Investing in code reviews for better research software

Thibault Lestang   Dominik Krzemiński   Valerio Maggio

Part 1

What is a code review?

Code review?

Main benefits:

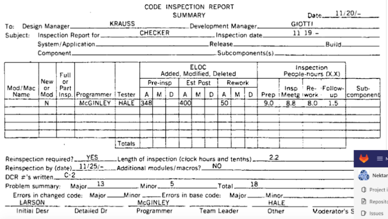
(1) Catching bugs

(2) Ensuring quality standard

(3) Spreading knowledge

(4) Training new developers

From formal inspections to code review



CODE INSPECTION REPORT
SUMMARY

Date 11/20/–

To: Design Manager KRAUSS Development Manager GIOTTI
Subject: Inspection Report for CHECKER Inspection date 11 19 –
System/Application Release Build
Component Subcomponents(s)

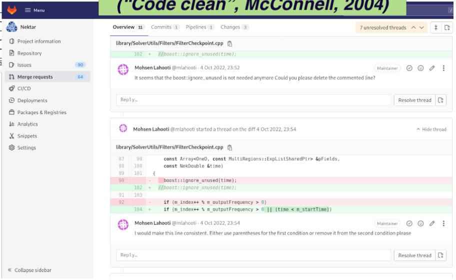
					ELOC Added, Modified, Deleted									Inspection People-hours (X.X)				
					Pre-insp			Est Post			Rework							
Mod/Mac Name	New or Mod	Full or Part Insp.	Programmer	Tester	A	M	D	A	M	D	A	M	D	Prep	Insp Meetg	Re-work	Follow-up	Sub-component
	N		McGINLEY	HALE	348			400			50			9.0	8.8	8.0	1.5	
				Totals														

Reinspection required? YES Length of inspection (clock hours and tenths) 2.2
Reinspection by (date) 11/25/– Additional modules/macros? NO
DCR #'s written C-2
Problem summary: Major 13 Minor 5 Total 18
Errors in changed code: Major Minor Errors in base code: Major Minor

LARSON		McGINLEY		HALE	
Initial Desr	Detailed Dr	Programmer	Team Leader	Other	Moderator's S

Design and code inspections to reduce errors in program development,
M. E. Fagan, 1976

Code reviews are an effective method for improving software quality. Unit testing finds approximately 25% of defects, function testing 35%, integration testing 45%, and code review 55-60%.
("Code clean", McConnell, 2004)



library/SolverUtils/Filters/FilterCheckpoint.cpp

```
+ //boost::ignore_unused(time);
```

Mohsen Lahooti @mlahooti · 4 Oct 2022, 23:52
It seems that the boost::ignore_unused is not needed anymore Could you please delete the commented line?

Mohsen Lahooti @mlahooti started a thread on the diff 4 Oct 2022, 23:54

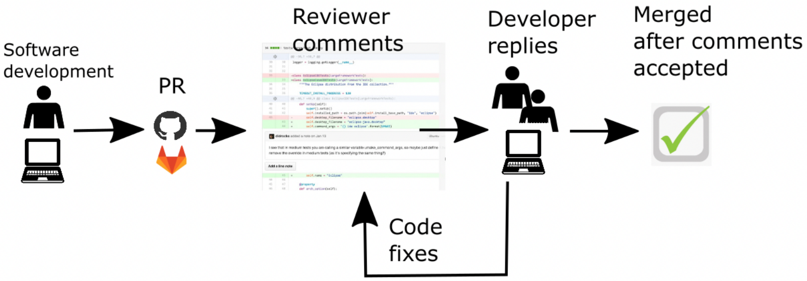
library/SolverUtils/Filters/FilterCheckpoint.cpp

```
      const Array<OneD, const MultiRegions::ExpListSharedPtr> &pFields,
      const NekDouble &time)
  {
-     boost::ignore_unused(time);
+   //boost::ignore_unused(time);

-     if (m_index++ % m_outputFrequency > 0)
+     if (m_index++ % m_outputFrequency > 0 || (time < m_startTime))
```

Mohsen Lahooti @mlahooti · 4 Oct 2022, 23:54
I would make this line consistent. Either use parentheses for the first condition or remove it from the second condition please

Asynchronous Code Review



```
@@ -30,7 +30,7 @@
 logger = logging.getLogger(__name__)


-class EclipseIDETests(LargeFrameworkTests):
+class EclipseJavaIDETests(LargeFrameworkTests):
     """The Eclipse distribution from the IDE collection."""

     TIMEOUT_INSTALL_PROGRESS = 120
@@ -40,7 +40,9 @@ class EclipseIDETests(LargeFrameworkTests):
     def setUp(self):
         super().setUp()
         self.installed_path = os.path.join(self.install_base_path, "ide", "eclipse")
-        self.desktop_filename = "eclipse.desktop"
+        self.desktop_filename = "eclipse-java.desktop"
+        self.command_args = '{} ide eclipse'.format(UMAKE)
```

didrocks added a note on Jan 13

I see that in medium tests you are calling a similar variable umake_command_args, so maybe just define [illegible] remove the override in medium tests (as it's specifying the same thing?)

```
+        self.name = "Eclipse"

     @property
     def arch_option(self):
```

Synchronous Code Review

Elise Özalp, Yaxin, Defne Ozan, Daniel Kelshaw (https://magrilab.ae.ic.ac.uk), Thibault Lestang. Photo cred: Neil Montague.

Department of Aeronautics, Imperial College London

Not a peer review for code

- Code review **throughout the research process**:
  - Frequent
  - Informal
  - Low stakes
- Commonly referred as "Modern Code Review" in the SE literature. Bachelli and Bird 2013
- Can be *asynchronous* (GitHub's Pull Requests) or *synchronous* (in person chat).

CODECHECK

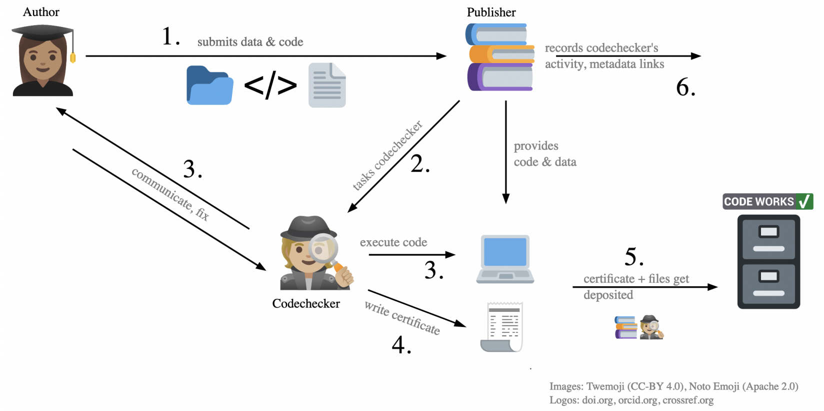

Author
1. submits data & code
Publisher
records codechecker's activity, metadata links
6.
tasks codechecker
2.
provides code & data
3.
communicate, fix
CODE WORKS ✓
execute code
3.
Codechecker
5.
certificate + files get deposited
write certificate
4.
Images: Twemoji (CC-BY 4.0), Noto Emoji (Apache 2.0)
Logos: doi.org, orcid.org, crossref.org

Figure 2: codecheck.org.uk

Two contexts

1. Individual developers writing their own specific software.
2. Developers collaboration on a common codebase.
   - Code review as gatekeeping.

Research on code reviews

Modern Code Review: A Case Study at Google (Sadowski, 2018)

Expectations, Outcomes, and Challenges of Modern Code Review (Bacchelli and Bird, 2013)

Code Reviewing in the Trenches: Understanding Challenges and Best Practices (McLeod et al, 2017)

Code review by and for scientists (Petre & Wilson, 2014)

Part 1

Benefits of code reviews for research software

Code review for software quality

1. Defects
2. Code improvements

Code review for software quality

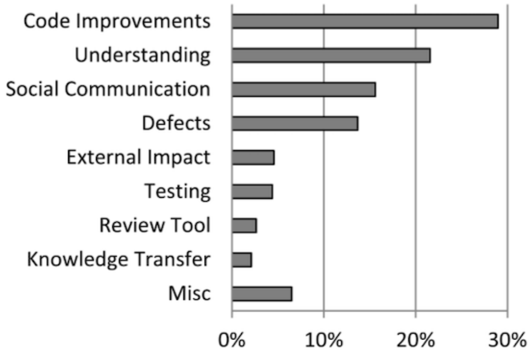


Figure 4. Frequency of comments by card sort category.

Figure 3: (Bachelli & Bird, 13)

Code reviews for understandability

More often than not source code is the only available form of documentation.

Understandability is key for **code reuse** and **transparency**.

Code reviews for team awareness

- Continuous knowledge exchange.
- Enhanced collaboration.
- Longer term resilience of project(s) (Bus factor!).

Code reviews for team awareness

Elise Özalp, Yaxin, Defne Ozan, Daniel Kelshaw (https://magrilab.ae.ic.ac.uk), Thibault Lestang. Photo cred: Neil Montague.

Department of Aeronautics, Imperial College London

Code reviews for knowledge transfer

Code review is peer learning.

- Spread of good practices.
- Homogeneisation of styles and practices

Code reviews for knowledge transfer

Code review is peer learning.

- Spread of good practices.
- Homogeneisation of styles and practices

```
filepath = "/my/own/specific/path/" + "data.csv"
```

Code reviews for knowledge transfer

Code review is peer learning.

- Spread of good practices.
- Homogeneisation of styles and practices

```
filepath = "/my/own/specific/path/" + "data.csv"
```

```
from pathlib import Path
# ...
datadir_path = Path("/my/own/specific/path/")
filepath = datadir_path / "data.csv"
```

Part 2: Challenges

A lot of good practices around...

...but what about **research software**?

Code review is time and energy

Two complementary courses of actions:

- Regularly reflect process and follow good practices.

Code review is time and energy

Two complementary courses of actions:

- Regularly reflect process and follow good practices.
- Acknowledge code review as a worthy investment:

Code review is time and energy

Two complementary courses of actions:

- Regularly reflect process and follow good practices.
- Acknowledge code review as a worthy investment:
  - “middle-term” benefits for individuals.

Code review is time and energy

Two complementary courses of actions:

- Regularly reflect process and follow good practices.
- Acknowledge code review as a worthy investment:
    - “middle-term” benefits for individuals.
    - Short and long term benefits for collectives.

Code review is time and energy

Two complementary courses of actions:

- Regularly reflect process and follow good practices.
- Acknowledge code review as a worthy investment:
    - “middle-term” benefits for individuals.
    - Short and long term benefits for collectives.

Code review is time and energy

Two complementary courses of actions:

- Regularly reflect process and follow good practices.
- Acknowledge code review as a worthy investment:
  - “middle-term” benefits for individuals.
  - Short and long term benefits for collectives.

Large return on investment

Being protective about code

1. There can be some unhealthy competition going on.
2. A large number of researchers feel shy about their coding practices:

- Lack of training.
- Other priorities, often structural (e.g. funding).
- Why would I share my code if nobody else does?

Being protective about code

1. There can be some unhealthy competition going on.
2. A large number of researchers feel shy about their coding practices:

- Lack of training.
- Other priorities, often structural (e.g. funding).
- Why would I share my code if nobody else does?

Code review can put software (back?) at the heart of the collaborative scientific process.

Strong heterogeneity among team members

- Experience.
- Skills (*e.g.* programming languages).
- Interest & motivation.

Other challenges

- Finding reviewers
- Finding guidance or mentors

Part 3: Code review good practices

A lot of the good practices from software engineering industry are applicable, **with a pinch of salt**.

Keep it short

3 times 30' instead of one time 90'

- Fit in a busy schedule.
- Doesn't feel like a big commitment.
- Code review can be a very demanding activity.

Remember that software isn't the primary driver.

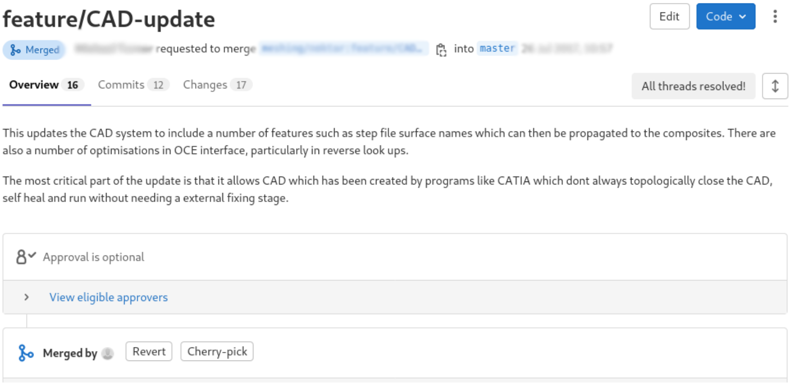
Avoid comfort mode

That doesn't look quite right but I guess that's okay...

I just must have missed something

In code review meetings, authors should make is easy for reviewers to interject.

The author's part

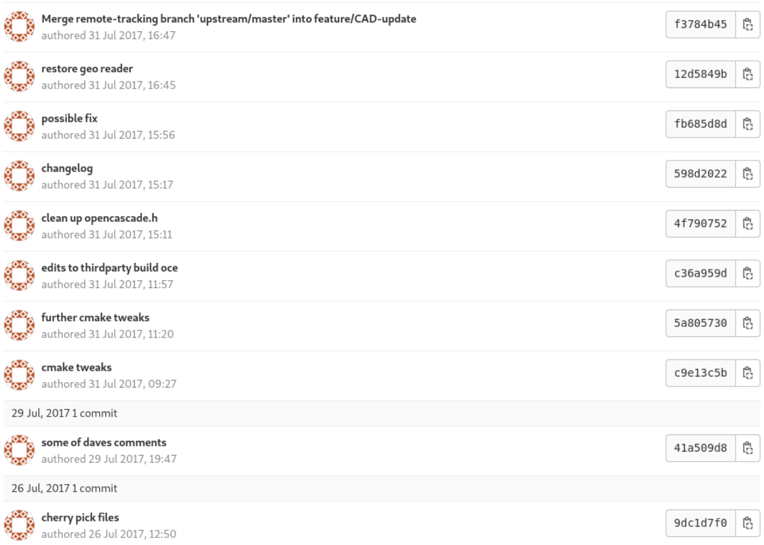

feature/CAD-update

This updates the CAD system to include a number of features such as step file surface names which can then be propagated to the composites. There are also a number of optimisations in OCE interface, particularly in reverse look ups.

The most critical part of the update is that it allows CAD which has been created by programs like CATIA which dont always topologically close the CAD, self heal and run without needing a external fixing stage.

Figure 4: A very scarce description

The author's part



Merge remote-tracking branch 'upstream/master' into feature/CAD-update
authored 31 Jul 2017, 16:47
f3784b45

restore geo reader
authored 31 Jul 2017, 16:45
12d5849b

possible fix
authored 31 Jul 2017, 15:56
fb685d8d

changelog
authored 31 Jul 2017, 15:17
598d2022

clean up opencascade.h
authored 31 Jul 2017, 15:11
4f790752

edits to thirdparty build oce
authored 31 Jul 2017, 11:57
c36a959d

further cmake tweaks
authored 31 Jul 2017, 11:20
5a805730

cmake tweaks
authored 31 Jul 2017, 09:27
c9e13c5b

29 Jul, 2017 1 commit

some of daves comments
authored 29 Jul 2017, 19:47
41a509d8

26 Jul, 2017 1 commit

cherry pick files
authored 26 Jul 2017, 12:50
9dc1d7f0

Figure 5: A very scarce description

The author's part

- Keep it small! (~30')
- Provide a description of the purpose and structure of the code.
- Think ahead what reviewers will and will not be familiar with
    - Specific libraries?
    - Specific domain knowledge?
- Ensure minimum quality standard (*e.g.* style, naming)

Put yourself into your reviewer(s)' shoes: what would you want to be told if asked to review your code?

Specify the feedback you are after

I'm not happy with this loop

```
for i in `seq 1 $NUMOFFIG`
do
  FIG=$(ls $IMDIR | head -n $i | tail -n 1)
  echo "    ${placeholderpath}/${FIG}" >> $FILE
done
```

Specify the feedback you are after

I'm not happy with this loop

```
for i in `seq 1 $NUMOFFIG`
do
  FIG=$(ls $IMDIR | head -n $i | tail -n 1)
  echo "    ${placeholderpath}/${FIG}" >> $FILE
done
```

I'm having to define a lot of classes that don't do much, what do you think of my design?

Specify the feedback you are after

I'm not happy with this loop

```
for i in `seq 1 $NUMOFFIG`
do
  FIG=$(ls $IMDIR | head -n $i | tail -n 1)
  echo "    ${placeholderpath}/${FIG}" >> $FILE
done
```

I'm having to define a lot of classes that don't do much, what do you think of my design?

I don't have any specific issue in mind, but I'm curious to see whether or not you find it hard to to follow the code's logic.

Define (and enforce) a scope

Example default scope: understandability

- Obscure variable names.
- Complex conditionals.
- Duplicated code.
- Long parameter lists.
- Shallow modules.
- ~~Standard compliance.~~
- ~~Performance sinks~~.
- ~~Security concerns~~.

Default scope can be overrriden at will.

Whether “it works” or not is irrelevant

- Code review is not an evaluation of a finished product.
- It is more rewarding to look at code that is WIP.
- The only expectation is that code is readable by reviewers.

Make it formal – but safe

Code review is more effective with a clear process (formal)

At the same time, Code review meetings *must* remain inclusives and supporting spaces.

It's about creating an environment where people feel confident about discussing their code to each other.

Overheard in the next meeting room

Author: *This loop I wrote looks too complicated to me.*

Overheard in the next meeting room

Author: *This loop I wrote looks too complicated to me.*

Reviewer: *Hmmm yes. You could just use a pipe and `xargs`.*

Overheard in the next meeting room

Author: *This loop I wrote looks too complicated to me.*

Reviewer: *Hmmm yes. You could just use a pipe and `xargs`.*

Author: *What's `xargs`?*

Overheard in the next meeting room

Author: *This loop I wrote looks too complicated to me.*

Reviewer: *Hmmm yes. You could just use a pipe and `xargs`.*

Author: *What's `xargs`?*

Reviewer: *It's basically mapping a command over a set of inputs - think functional programming!*

Overheard in the next meeting room

Author: *This loop I wrote looks too complicated to me.*

Reviewer: *Hmmm yes. You could just use a pipe and* `xargs`*.*

Author: *What's* `xargs`*?*

Reviewer: *It's basically mapping a command over a set of inputs - think functional programming!*

Author: *. . .*

Overheard in the next meeting room

Author: *This loop I wrote looks too complicated to me.*

Reviewer: *Hmmm yes. You could just use a pipe and `xargs`.*

Author: *What's `xargs`?*

Reviewer: *It's basically mapping a command over a set of inputs - think functional programming!*

Author: *. . .*

Reviewer: *Alhtough you could also do the same thing with `sed`.*

Overheard in the next meeting room

Author: *This loop I wrote looks too complicated to me.*

Reviewer: *Hmmm yes. You could just use a pipe and `xargs`.*

Author: *What's `xargs`?*

Reviewer: *It's basically mapping a command over a set of inputs - think functional programming!*

Author: *. . .*

Reviewer: *Alhtough you could also do the same thing with `sed`.*

Author (looking frustrated): *I have no idea what you're talking about.*

All feedback isn't helpful

. . . at least for now.

Reviewers with more programming experience/enthusiasm must be careful not to overwhelm beginners.

Use a checklist

- ☐ Poor formatting.
- ☐ Dead code.
- ☐ Missing documentation.
- ☐ Obscure names.
- ☐ Complex conditionals.
- ☐ Obscure one-liners.
- ☐ Duplicated code.
- ☐ Long procedures.
- ☐ Long parameter lists.
- ☐ Global state.
- ☐ Abuse of primitive types.
- ☐ Data clumps.
- ...

Critique the code, not the programmer

You clearly made little effort in naming things. . .

You should name this differently

I think this name is misleading

Giving feedback is not trivial

1. Own you opinions.
2. Make it about the code.
3. Be specific.
4. Suggest an alternative.

Giving feedback is not trivial

1. Own you opinions.
2. Make it about the code.
3. Be specific.
4. Suggest an alternative.

I think this function's purpose would be much clearer if it was given a more explicit name.. perhaps `apply_bwd_transform`*?*

Code review is both **technical** and **social**

Code reviews can drive both inclusion and exclusion.

Code review is both **technical** and **social**

Code reviews can drive both inclusion and exclusion.

> *A bad reviewer tries to force their preference on you. A good code reviewer makes your code confrom to certain principles, but not opinion. (Quote from survey participant from Greiler, 2016)*

Define (and refine) a policy

- Well defined process.
- Default scope.
- Moderator(s).
- Code of conduct.
- Conflict resolution.

A culture of openess and collaboration

- Components of a successful software project are
  - Code
  - People
  - Communication
- Research code review goes along with collective ownersip of research project.