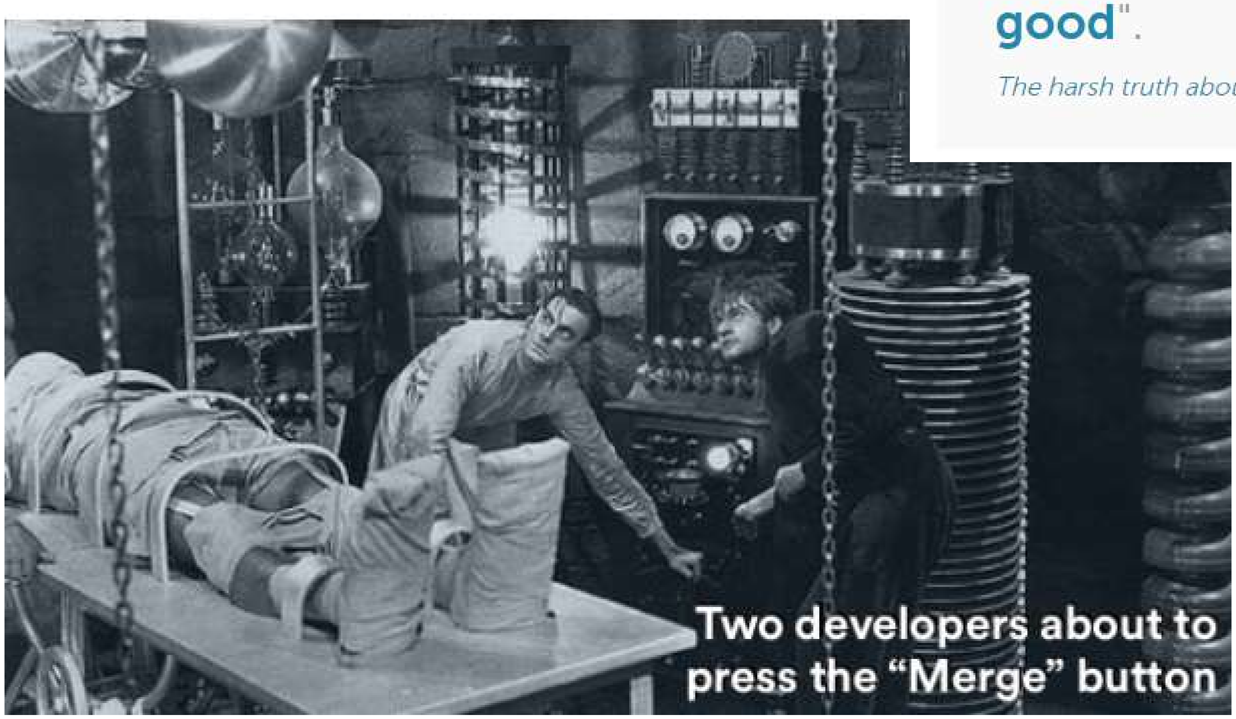
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
ROUEN NORMANDIE

CODE REVIEW

Where we speak of code approval

"Ask programmers to review **10 lines of code**, they'll find **10 issues**.
Ask them to do **500 lines** and they'll say **it looks good**".

The harsh truth about code reviews.



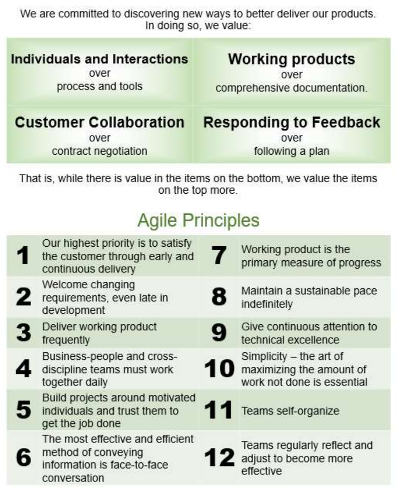
Source : https://fr.slideshare.net/GoAtlassian/code-reviews-vs-pull-requests-75670325

Code Review

How code review aligns with :

- **Agile manifesto**
  - Value : Individuals and interactions over processes and tools
    - *collective code ownership* (or *shared code*) : the code base is owned by the entire team (no notion of individual ownership of modules)

Source : https://agilemanifesto.org/



We are committed to discovering new ways to better deliver our products.
In doing so, we value:

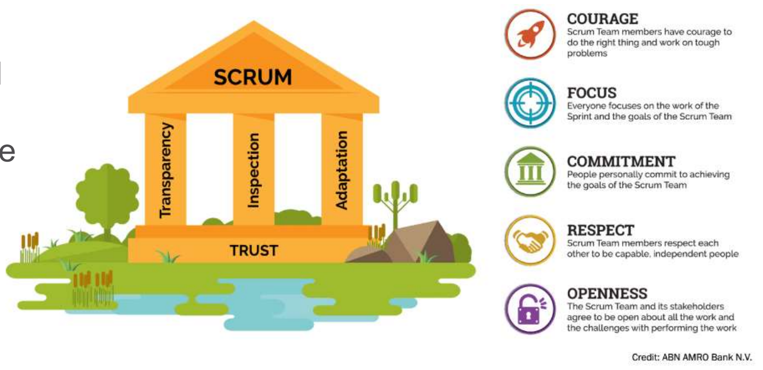
Individuals and Interactions over process and tools	**Working products** over comprehensive documentation.
Customer Collaboration over contract negotiation	**Responding to Feedback** over following a plan

That is, while there is value in the items on the bottom, we value the items on the top more.

Agile Principles

1	Our highest priority is to satisfy the customer through early and continuous delivery	7	Working product is the primary measure of progress
2	Welcome changing requirements, even late in development	8	Maintain a sustainable pace indefinitely
3	Deliver working product frequently	9	Give continuous attention to technical excellence
4	Business-people and cross-discipline teams must work together daily	10	Simplicity – the art of maximizing the amount of work not done is essential
5	Build projects around motivated individuals and trust them to get the job done	11	Teams self-organize
6	The most effective and efficient method of conveying information is face-to-face conversation	12	Teams regularly reflect and adjust to become more effective

- **SCRUM guide**
  - Scrum pillars : Inspection
    - The Scrum artifacts and the progress toward agreed goals must be inspected frequently and diligently to detect potentially undesirable variances or problems.



COURAGE Scrum Team members have courage to do the right thing and work on tough problems

FOCUS Everyone focuses on the work of the Sprint and the goals of the Scrum Team

COMMITMENT People personally commit to achieving the goals of the Scrum Team

RESPECT Scrum Team members respect each other to be capable, independent people

OPENNESS The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work

Credit: ABN AMRO Bank N.V.

Source : https://www.scrum.org/resources/what-is-scrum

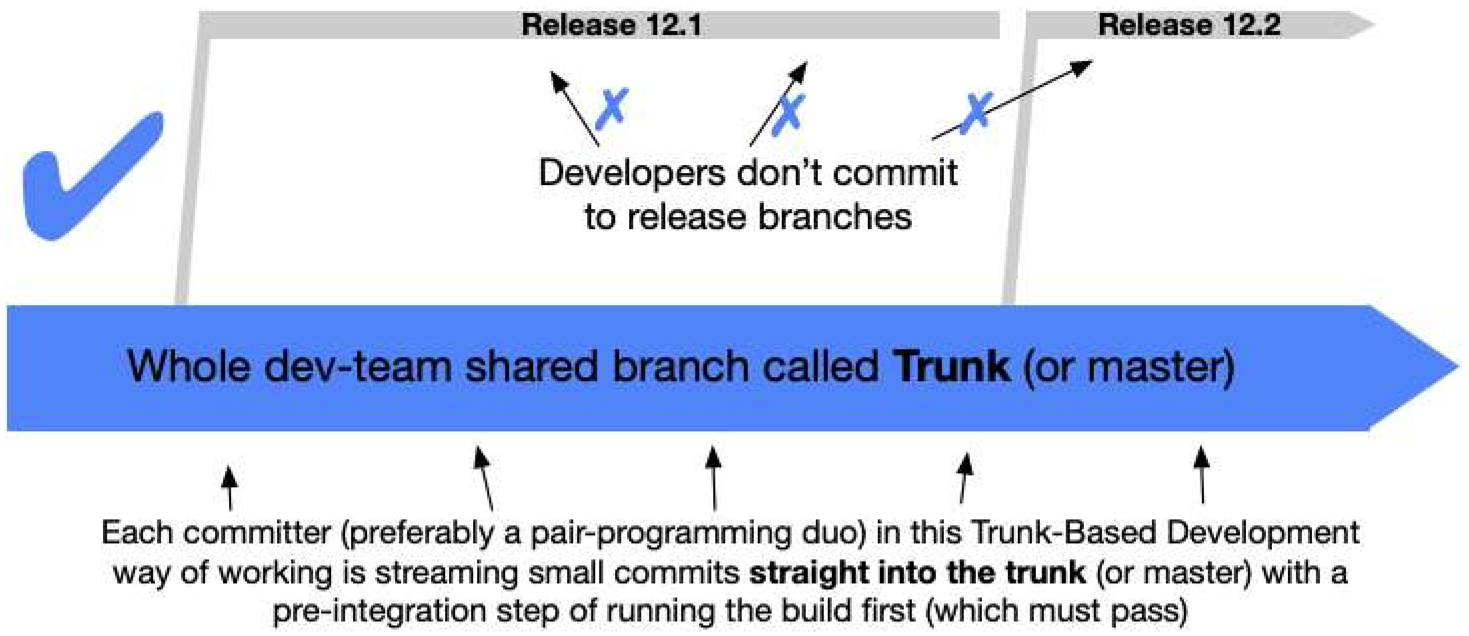
Code review – a common definition

Two acceptions of *Code Review* term

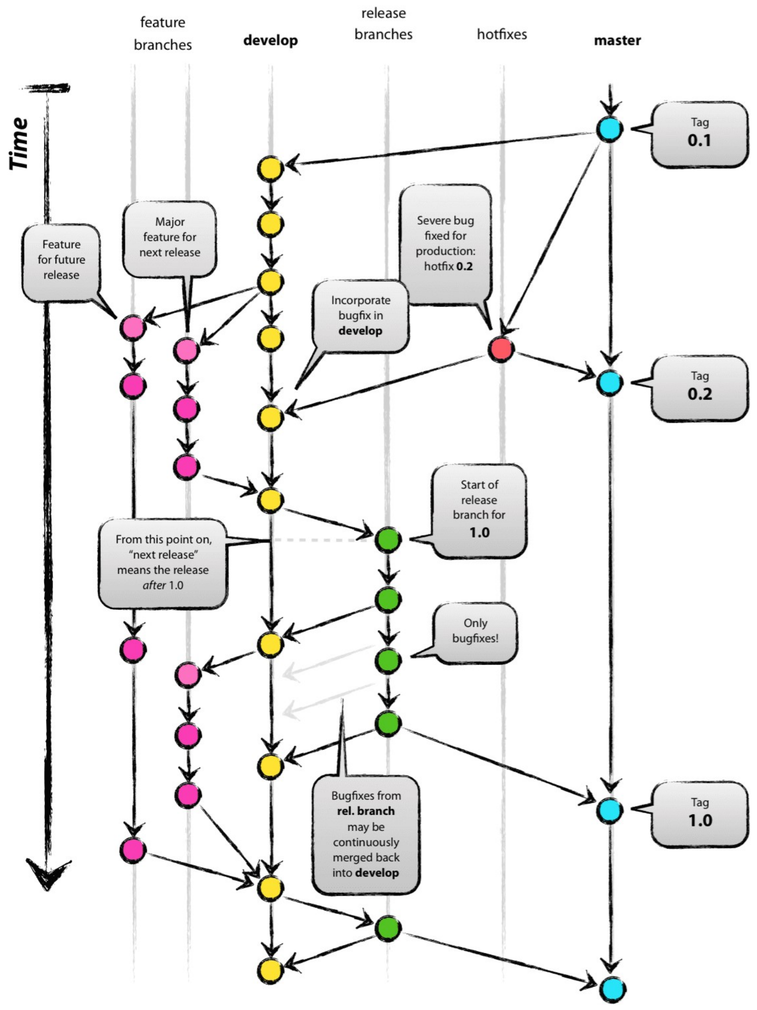
- **Code approval**
    - Domain : branching strategy
    - Intention : get approval of an *authority* to merge newly developped code
    - Implementation : PR (MR) requests
- **Peer review**
    - Domain : pair programming, mob programming, …
    - Intention : get feedback from others to improve skills, share the best pratices
    - Implementation : out of scope

Here the term *code approval* will be preferred !

Branches strategy : Trunk-Based development

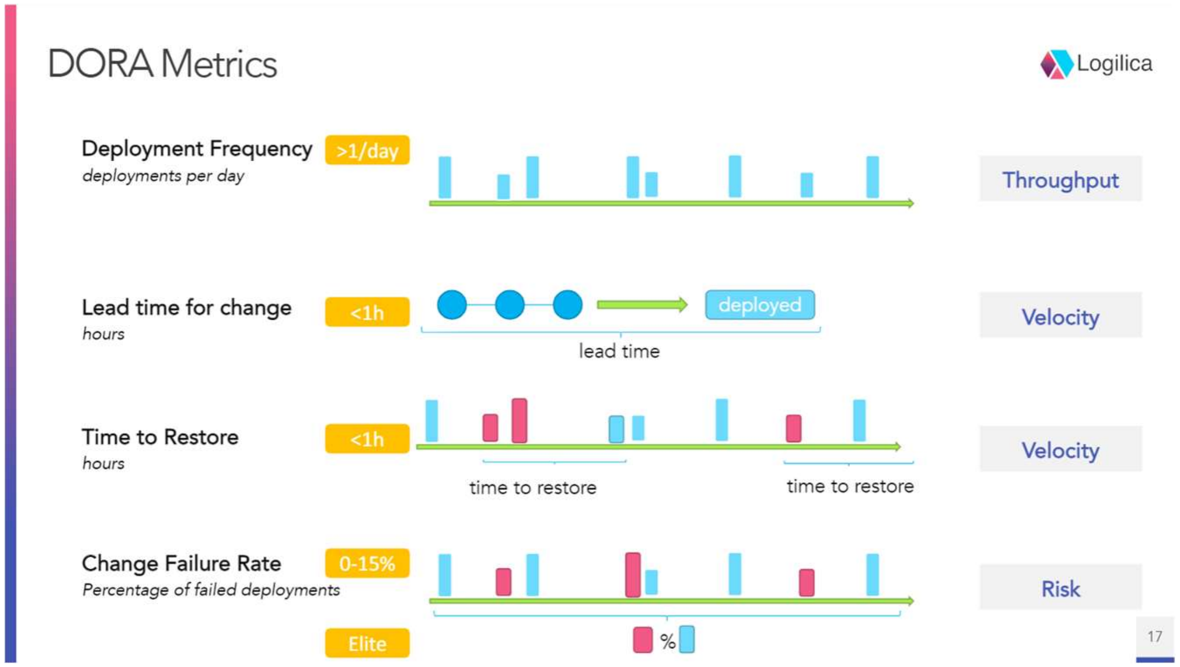

Release 12.1
Release 12.2
Developers don't commit to release branches
Whole dev-team shared branch called **Trunk** (or master)
Each committer (preferably a pair-programming duo) in this Trunk-Based Development way of working is streaming small commits **straight into the trunk** (or master) with a pre-integration step of running the build first (which must pass)

Branches strategy : GitFlow


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Source : https://nvie.com/posts/a-successful-git-branching-model/

DORA metrics



Code approval must not impair DORA's « Lead time for change »

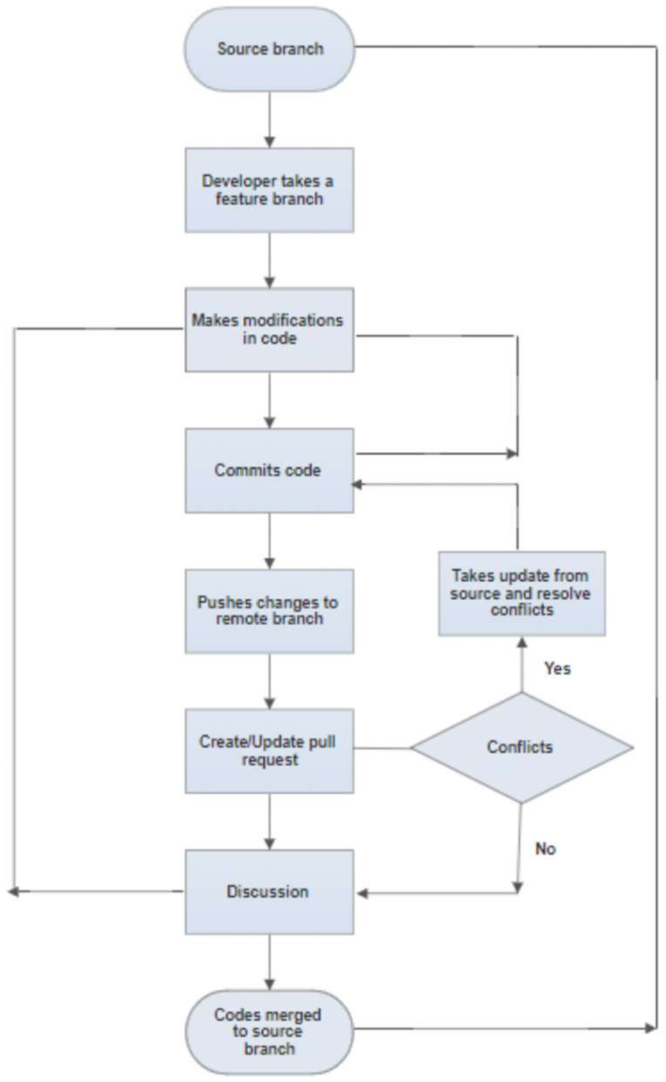
A code must not be approved when

- **Use case not covered**
- **Security matter**

All other reason should not prevent approval, namely

- **Maintenability**

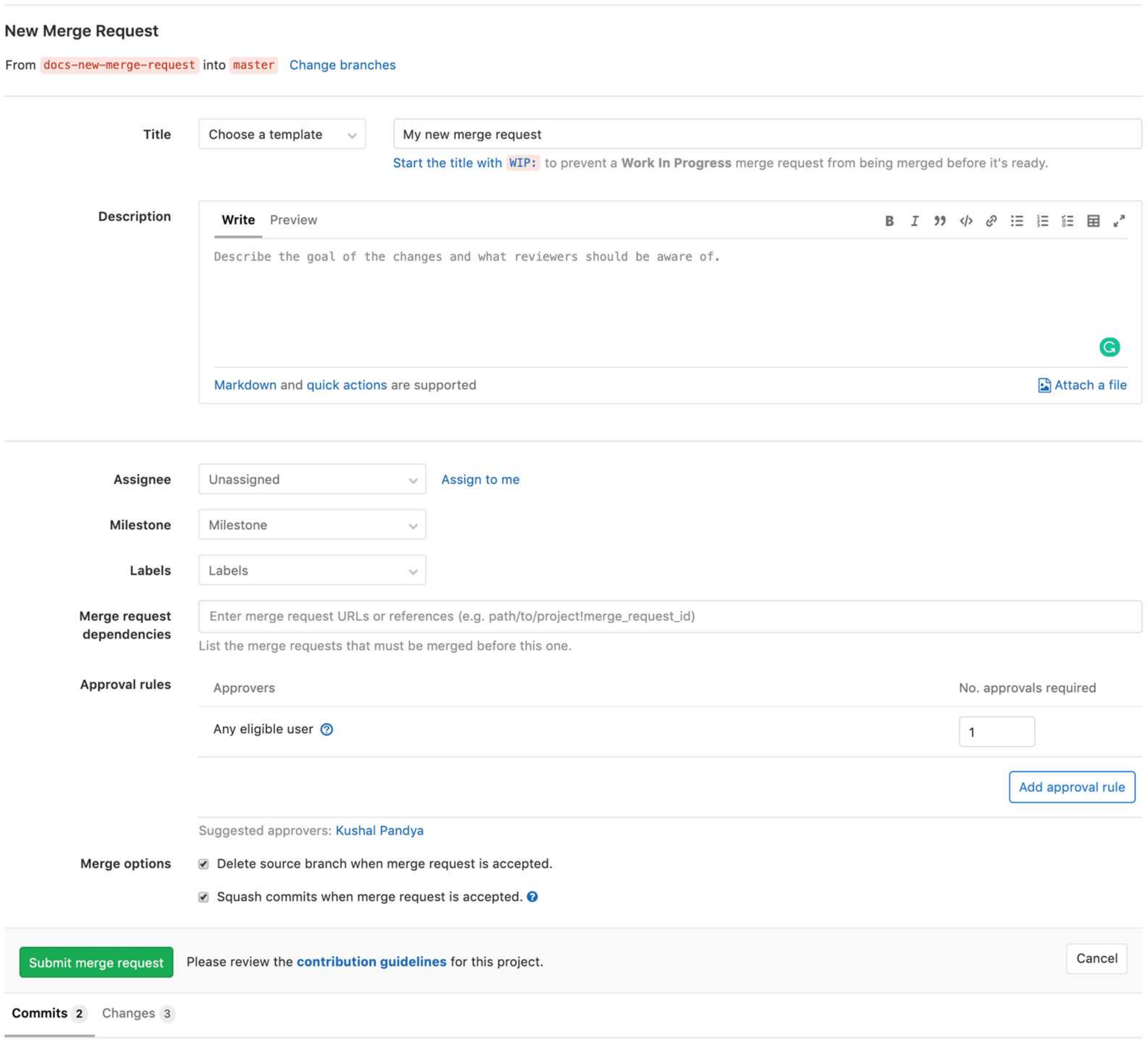
List of main steps of a MR or PR workflow


Source branch
Developer takes a feature branch
Makes modifications in code
Commits code
Pushes changes to remote branch
Takes update from source and resolve conflicts
Yes
Create/Update pull request
Conflicts
No
Discussion
Codes merged to source branch

Normandie Université

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

[Gitlab] Merge Request screen copy



New Merge Request

From `docs-new-merge-request` into `master` Change branches

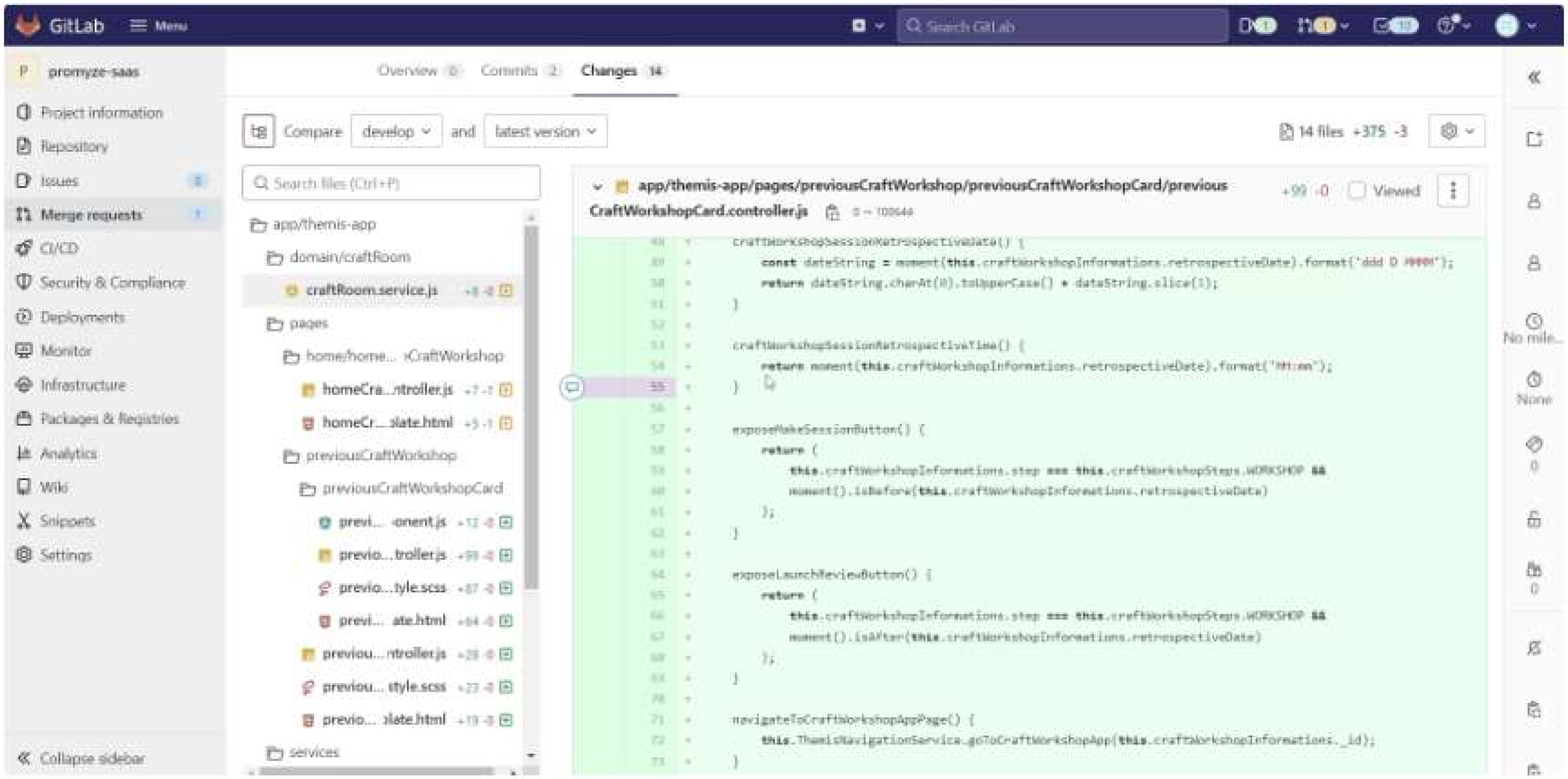
Title	Choose a template / My new merge request
	Start the title with `WIP:` to prevent a **Work In Progress** merge request from being merged before it's ready.
Description	Write / Preview
	Describe the goal of the changes and what reviewers should be aware of.
	Markdown and quick actions are supported — Attach a file
Assignee	Unassigned — Assign to me
Milestone	Milestone
Labels	Labels
Merge request dependencies	Enter merge request URLs or references (e.g. path/to/project!merge_request_id)
	List the merge requests that must be merged before this one.
Approval rules	Approvers — No. approvals required
	Any eligible user — 1
	Add approval rule
	Suggested approvers: Kushal Pandya
Merge options	[✓] Delete source branch when merge request is accepted.
	[✓] Squash commits when merge request is accepted.

Submit merge request — Please review the **contribution guidelines** for this project. — Cancel

Commits 2 Changes 3

Normandie Université

[Gitlab] Code review for Code Approval

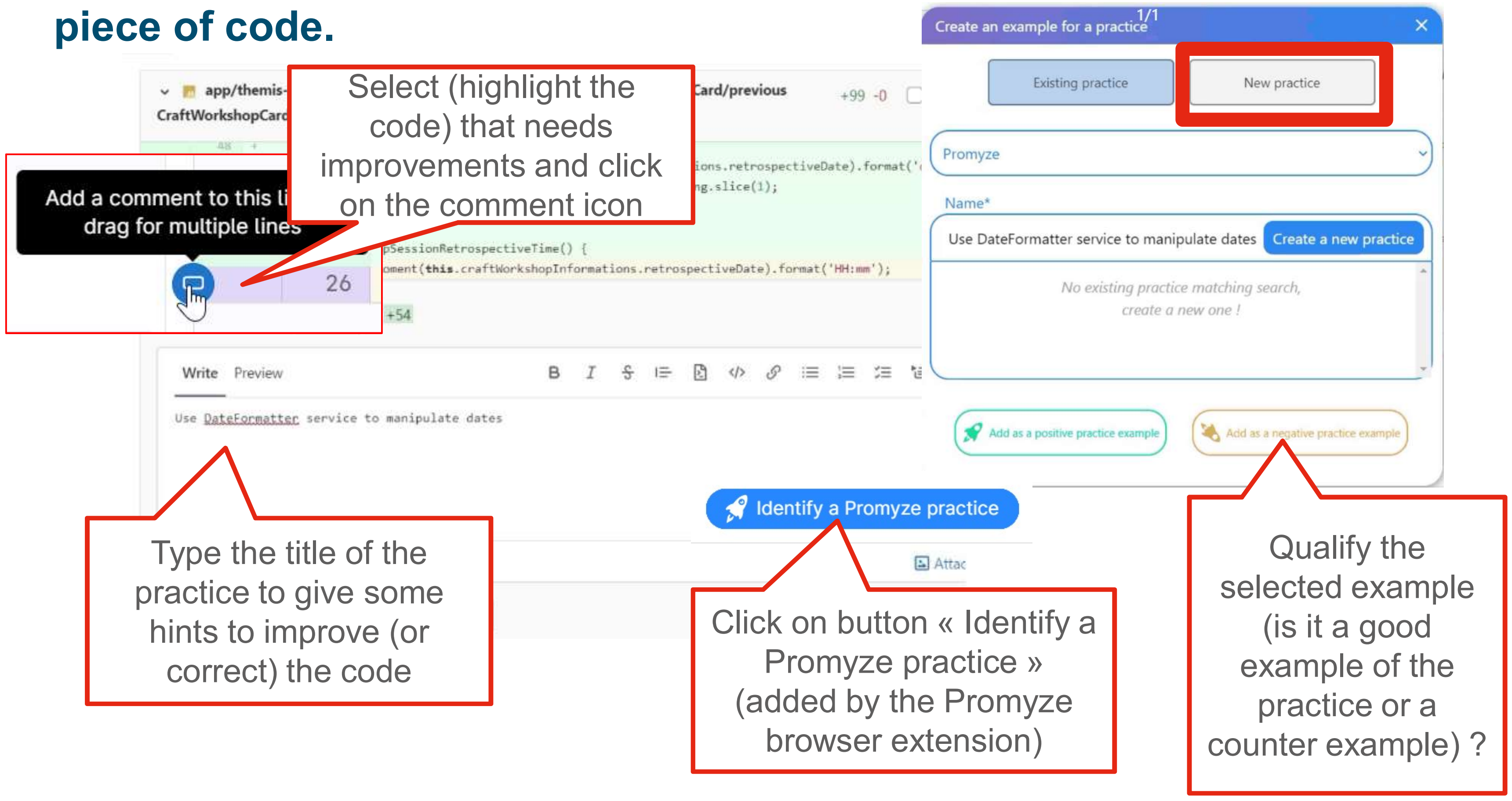


app/themis-app/pages/previousCraftWorkshop/previousCraftWorkshopCard/previousCraftWorkshopCard.controller.js 0 → 100644

```
    craftWorkshopSessionRetrospectiveDate() {
        const dateString = moment(this.craftWorkshopInformations.retrospectiveDate).format('ddd D MMMM');
        return dateString.charAt(0).toUpperCase() + dateString.slice(1);
    }

    craftWorkshopSessionRetrospectiveTime() {
        return moment(this.craftWorkshopInformations.retrospectiveDate).format('HH:mm');
    }

    exposeMakeSessionButton() {
        return (
            this.craftWorkshopInformations.step === this.craftWorkshopSteps.WORKSHOP &&
            moment().isBefore(this.craftWorkshopInformations.retrospectiveDate)
        );
    }

    exposeLaunchReviewButton() {
        return (
            this.craftWorkshopInformations.step === this.craftWorkshopSteps.WORKSHOP &&
            moment().isAfter(this.craftWorkshopInformations.retrospectiveDate)
        );
    }

    navigateToCraftWorkshopAppPage() {
        this.ThemisNavigationService.goToCraftWorkshopApp(this.craftWorkshopInformations._id);
    }
```

[Promyze] Reuse of a piece of knowledge (practice)

While code reviewed, you can put in the <u>knowledge base</u> any negative or positive application of a <u>practice</u> by simply hightlighting it the related piece of code.



app/themis-… CraftWorkshopCar… …Card/previous +99 -0

```
…ions.retrospectiveDate).format('…
…ng.slice(1);
…pSessionRetrospectiveTime() {
…oment(this.craftWorkshopInformations.retrospectiveDate).format('HH:mm');
```

Add a comment to this li… drag for multiple lines
26
+54
Write
Preview
Identify a Promyze practice
Attac

```
Use DateFormatter service to manipulate dates
```

Create an example for a practice
1/1
Existing practice
New practice
Promyze
Name*

Use DateFormatter service to manipulate dates

No existing practice matching search, create a new one !

Add as a positive practice example
Add as a negative practice example

Select (highlight the code) that needs improvements and click on the comment icon

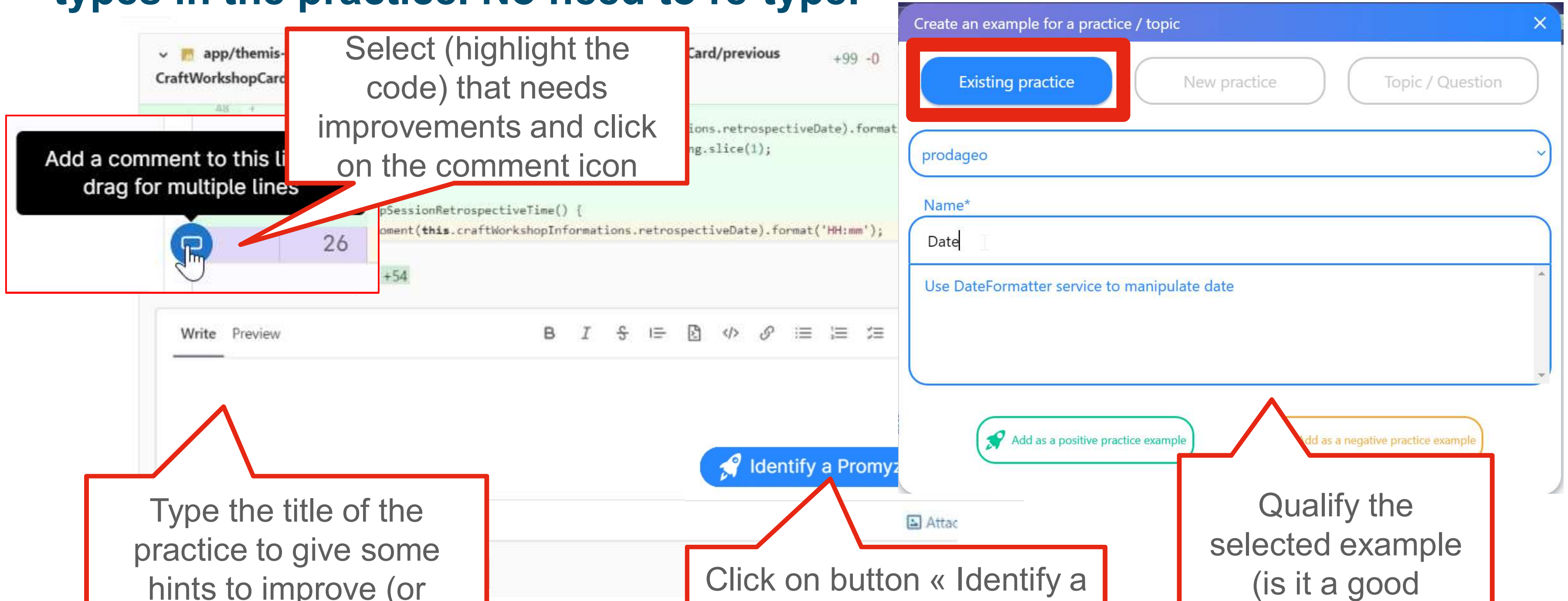
Type the title of the practice to give some hints to improve (or correct) the code

Click on button « Identify a Promyze practice » (added by the Promyze browser extension)

Qualify the selected example (is it a good example of the practice or a counter example) ?

[Promyze] Reuse of a piece of knowledge (practice)

Next code review you see a counter-example of the practice, just add a reference to the pratice and the developper will get all the knowledge types in the practice. No need to re-type.



```
ions.retrospectiveDate).format
ng.slice(1);
pSessionRetrospectiveTime() {
oment(this.craftWorkshopInformations.retrospectiveDate).format('HH:mm');
```

Create an example for a practice / topic

prodageo

Name*: Date

Use DateFormatter service to manipulate date

Select (highlight the code) that needs improvements and click on the comment icon

Type the title of the practice to give some hints to improve (or correct) the code

Click on button « Identify a Promyze practice » (added by the Promyze browser extension)

Qualify the selected example (is it a good example of the practice or a counter example) ?

Numerous other tools are available to assist and optimize code approval

- SmartBear Collaborator : customize code approval workflows to your requirements
- GitColony : avoid titanic reviews before deploying