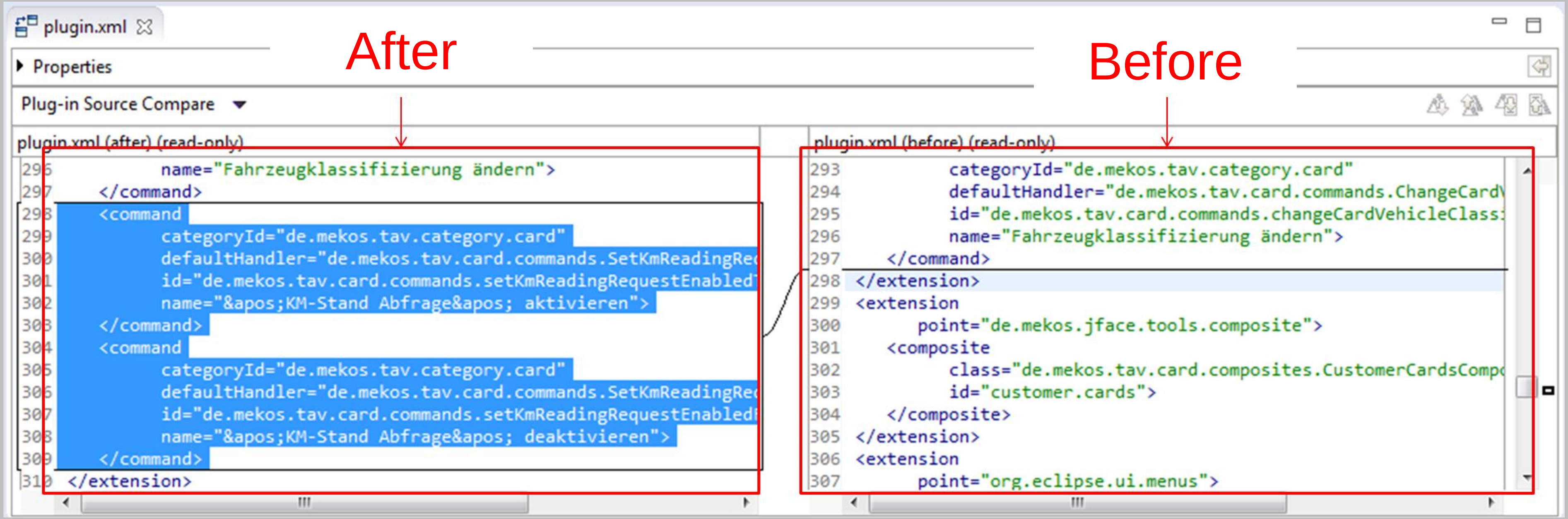
Change Summary

Showing 3 change sets - paths resolved with OTIS Stream

OTIS
de.mekos.tav.card.ui.contrib.otis
fragment.xml
OTIS TAV
de.mekos.tav.card.ui
plugin.xml
de.mekos.tav.card.ui/src/de/mekos/tav/card/commands
ASetKmReadingRequestEnabledActionHandler.java
SetKmReadingRequestEnabledFalseActionHan
SetKmReadingRequestEnabledTrueActionHan

Comment	Date Created
4738: KM Stand Ja/Nein Kontextmenü - ActionHandler fü	eb 16, 2015 10:59 AM

RTC: Diff View


plugin.xml
After
Before
Properties
Plug-in Source Compare
plugin.xml (after) (read-only)
plugin.xml (before) (read-only)

```
            name="Fahrzeugklassifizierung ändern">
    </command>
    <command
            categoryId="de.mekos.tav.category.card"
            defaultHandler="de.mekos.tav.card.commands.SetKmReadingRe
            id="de.mekos.tav.card.commands.setKmReadingRequestEnabled
            name="&apos;KM-Stand Abfrage&apos; aktivieren">
    </command>
    <command
            categoryId="de.mekos.tav.category.card"
            defaultHandler="de.mekos.tav.card.commands.SetKmReadingRe
            id="de.mekos.tav.card.commands.setKmReadingRequestEnabled
            name="&apos;KM-Stand Abfrage&apos; deaktivieren">
    </command>
 </extension>
```

```
            categoryId="de.mekos.tav.category.card"
            defaultHandler="de.mekos.tav.card.commands.ChangeCard
            id="de.mekos.tav.card.commands.changeCardVehicleClass
            name="Fahrzeugklassifizierung ändern">
    </command>
 </extension>
 <extension
       point="de.mekos.jface.tools.composite">
    <composite
          class="de.mekos.tav.card.composites.CustomerCardsComp
          id="customer.cards">
    </composite>
 </extension>
 <extension
       point="org.eclipse.ui.menus">
```

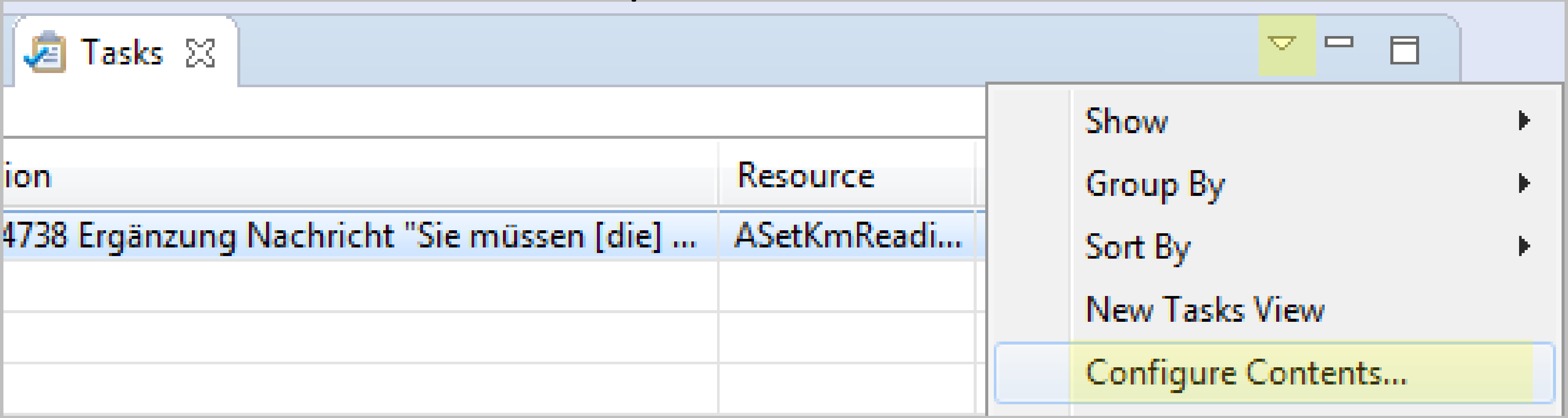
```java
public class ReviewCodeExample {

  public static BigDecimal FAC = new BigDecimal(0.1);

  public Collection<String> getCarNames() {
    List<Car> cars = getCarsFromDatabase();
    List<String> carNames = new ArrayList<>();
    for (Car car : cars) {
      //FIXME 4738 Use set instead of List
      if (!carNames.contains(car))
        carNames.add(car.getName());
    }
    return carNames;
  }
```

Review with Eclipse

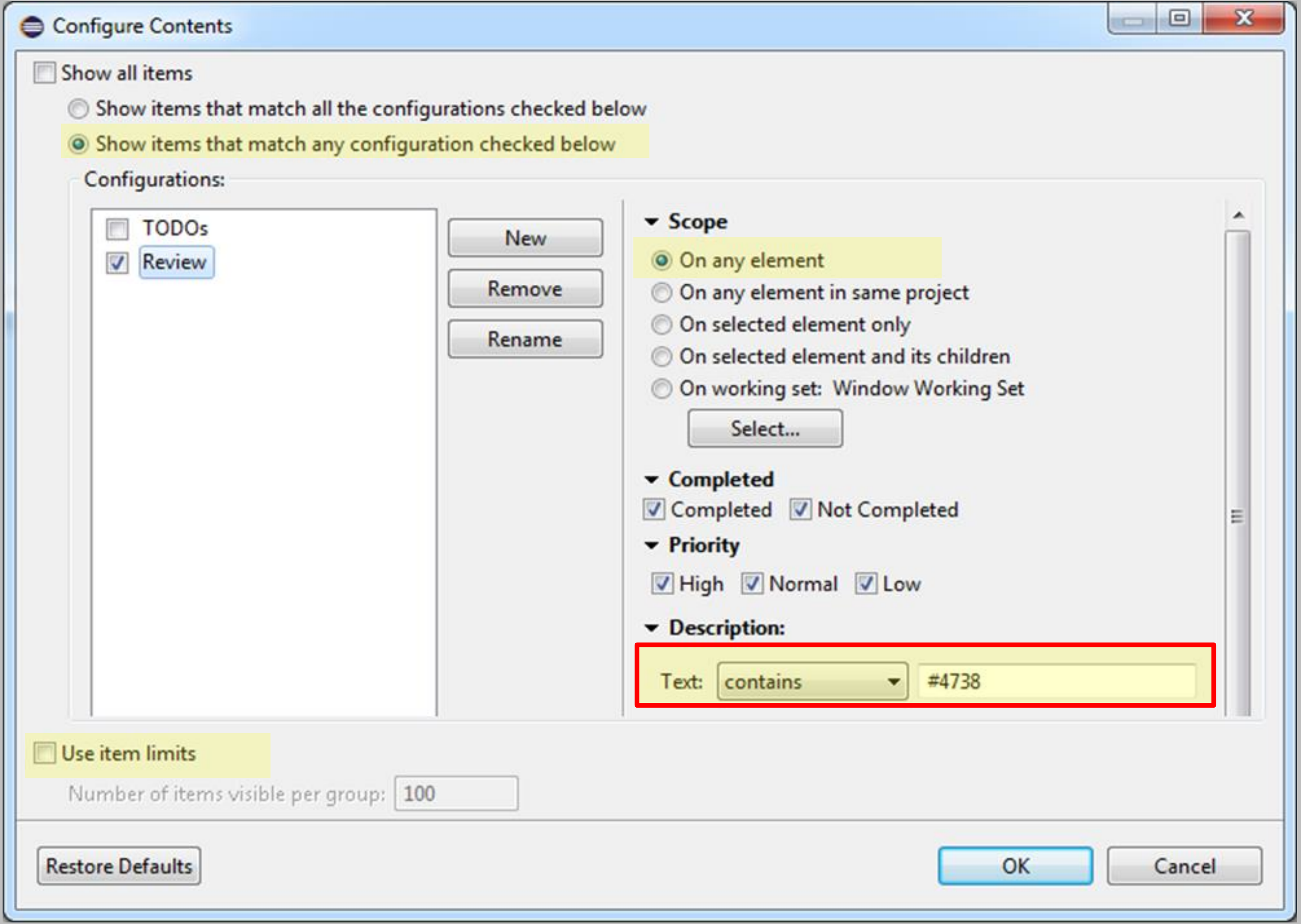
- Write comments in code
- Prefix with Task-Tags TODO/FIXME + ID
  //TODO #4738
- Deliver comments with commit message “Review”
- Review gets Rejected
  => Work Item Reopen

View Review Comments

- Author gets notified about rejected review
- Find comments with Eclipse View Tasks



4738 Ergänzung Nachricht "Sie müssen [die] ... ASetKmReadi...


Configure Contents

[ ] Show all items

( ) Show items that match all the configurations checked below

(●) Show items that match any configuration checked below

Configurations:

[ ] TODOs
[✓] Review

Scope

(●) On any element
( ) On any element in same project
( ) On selected element only
( ) On selected element and its children
( ) On working set: Window Working Set

Completed

[✓] Completed [✓] Not Completed

Priority

[✓] High [✓] Normal [✓] Low

Description:

Text: contains #4738

[ ] Use item limits

Number of items visible per group: 100

Rework

Author

- Rewrite code and fix all comments
- Remove task tag comments
- Commit with comment „Rework Review“
- Work-Item to Verification state
- Invite reviewer for next review

Reviewer

- All task tags removed
- Re-Review code:
- Changes between “Review” and “Rework” changesets

STATISTICS

Statistics

Some review tools help to quantify positive effects of review

Examples:

- Issues by classification
- Found issues
- % reviewed code compared to full code base

Found Issues

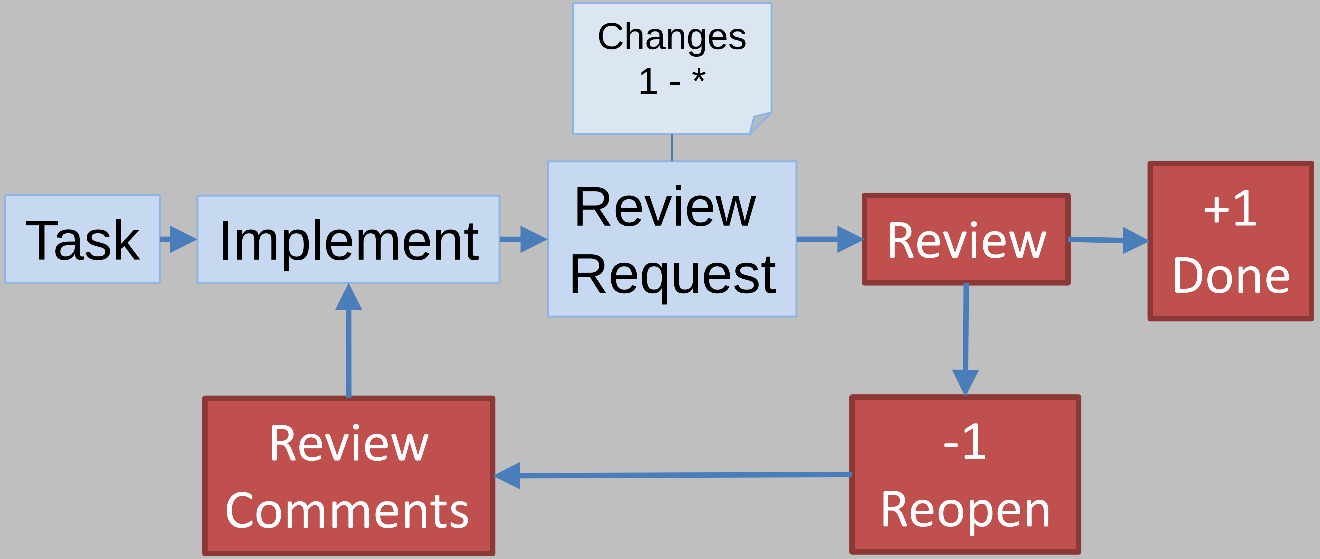
• Maintenance	71,7%
– Naming, Comments	16,7 %
– API Use/Formatting	13,0 %
– Structure/Organisational	16,2 %
– Solution Approach	20,6 %
• Functional Problems	21,4%
• False positives	7,5%

x

Industrial review, domain: Engineering, 9 Reviews, 1-4 Reviewer, 388 issues found [12]

PROCESS VARIATIONS

Example: Task based review process



Roles: Author / Reviewer 1-*

Process embedding

Unit of work (IV-C1)
- Release
- Story/ Requirement
- Task
- Push/Pull/Comb. commit
- Singular commit

Trigger (IV-C2)
- Tool
- Conventions

Swift completion (IV-C5)
- Priority
- WIP limit
- Time slot
- Author’s responsibility

Unreviewed Release Prevention (IV-C4)
- Organizational
- Pre commit review
- Release branch

Publicness (IV-C3)
- Pre-commit
- Post-commit

Blocking of process (IV-C6)
- Full Follow-up
- Wait for Review
- No Blocking

[20]

Reviewers

Rules Count/Skip (IV-D2)
- Component
- Author’s experience
- Lifecycle phase
- Change size
- Pair programming
- Reviewer’s choice
- Author’s choice

Population (IV-D3)
- Everybody
- Elite
- Fixed

Assignment (IV-D4)
- Pull
- Push
- Mix
- Fixed

Assignment Tool (IV-D5)
- No Tool
- Reviewer Recommendation

[20]

Checking

Interaction (IV-E1)
- On-demand
- Asynchronous Discussion
- Meeting with author
- Meeting without author

Temporal Arrangement (IV-E2)
- Parallel
- Sequential

Roles (IV-E3)
- Yes
- No

Reviewer changes code (IV-E4)
- Never
- Sometimes

Detection Aids (IV-E5)
- Checklists
- Static code analysis
- Testing

[20]

Feedback

Communication of issues
(IV-F1)
- Written
- Oral only
- Oral stored

Handling of issues
(IV-F2)
- Resolve
- Reject
- Postpone
- Ignore

[20]

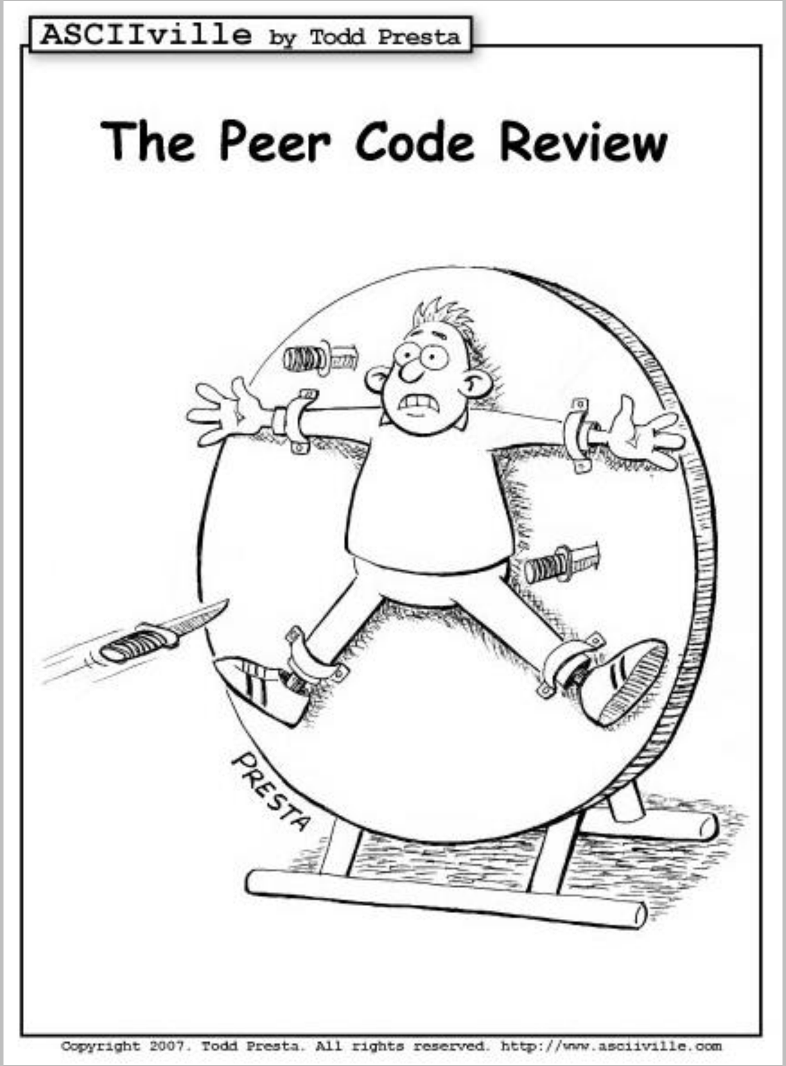
Overarching

Use of metrics (IV-G1)
Metrics in use
No metrics use

Tool specialization (IV-G2)
General-purpose
Specialized

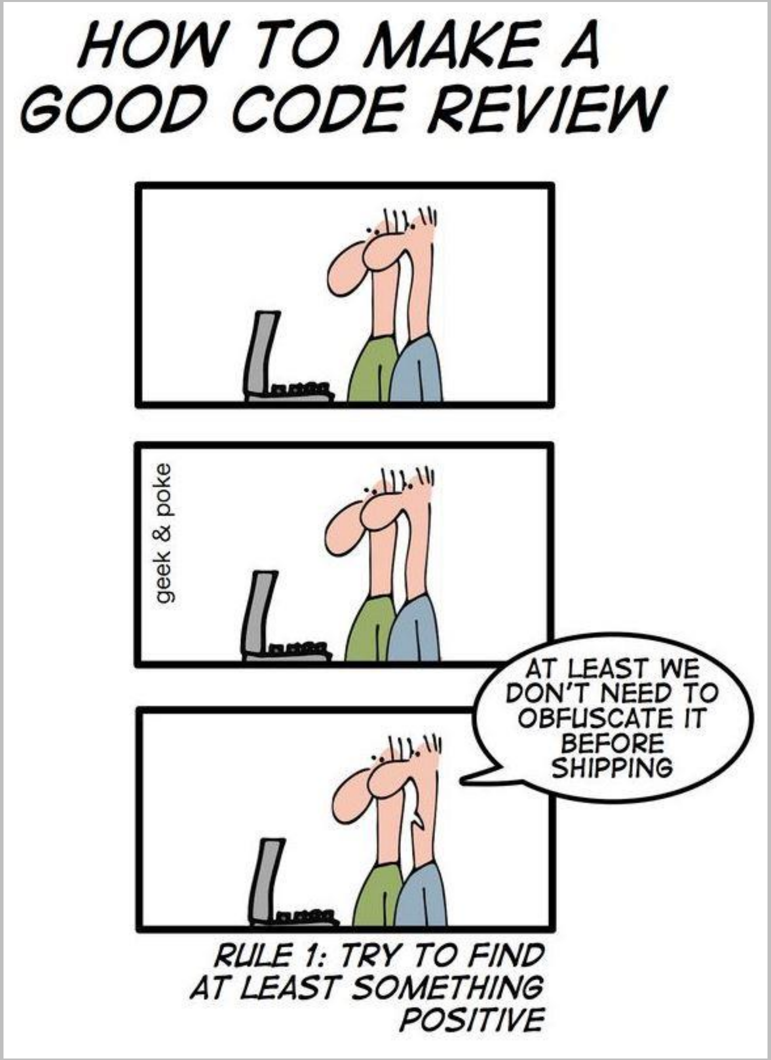
[20]

TIPS & PRACTICAL EXPERIENCE


ASCIIville by Todd Presta
The Peer Code Review
PRESTA
Copyright 2007. Todd Presta. All rights reserved. http://www.asciiville.com

Tips for developers

- Mistakes = Learn, don’t take personal!
- Reviews don’t replace questions. Talk!
- Refactoring in separate change set
- Checklist review own changes before commit
- Remind reviewer of important reviews
- Reviewer isn’t necessarily right. Discuss

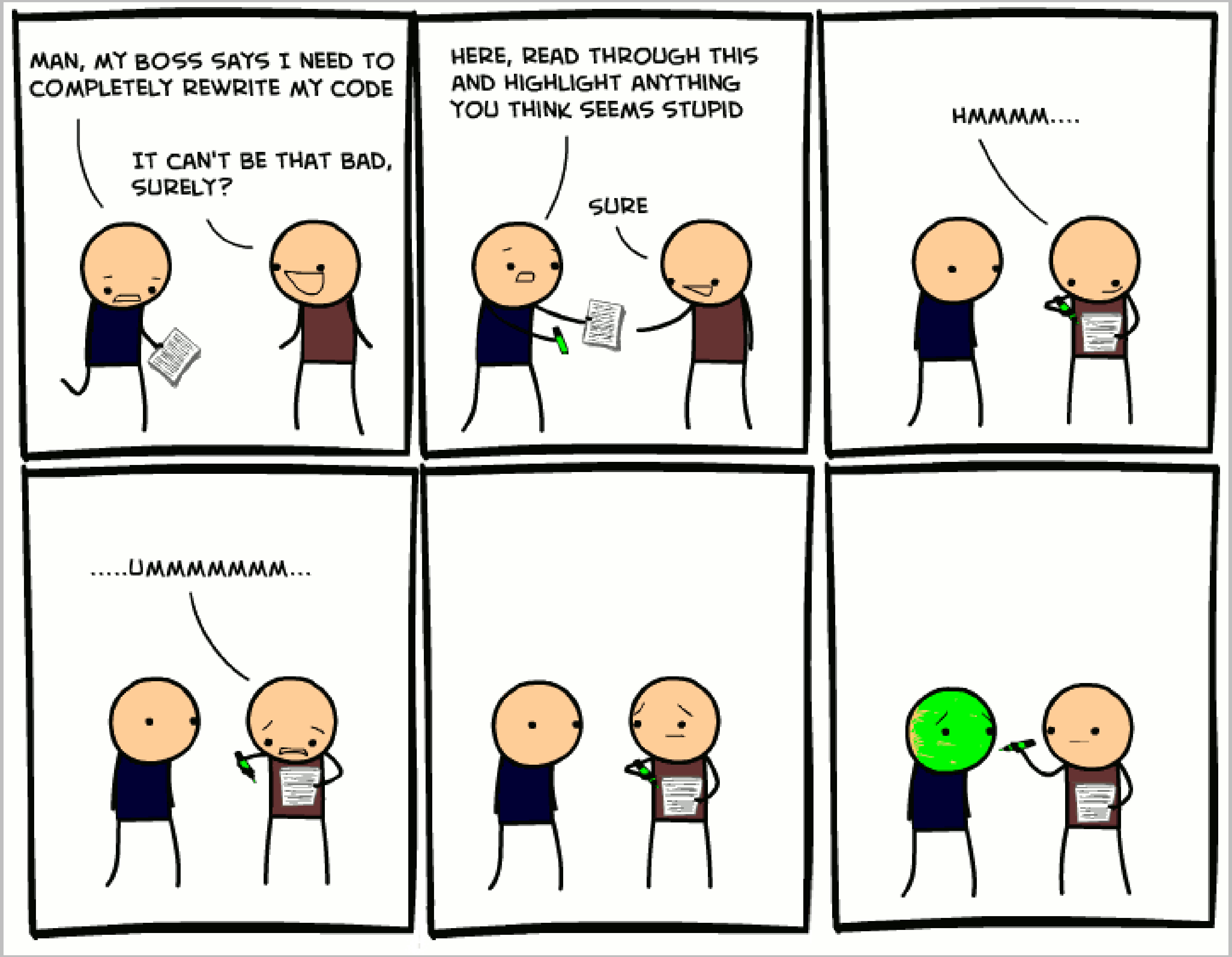

HOW TO MAKE A GOOD CODE REVIEW
geek & poke
AT LEAST WE DON'T NEED TO OBFUSCATE IT BEFORE SHIPPING
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Tips for Reviewers

- Make sure you are not disturbed
- Prioritize if too many requests
- Take time, don’t rush and accept
- Don’t postpone reviews with many files
- If you can’t test it, ask for walkthrough

Tips for Reviewers

- ~~Wrong!~~ Provide advice on how to do better
- Question don’t criticize. Don’t get personal!
- Don’t fix code while reviewing (Bad fixes)
- Praise good code and personal advances
- Learn from team mates code


MAN, MY BOSS SAYS I NEED TO COMPLETELY REWRITE MY CODE
IT CAN'T BE THAT BAD, SURELY?
HERE, READ THROUGH THIS AND HIGHLIGHT ANYTHING YOU THINK SEEMS STUPID
SURE
HMMMM....
.....UMMMMMMM...

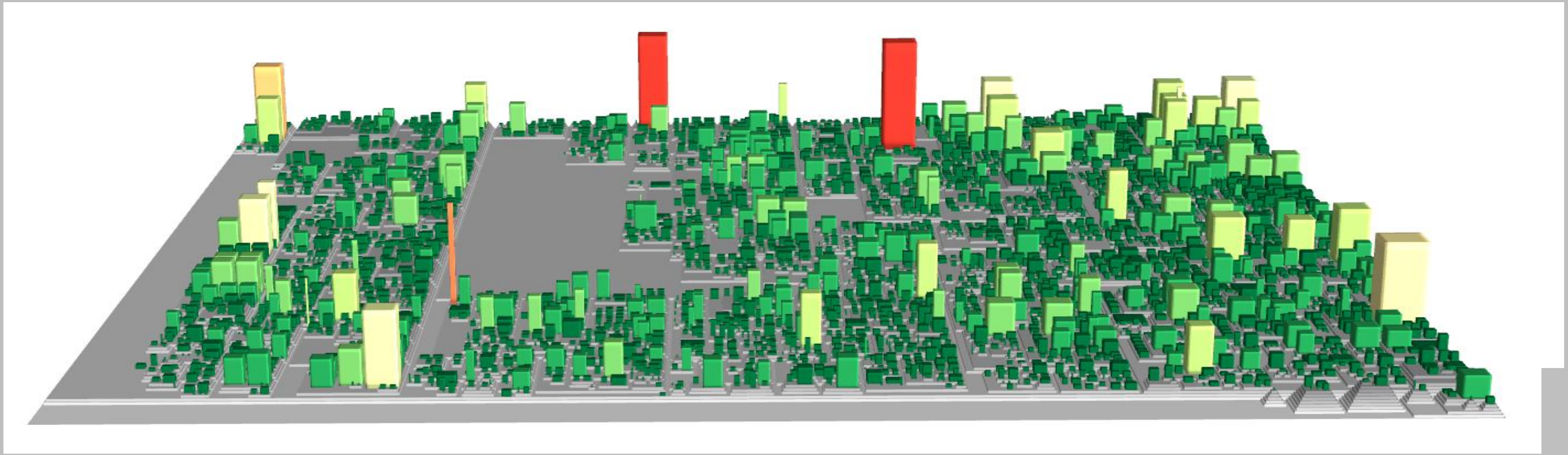
Social Aspects

- Reviews are unnecessary, they just cost time.
- Process is boring
- Author and reviewer get into conflict
- Team members block process / approve fast

Social Aspects

- Experience != Quality
- Critique can cause depression
- Big Brother Effect
- Review gets rejected x-times

Code City



Codetrails Code City Plugin

Related Tools / Concepts

- Code Coverage
- Code City / Code as a crime scene
- Continuous Integration Server
- Continuous Testing
- Mutation testing
- Random testinc

SUMMARY

Summary

- Begin slowly & use existing tools
- Define standards/checklists and use them
- Configure tools for automated reviews
- Create relaxed atmosphere
- Reward: Less support calls / happy customers
- Lowers overall project cost
- Adjust process as you go

Summary

- Speak to each other

- Every code review helps!

Questions?

Slides / Recordings:

- http://rgra.github.io

Contact information:

- Rabea Gransberger (LinkedIn, Xing)
- Twitter: @rgransberger

Feedback welcome!

Sources

1. Understanding Open Source Software Peer Review: Review Processes, Parameters and Statistical Models, and Underlying Behaviours and Mechanisms, Rigby, Dissertation, 2011
2. Convergent Contemporary Software Peer Review Practices, Rigby, 2013
3. Software Quality in 2002: A survey of the state of the art, Capers Jones, 2002
4. IEEE Standard for Software Reviews and Audits, IEEE Std 1028™-2008
5. Modernizing The Peer Code Review Process, KLOCWORK, White Paper, 2010
6. Code Reviews should be the universal rule of serious Software Development, Chhabra, Blog, 2012
7. How to hold a more effective code review, Stellman & Green, Blog, 2008
8. Code Review in Four Steps, Hayes, Blog, 2014
9. 11 proven practices for more effective, efficient peer code review, Cohen, 2011
10. Best Kept Secrets of Peer Code Review, Cohen, Smart Bear Inc., 2006
11. Don't waste time on Code Reviews, Bird, Blog, 2014
12. Code Review Defects, Mäntylä & Lassenius, 2007
13. The Ten Commandments of Egoless Programming, Atwood, Blog, 2006
14. Improve Quality and Morale: Tips for Managing the Social Effects of Code Review, Smartbear, 2011
15. Code Reviews Resourcen von Tobias Baum
16. What to Look for in a Code Review, Gee, 2016
17. 20 Best Code Review Tools for Developers, Blog, 2015
18. Effektiver Einsatz von Code Review, OIO, 2015
19. Technische Schulden in Architekturen erkennen und beseitigen, Dr. C. Lilienthal, 2016
20. A Faceted Classification Scheme for Change-Based Industrial Code Review Processes, T. Baum, 2016