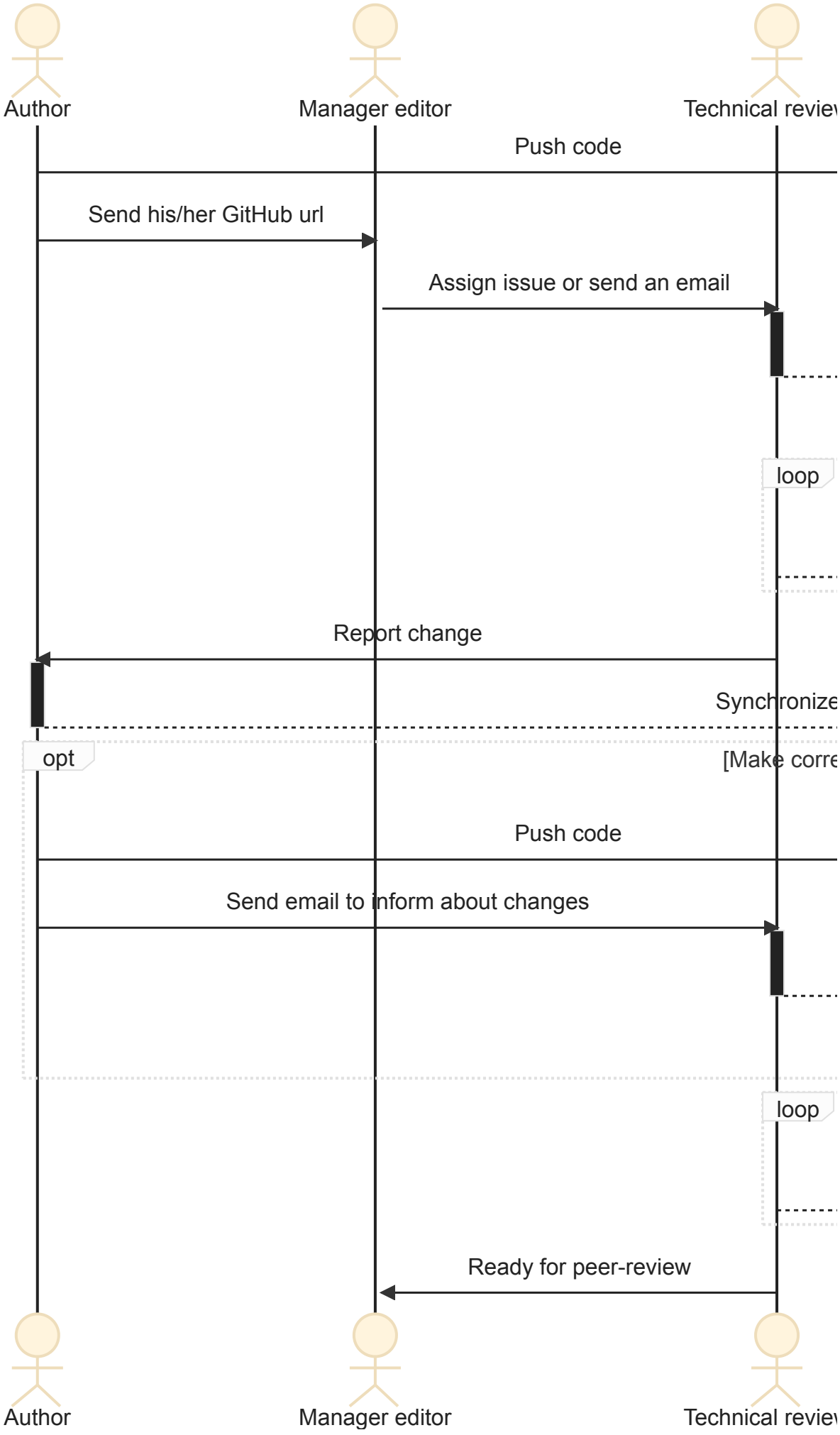
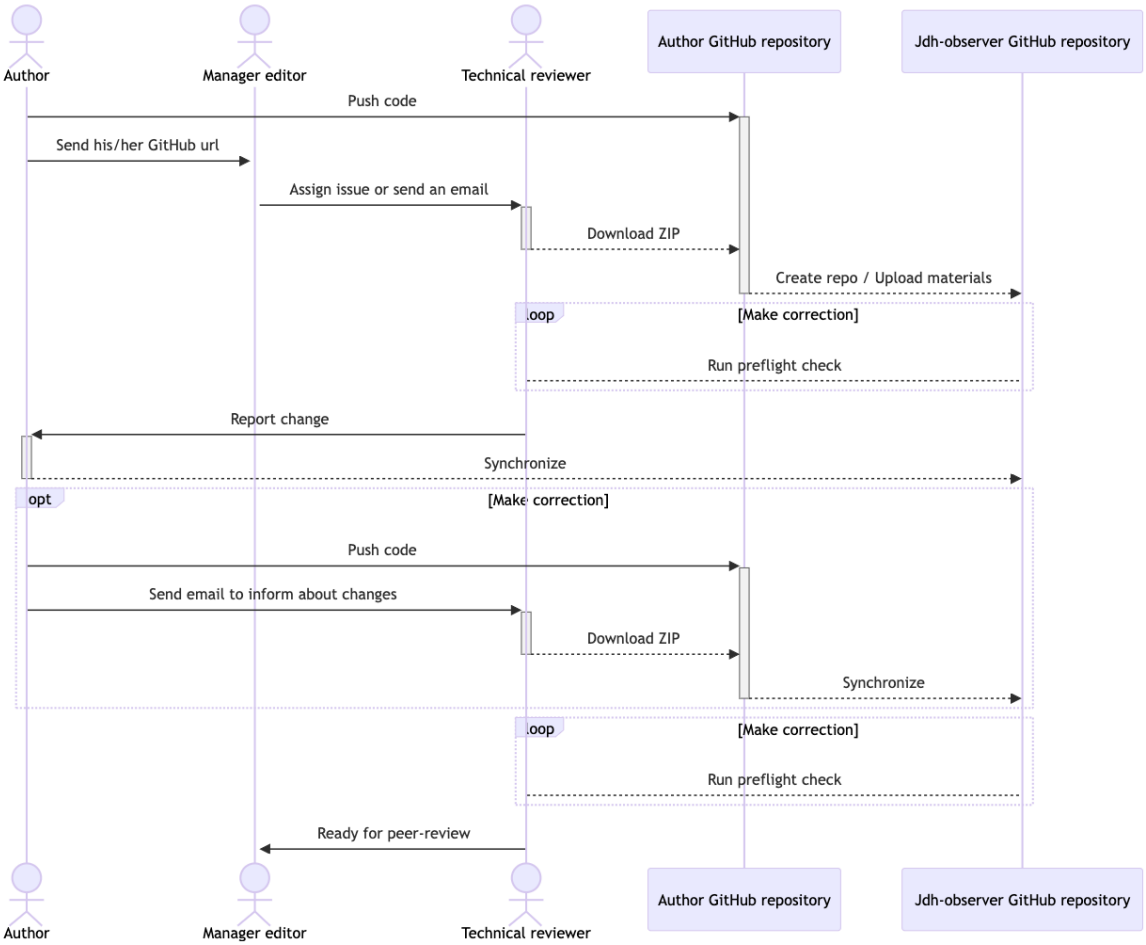
Workflow single blind

Technical peer-review

Current workflow : technical peer-review


Author
Manager editor
Technical reviev
Push code
Send his/her GitHub url
Assign issue or send an email
loop
Report change
Synchronize
opt
[Make corre
Push code
Send email to inform about changes
loop
Ready for peer-review
Author
Manager editor
Technical reviev



- Difficulty of the synchronization of the Jdh-observer GitHub 's repository with the Author Github's repository

  > as they are not fork repository

- Before submission of the article no mandatory check
  - the use of the preview in the JDH is an advice
  - the GitHub's action is now in the template but define as workflow dispatch (manually trigger)
- Email communication , difficulty sometimes to retrieve the information
- Long delay in order to perform the first technical review as it's involved a lot of manual step at the creation of the repo / upload materials , at some points we have 15 notebooks in the same time

---

Inspiration

Many journals use now GitHub

Template

- In order to provide a template:
  - Example of the JOVI journal https://github.com/journalovi/jovi-template-quarto
  - Example of the rzine journal, package for R studio https://gitlab.huma-num.fr/rzine/package

---

Workflow

- In order to manage their workflow, it means using issues and pull request
  - Example of the JOVI journal https://www.journalovi.org/submit.html#experimental

> All review on the experimental track will proceed as Github issues and pull requests, and published papers on this track can be updated using pull requests, even after publication. The Github repository for each submission will be public throughout the process, regardless of the final decision on the submission. Therefore, while withdrawal of a submission is possible (in the sense that authors can ask for their paper not to be considered "published" at JoVI), the submitted paper and reviews will still be public even if the submission is withdrawn or is not accepted.

5 Code
6 Equations
References

1 Setting up and writing a JoVI article

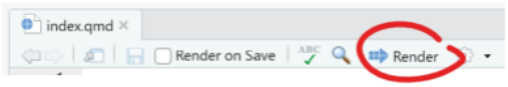
Articles on the JoVI experimental track are written in Quarto, which is a simple markdown-based text format. This template outlines some of the features that are most useful for writing academic articles in Quarto; for more visit https://quarto.org.

You can find the source for this article in the journalovi/jovi-template-quarto repository on Github. To create a new JoVI article, we recommend following these steps:

1. Install Quarto by following these instructions.
2. Create a new git repository to hold your article by forking our template repository.

You can then edit `index.qmd` in your new repository to write your paper. There are several options for editing and rendering the paper:

1. You can execute `quarto serve` from the commandline to render the paper to `index.html` and preview it in the browser.
2. You can edit the paper in RStudio and render it by clicking on the *Render* button:



If you plan to use RStudio, read more about RStudio and Quarto here. In particular, you should install RStudio > version 1.5. If you do, you can also enable the visual editor:

```
- Example of the Programming Historians
https://programminghistorian.org/en/author-guidelines#step-3-submitting-a-new-lesson

![[Screenshot 2023-10-31 at 10.48.11.png]]
```

The peer review process normally happens in 3 stages:

1) The editor assigned to your lesson will carefully read and try your lesson, providing a first round of feedback that you will be asked to respond to. The purpose of this first round of feedback is to ensure that your lesson addresses the needs of *Programming Historian* readers, and to make sure that the external peer reviewers receive a lesson that works. You will normally be given one month to respond to this first peer review.

2) The editor will then open the lesson for formal peer review. This will include at least two reviewers invited by the editor, and may also include comments from the wider community, who are welcome to contribute views. We generally try to ask reviewers to provide their comments within one month, but sometimes unforeseen circumstances mean this is not possible. The editor should make it clear to you that you should not respond to reviews until after both reviews have been published and the editor has summarised and provided clear instructions for moving forward. In some cases this may be a suggestion to substantially revise or rethink the lesson. In other cases it will be a matter of making some changes. Depending on the peer review comments, and the nature of issues raised, you may need to revise the tutorial more than once, but the editor will endeavour to ensure that you are given a clear pathway towards publication. You always have the option of withdrawing from the review process if you so choose.

3) Once your editor and peer reviewers are happy with the piece, the editor will recommend publication to the Managing Editor, who will read the piece to ensure that it meets our Author's Guidelines and standards. In some cases there may be additional revisions or copy editing at this stage to bring the piece in line with our publishing standards. If the Managing Editor is happy with the piece, it will be moved to the live site for publication. Your editor will inform you of any additional information required at this stage.

Interesting points:

- delay provide (one month to respond to the first review)
- incremental way : my need to revise the tutorial more than once
    - your lesson addresses the needs of the Programming historians

> On this point i will like to encourage a bit discussion with the manager editorial team , feeling alone about at the technical review for issues not related to technical, for example: definition of the hermeneutics paragraph, definition of the new development related to an article

---

Missing points

- No validation is provided when the article is delivered.

Necessity to automatise this feedback in order to provide also quick technical feedback

---

New workflow: technical peer-review

Receive the Author's GitHub repository url by form

Automatic:

- track in the DB the delivery date
- automatic answer at the reception of the form
- fork the repository

> Will facilitate the synchronisation later

look at here : https://docs.github.com/en/rest/repos/forks?apiVersion=2022-11-28

- run the preflight check

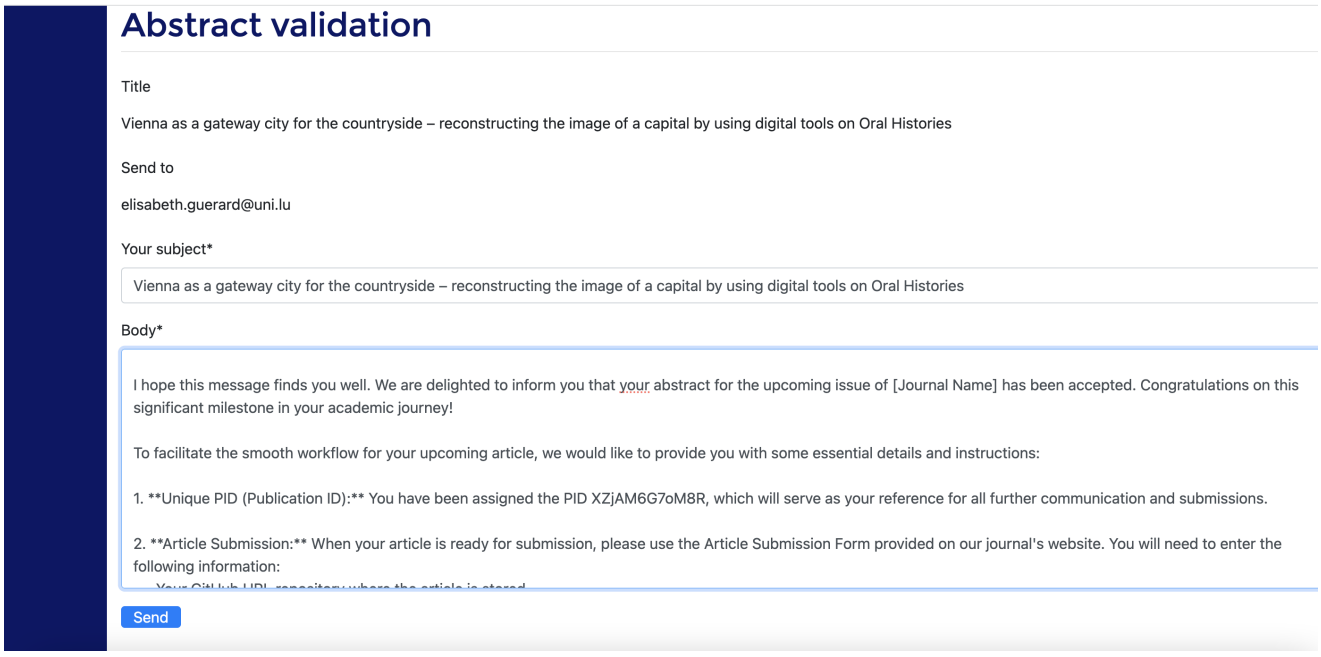
Human intervention:

- analyse of the technical reviewer
- notify the author about possible issue but using the GitHub pull request / issue / comment

---

At the abstract submission

- Give the author the pid in order to keep it for further communication

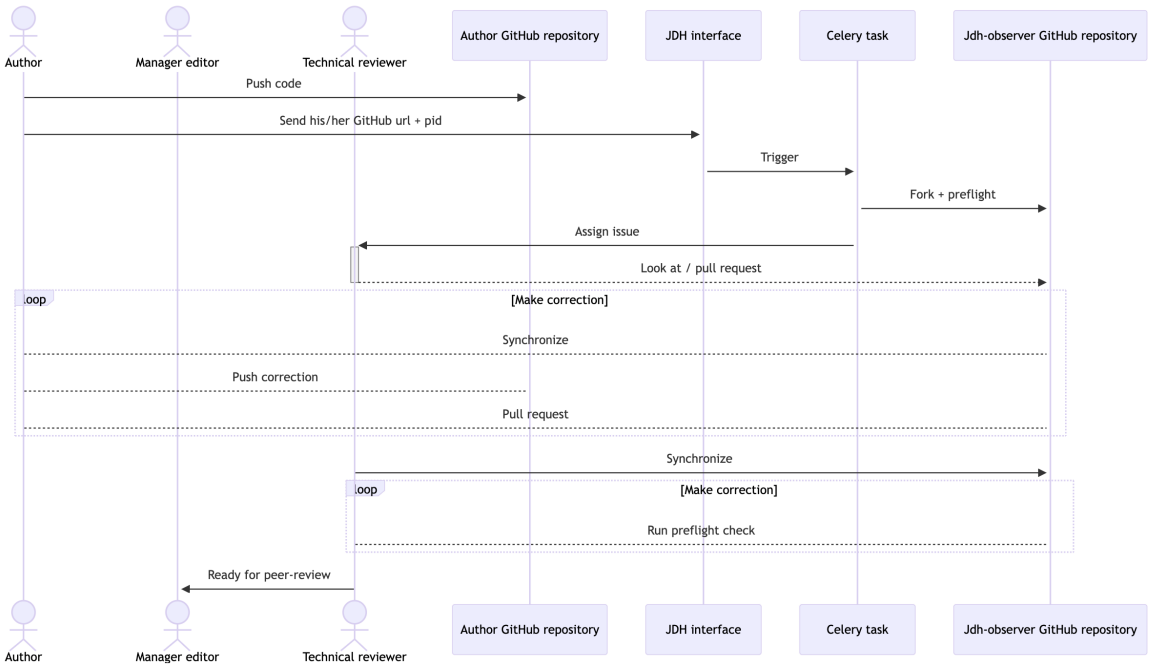


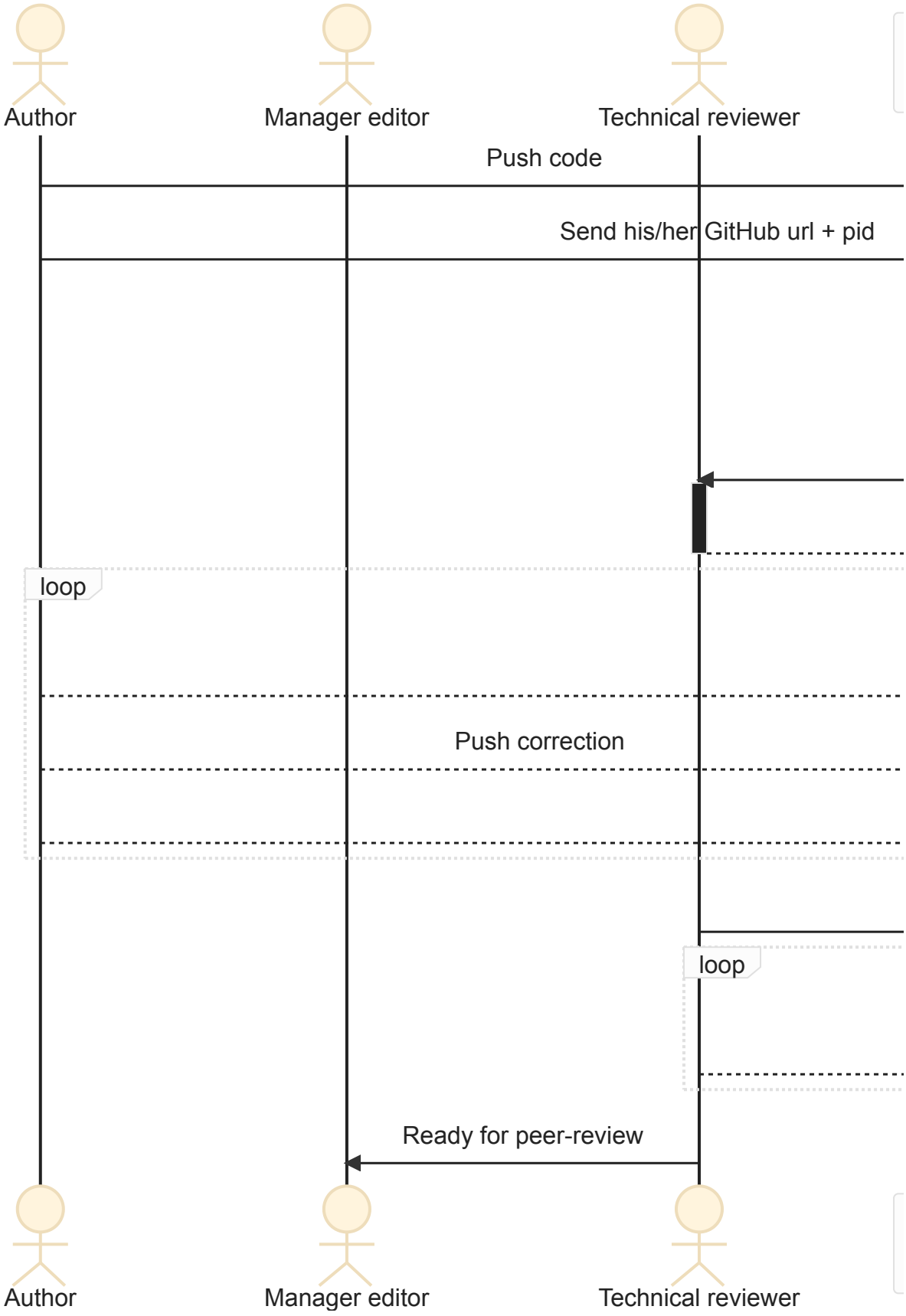
At the article submission

Send :

- article github url
- pid of the article in order to retrieve the abstract

Workflow



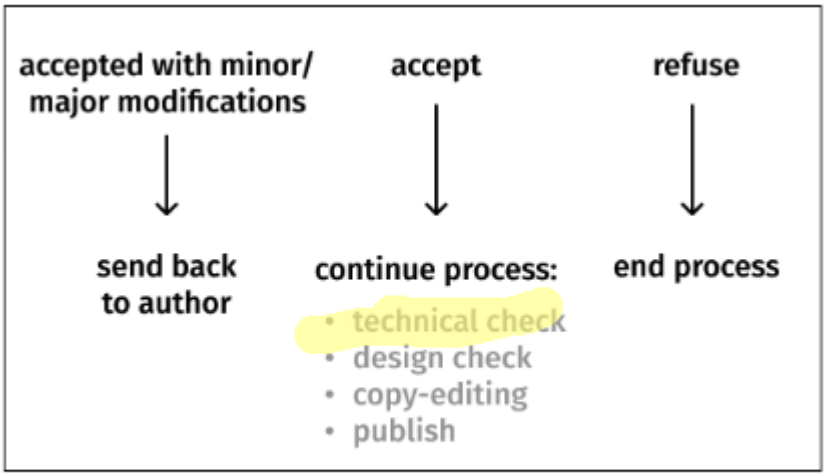


Peer-review

Current workflow double blind Peer-review

https://docs.google.com/document/d/1RmBFVMZYCcxQjyL6CkDEgQeThn6sDVNNtI26lV8PwYg/edit?pli=1

Issue type	Varia issue	Special issue	Open ended issue
editors	Managing editor, 1-2 junior editors	2-3 external editors, 1-2 junior editors	External editors, 1 junior editor
Call for paper	yes	yes	yes
Submission period	anytime	Specific period to submit article	Open ended
publishing	immediately	All at the same time	immediately
process	• receive abstract • first meeting with author (do JDH fit and show guideline) • validate abstract in backend • author writes article • technical check of preliminary notebook • anonymize article and add to repo (double-blind) • editor/author adds article to Scholar One • select & send to reviewer in Scholar One • accept with minor/major modifications, send back to author	• receive abstract • first meeting with ALL authors (seminar) • external editor validate abstract via mail and recommends reviewer(s) • author writes article • technical check of preliminary notebook • anonymize article and add to repo (double-blind) • author adds article to Scholar One • send to reviewer in Scholar One • accept with minor/major modifications, send back to author	• receive abstract • first meeting with author (do JDH fit and show guideline) • external editor validate abstract via mail and recommends reviewer(s) • author writes article • technical check of preliminary notebook • anonymize article and add to repo (double-blind) • author adds article to Scholar One • send to reviewer in Scholar One • accept with minor/major modifications, send back to author
Article status	Accepted with minor/major modifications -> send back to author Accepted -> continue process (technical check, design check, copy-editing, publish) Refuse -> end process		



- Send back to the author , currently we re-run the process of the technical check

  > Author needs to inform us about changes made - problem of synchronization again

- Accept
  - Manager editor set the status of the Journal at "Design review"
- Refuse: no case for the moment

Workflow single blind Peer-review

What needs to be change in ScholarOne?

- documentation
- default email when article is accepted with minor/major modifications

> will be good to have the same mention than at the abstract validation in order to the author to submit the form

Dashboard

- At the validation of the article

Backend

- Introduce DOI when article has been accepted
- Set to "Design review"