Berger Levrault

Mining software repository with Pharo

03 July 2025

Nicolas Hlad

with Benoit Verhaeghe & Kilian Bauvent

Introduction

1. What is Mining Software Repository ?
2. What are the specificities of Berger-Levrault regarding MSR ?
3. How we develop *GitProjectHealth*?
4. What can you do with it ? (demo)
5. What is next ?

What is
Mining Software Repository ?

(MSR)

Definitions - 2

- Today's collaborative development relies on **Git Social Platforms** (GSP) **[Dabbish et al. 2012]**
  - they are **server** implementations of **Git** with builtin **social features**
  - They contain valuable historical information over a software development

Fig - Discussing Pull Request in GitHub

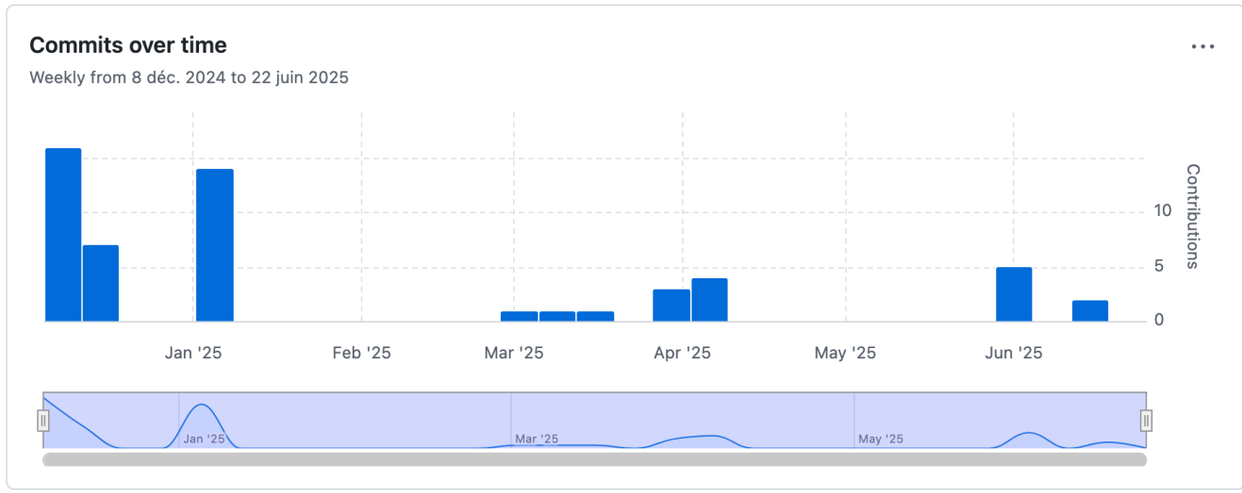


Fig - Commits distribution over time (GitHub)

Definitions - 2

- Mining Software Repository (MSR) provides methods and tools to extract data from these platforms **[Hassan 2008]**.
- Among other, it allows us to:
  - Studying the impact of code smells **[Steffen et al. 2010, Palomba et al. 2014]**
  - Exploring developers code review **[Bacchelli et al. 2013]**
  - Predicting classes prone to change and defect **[Bacchelli et al. 2010]**
  - Retro-analysing a entire development process **[Mockus et al. 2000]**

Existing tools

- **General Mining data**
  - PyDriller — python tool for mining commit
  - Git-Miner — Pharo tool, based on CLI-GitMiner
- **Specific Mining Data**
  - Javapers — java lib mostly for Java file analysis in git repository (leveraging SPOON)
  - ModelMine — retrieve UML model from project's artefacts
- **Data Storage**
  - Pandora —focus on long terme storage of Git social platform data
  - Software Heritage — Archive of Git repository from GSP
- **Computing Metrics**
  - LinearB — Productions and deployment metrics, data positioning with other companies

What are the specificities of Berger-Levrault regarding MSR

A quick word on Berger-levrault

- Berger-Levrault is
  - a group of international software editors
  - with divers sectors of activity (eduction, health, public administration, etc)
- The group acquired divers software editors over the past 30 years.
  - From different countries (France, Spain, Canada, Maroco, etc);
  - working with different technology (Java, C#, Typescript, Dart, etc);
  - and different Git Social Platform (Gitlab, Bitbucket, Azure Devops, etc).

A quick word on Berger-levrault - 2

- We use the *project management system* Altelissan's **Jira** to manage:
  - tickets (Bug, features, Hotfix, etc)
  - SPRINT (Agile development)
  - releases (delivering dates, testing software, etc).

Fig - Kanban view of a SPRINT in Jira

A quick word on Berger-levrault - 3

- Our Jira and Git Social Platform environment are connected

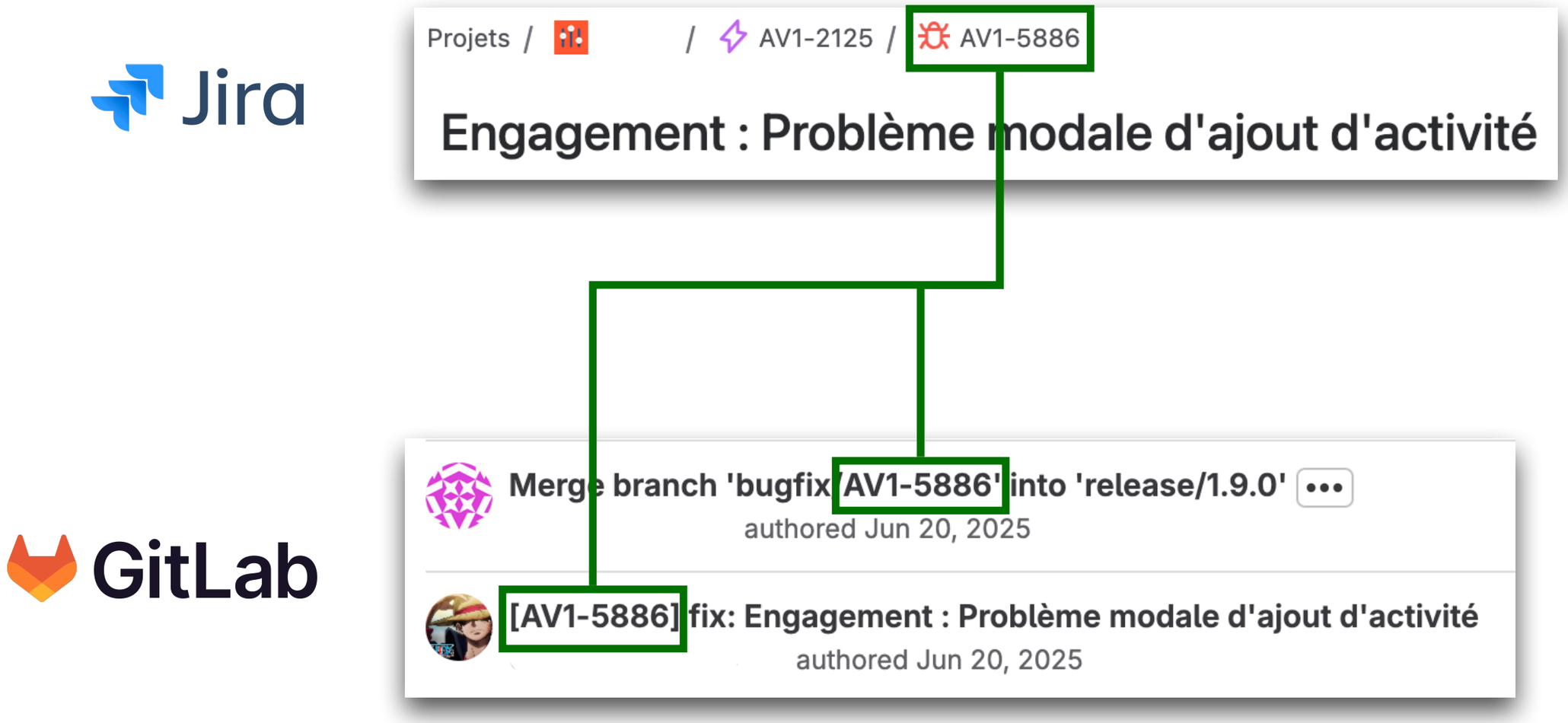


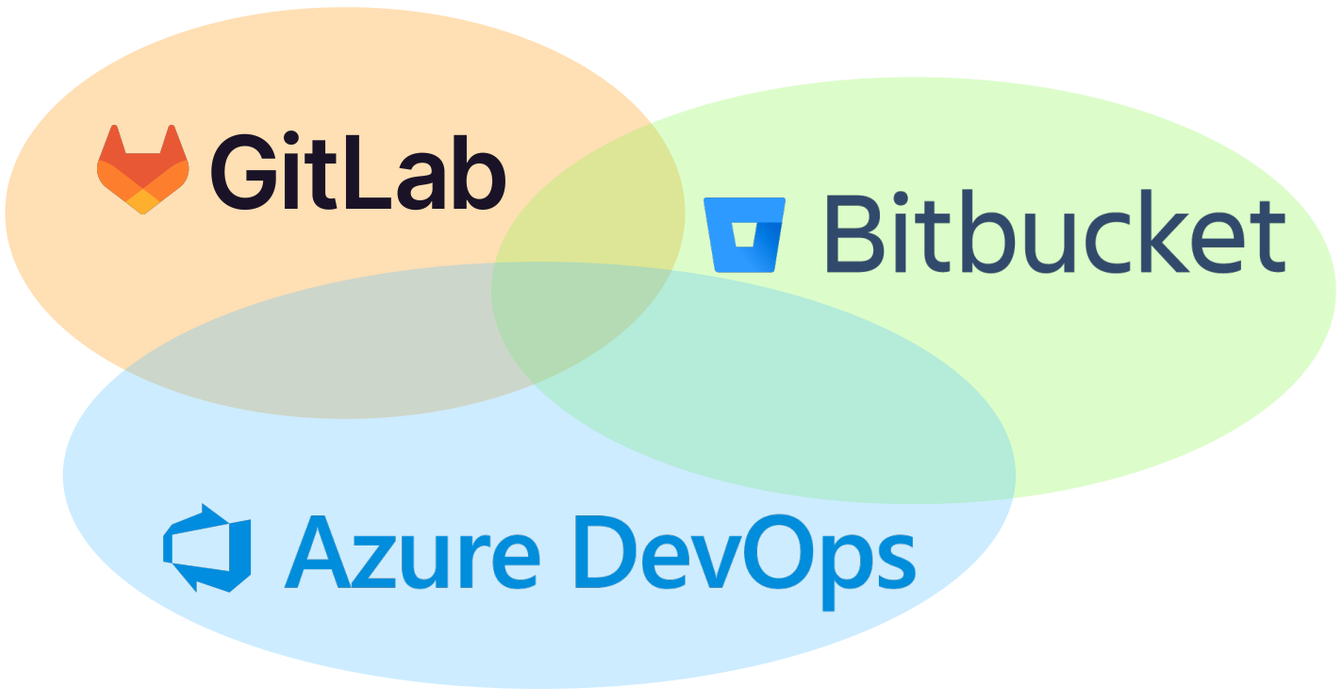
Fig - Linking Jira Tickets to Commit and Merge activity in GitLab

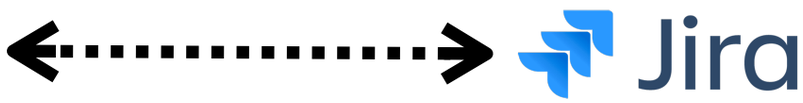
Industrial context

The specificities of Berger-Levrault

Git Social Platform

project management systems





How to mine from different Git Social Platforms (GSP) ?

How to implement MSR metrics efficiently ?

How to connect GSP data to Jira efficiently ?

We use Model Driven Engineering with Pharo-Moose

How we develop our solution with MDE

The conception of *Git Project Health*

Our MSR solution

GitProjectHealth

GitProjectHealth (GPH) is framework to extract and analyse data from Git Social platforms using Model-Driven Engineering (MDE).

tool for **General Mining data** & **Metrics computing**

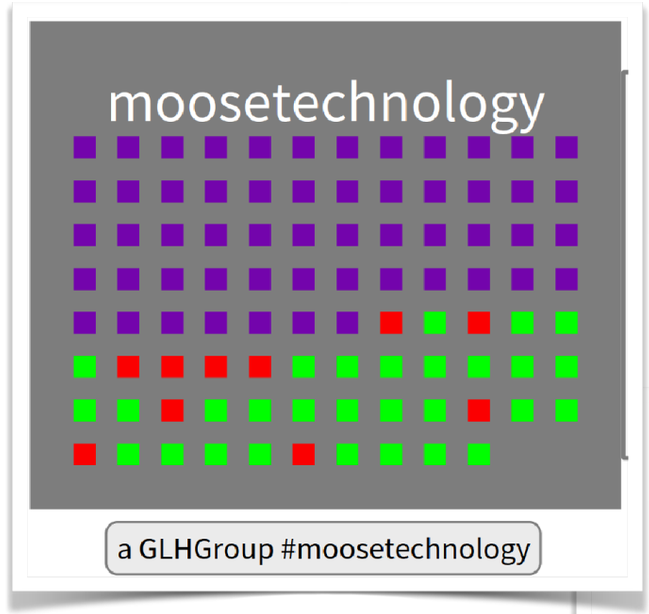
- GitProjectHealth contributions are :
  - A unify model for all Git Social Platform
  - A framework to build custom metric from the model
  - A use of *metamodel connector* to extend any analysis to other platforms (e.g., Jira)

github.com/moosetechnology/GitProjectHealth

Main feature

GitProjectHealth

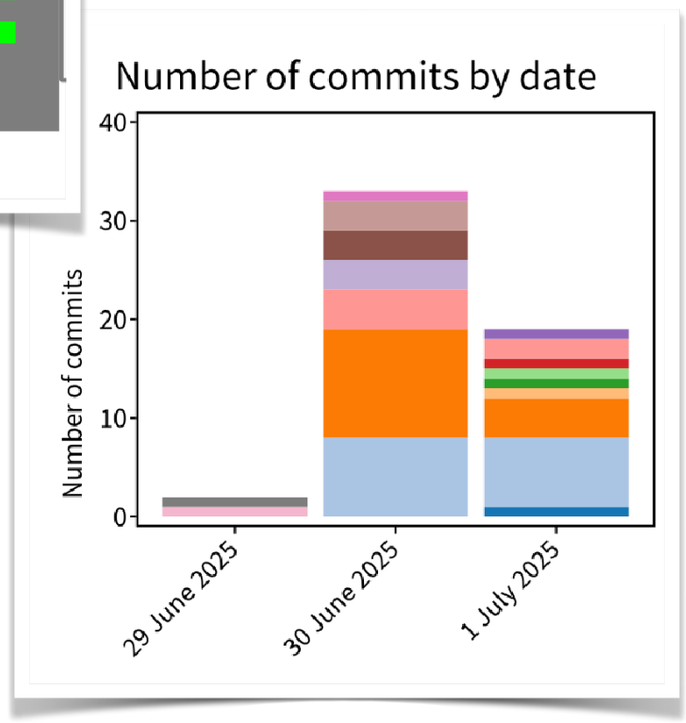
```
"INKSCAPE"
ink_commits := gitlabImporter importCommitsOfProject: inkscape
   since: ('30 June 2025' asDate)
   until: ('2 July 2025' asDate).
ink_commits do: [ :c | |user|
   c commitCreator: (gitlabImporter importUserByUsername: c author_name).
   ].
gitlabImporter chainsCommitsFrom: ink_commits.
```



Key Features:

- **Data Extraction & importation**:
  Extracts data from major social platforms.
  Imports a model of specific Git entities.
- **Visualization and Metrics**:
  Visualizes data and computes metrics.
- **Model Connection**:
  Connects models (e.g., Gitlab and Jira).

Project name	Project ID	closed merge request duration (second) - 22 June 2025	churn % in project (W=3) - 22 June 2025
Client	36189		19.49511162994309
inkscape	3472737	2505	1.5639652524938998



Git Model

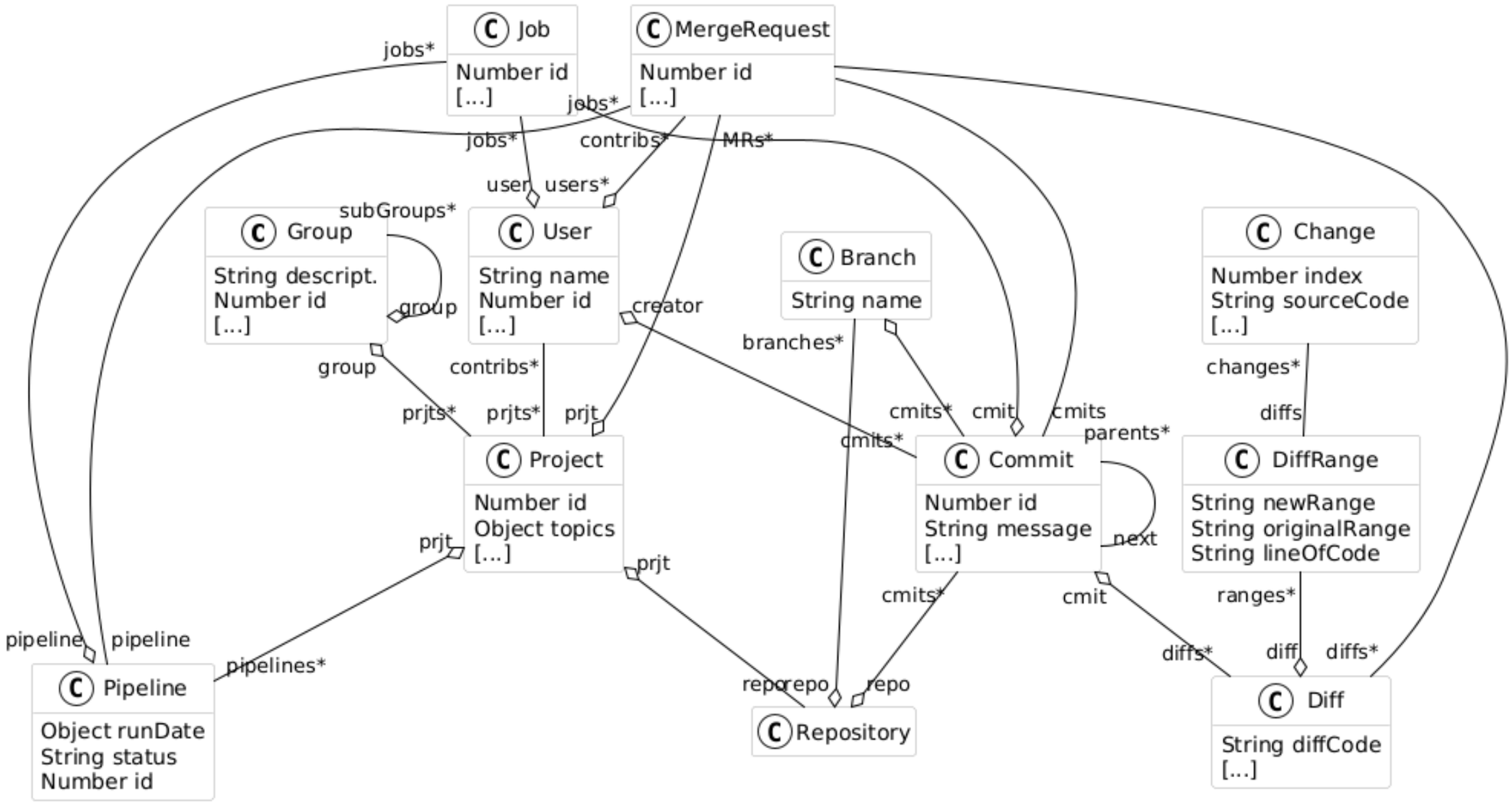


Fig - Simplify Metamodel of a Git Social Platform in GitProjectHealth

Git Model

1. Naming decisions

Merge Request vs Pull Request

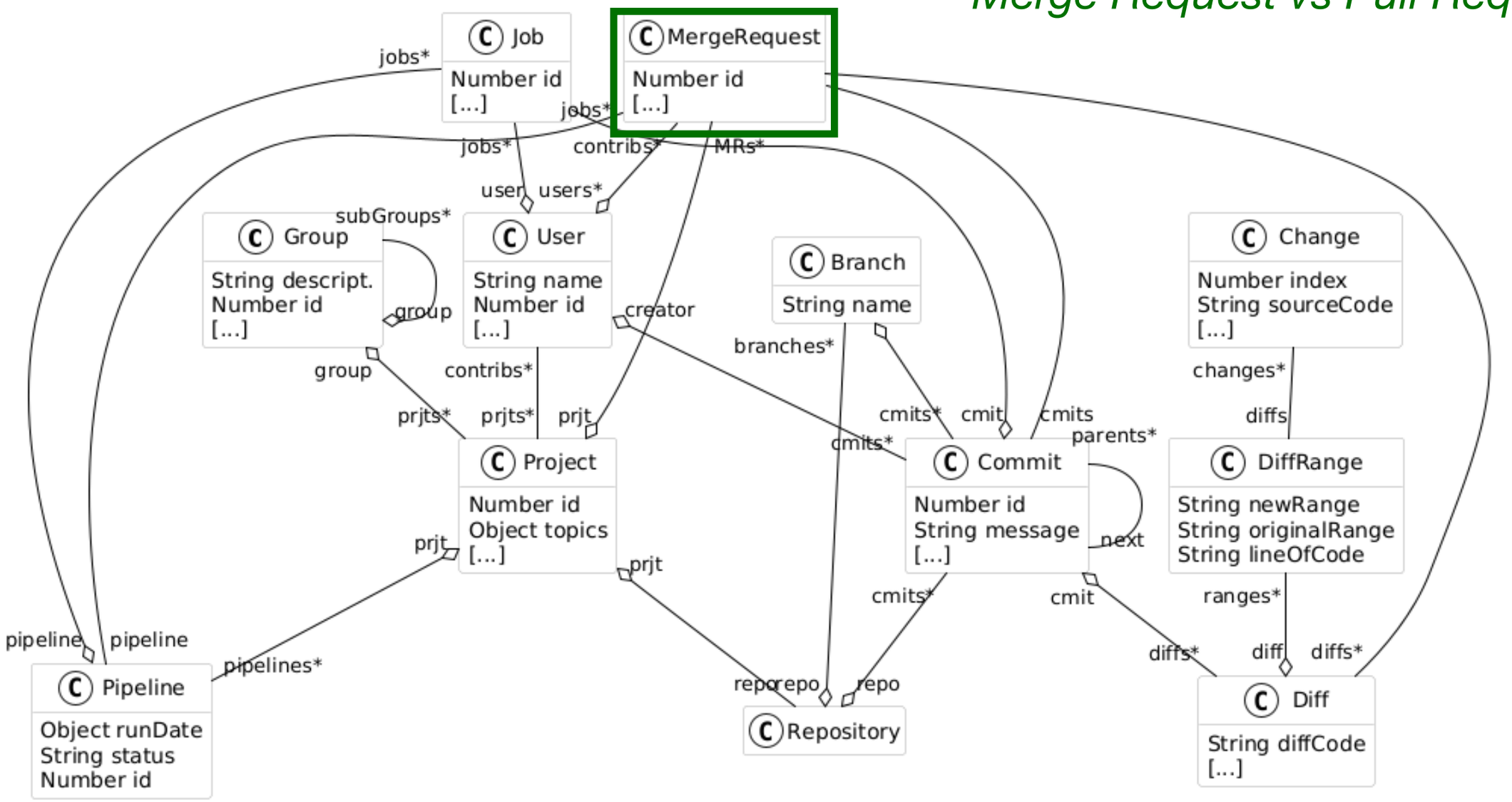


Fig - Simplify Metamodel of a Git Social Platform in GitProjectHealth

Git Model

2. New relations

Commits link to a User, not author

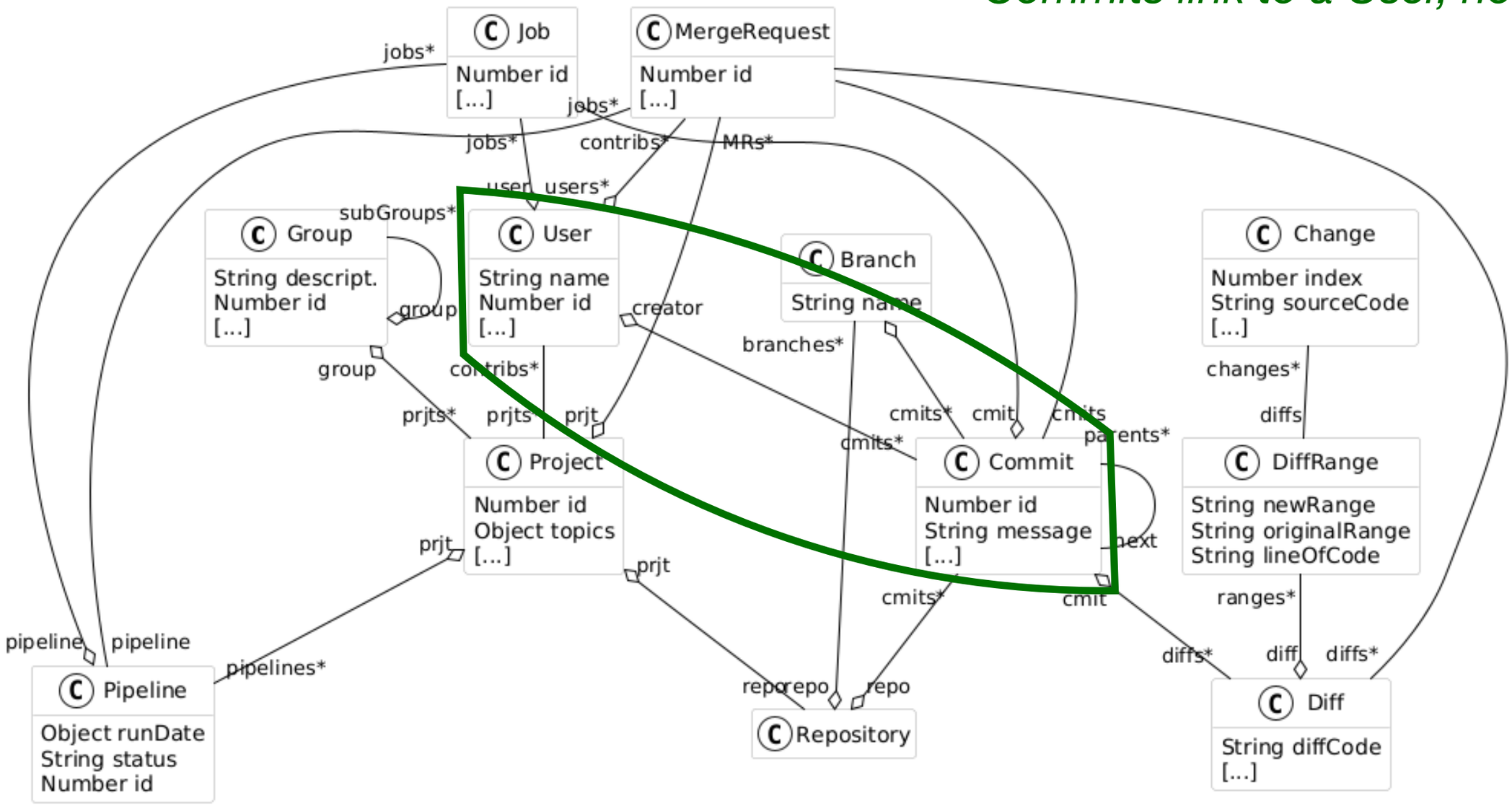


Fig - Simplify Metamodel of a Git Social Platform in GitProjectHealth

GitProjectHealth

Git Model

3. Concepts at fine granularity

Modeling down to the changed line of code

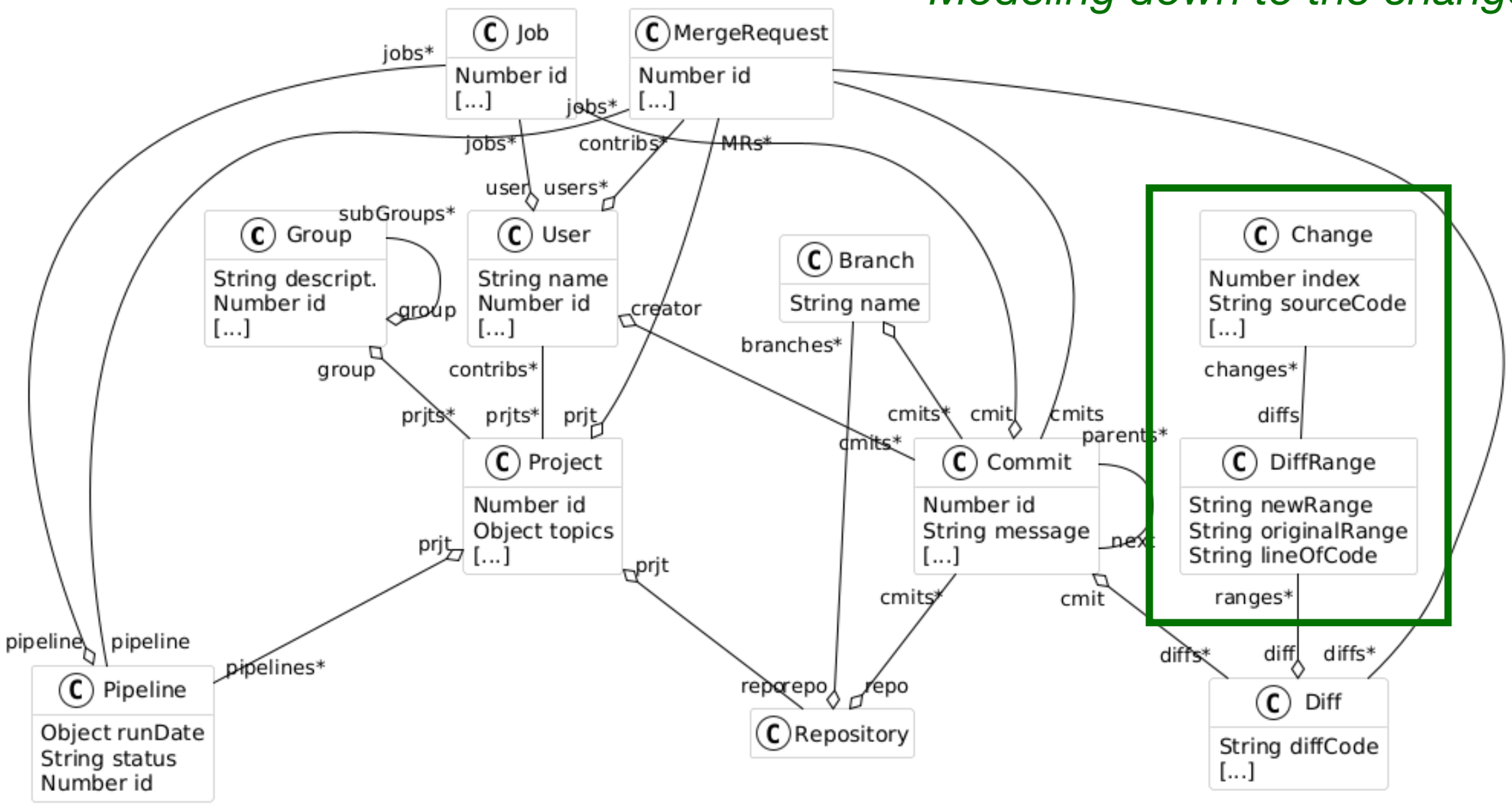


Fig - Simplify Metamodel of a Git Social Platform in GitProjectHealth

API and Importer

Our API are independent open source projects in Pharo.

Anyone can access them via `github/Evref-BL`

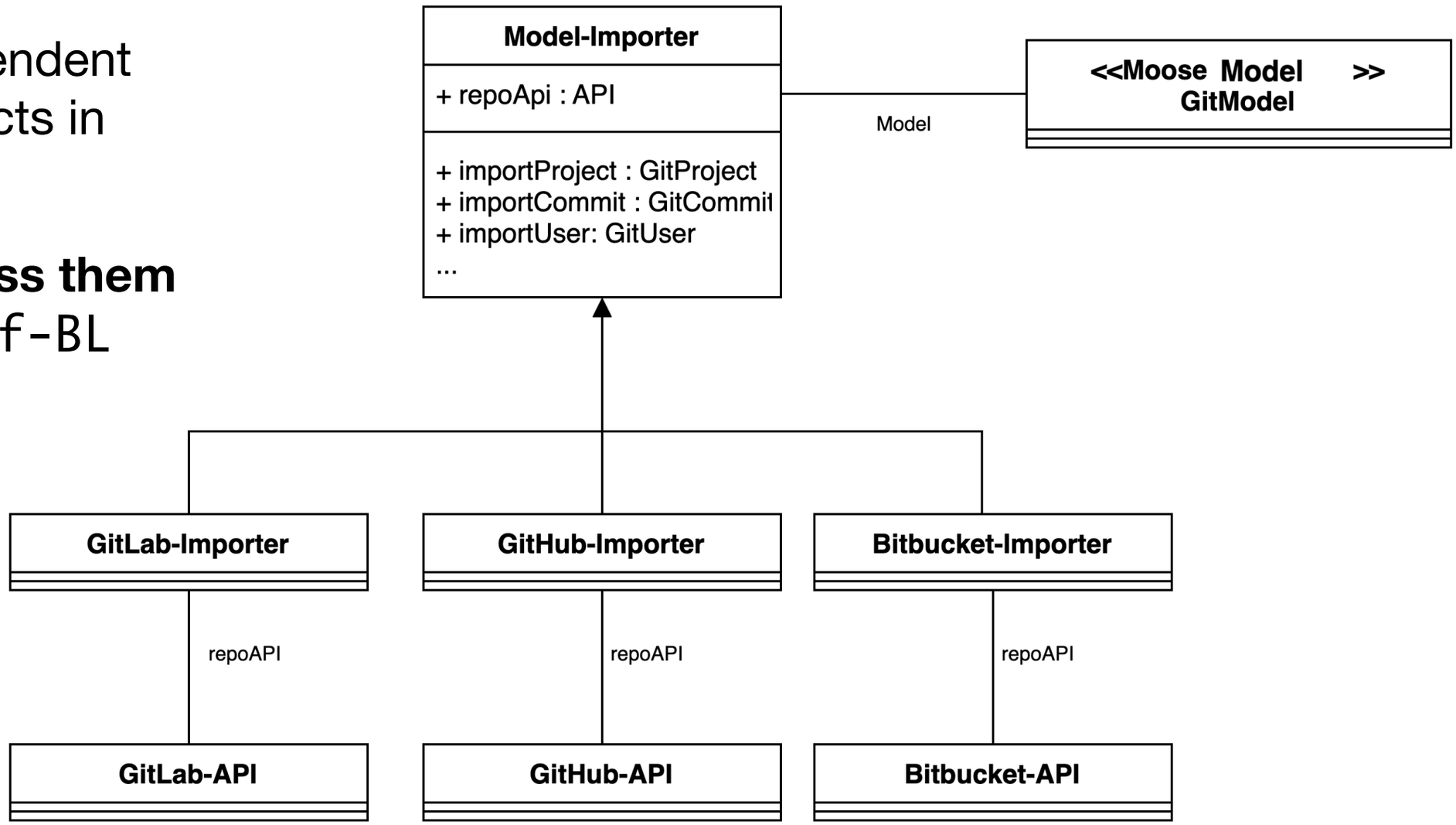
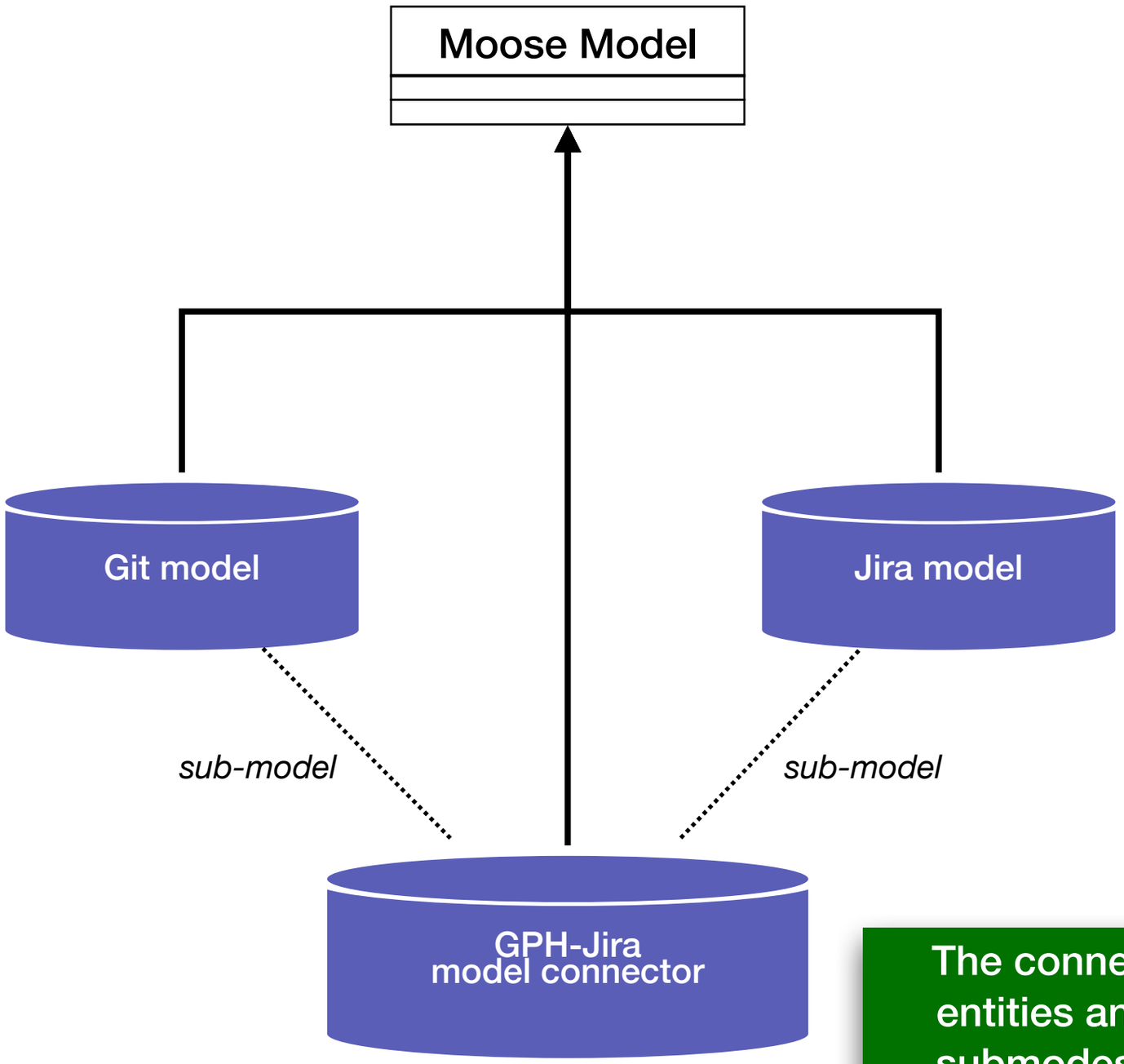


Fig — APIs and GSP importers related to our Git model

Metamodel connection: Jira - Git Model



The connector accesses all the entities and relations of its two submodes (Git and Jira model)

Metamodel connection: Jira - Git Model

GitProject-JiraConnector
GitProject-JiraConnector-Generator
Filter...

GitProjectJiraConnectorGenerator !
Filter...

instance sic
definition
overrides

defineClasses
defineRelations

All Packages | Scoped View | Projects | Flat | Hier. | Inst. side | Class side | Methods | Vars | Class refs. | Implementors

GitProjectJira(× | Comment × | *connectComr × | defineClasses × | Inst. side meth ×

```
1  defineClasses
2
3      super defineClasses.
4      glhMergeRequest := self
5                              remoteEntity: #MergeRequest
6                              withPrefix: #GPH.
7      glhCommit := self remoteEntity: #Commit withPrefix: #GPH.
8      jiraIssue := self remoteEntity: #Issue withPrefix: #JP
```

GitProject-JiraConnector
GitProject-JiraConnector-Generator
Filter...

GitProjectJiraConnectorGenerator !
Filter...

instance sic
definition
overrides

defineClasses
defineRelations

All Packages | Scoped View | Projects | Flat | Hier. | Inst. side | Class side | Methods | Vars | Class refs. | Implementors

Dependencies × | GitProjectJira(× | Comment × | *connectComr × | defineRelation ×

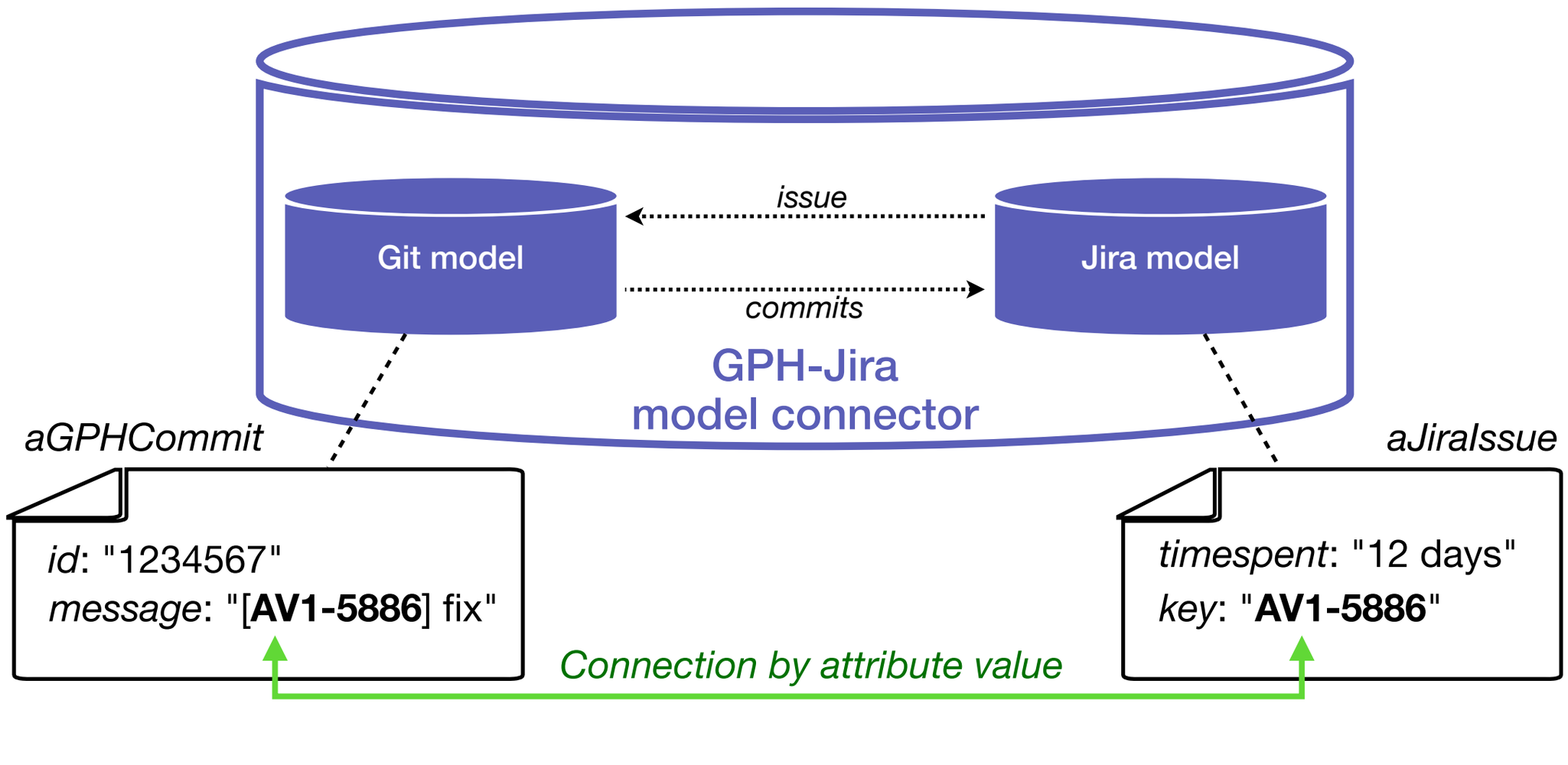
```
1  defineRelations
2
3      super defineRelations.
4      ((glhMergeRequest property: #jiraIssue) comment:
5          'the jira issue associated to this merge request')
6      -<> ((jiraIssue property: #mergeRequest) comment:
7              'A merge request in Git associated to this issue').
8      ((glhCommit property: #jiraIssue) comment:
9          'The jira issue associated with this commit')
10     *-<> ((jiraIssue property: #commits) comment:
11             'The commits associated with this jira issue')
```

s all the
its two
model)

Metamodel connection: Jira - Git Model

```
connectCommit: commit

   (jiraModel allWithType: JPIssue)
      detect: [ :issue | commit message includesSubstring: issue key ]
      ifFound: [ :issue | commit jiraIssue: issue ]
```



Quick demo

Our usage of *GitProjectHealth*

Deploying GPH at Berger-Levrault

Computing MSR Metrics

- We build a metric framework within GPH
  - They are either *Projet* or *User* centric
- For each Metric,
  - it **loads** entities from a time **period** (i.e., two dates)
  - it **calculates** the metric **over** a time windows (i.e. a Day, a Week, a Month, or a Year).
- **47 metrics** are implemented so far.

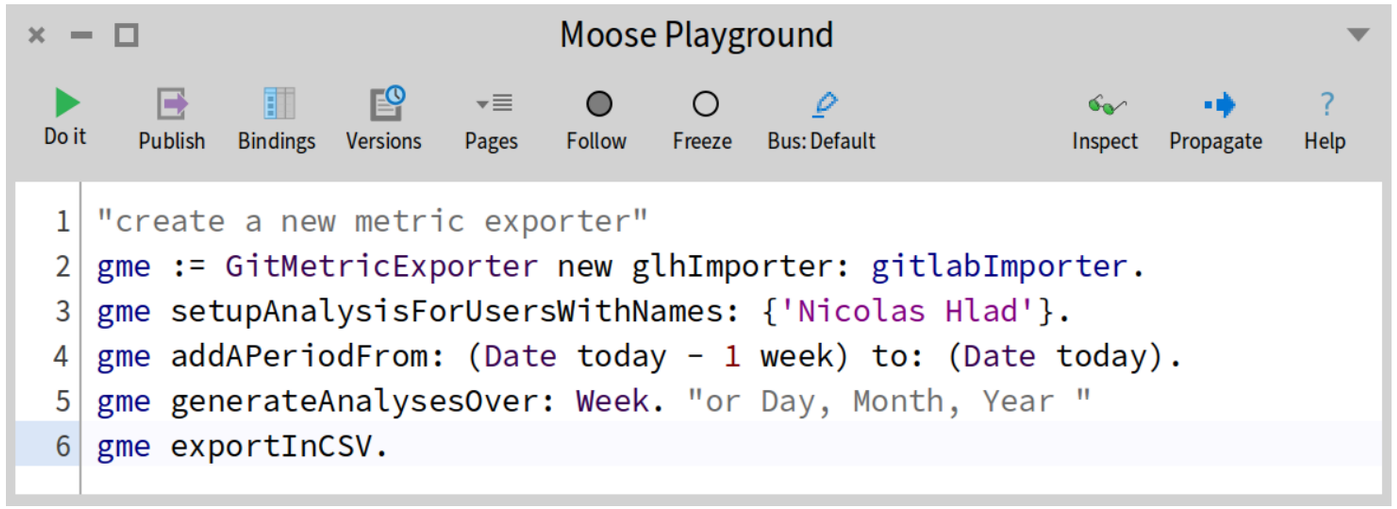


Fig - Running all metrics in GitProjectHealth from a playground (simplified)

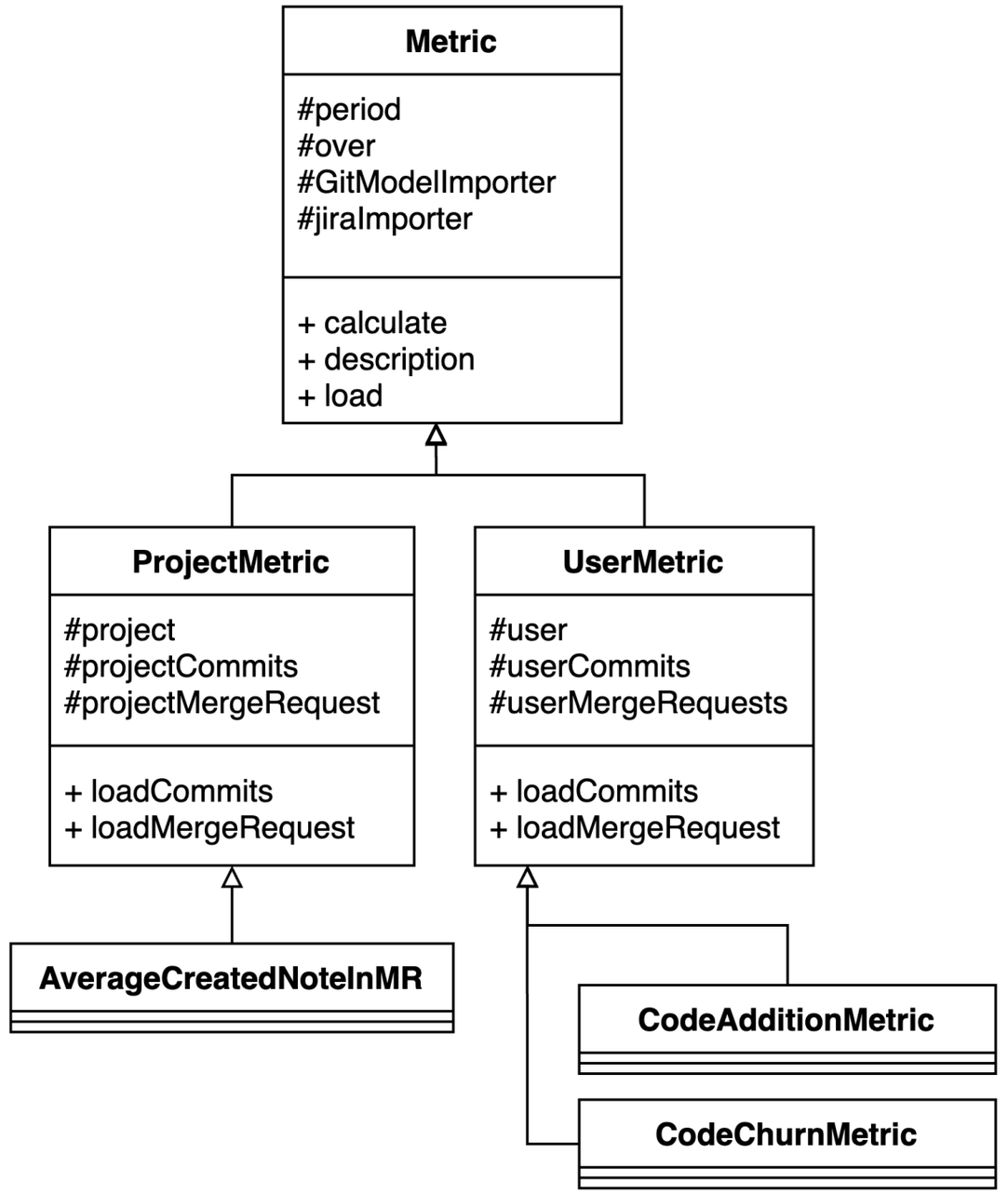


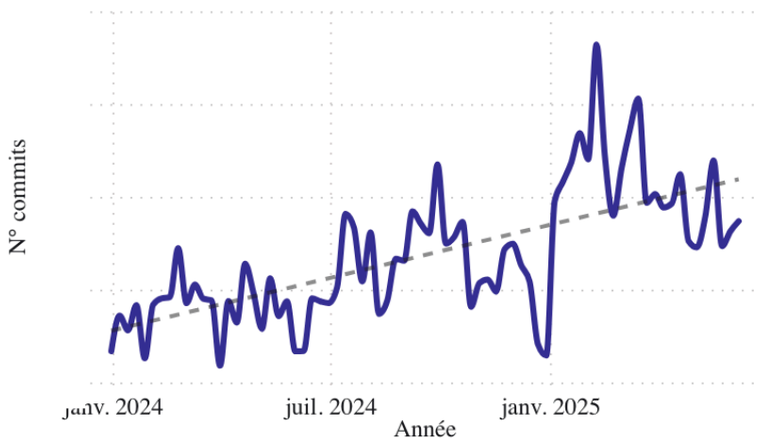
Fig - UML representation of the Metrics in GitProjectHealth

Using GPH at Berger-Levrault

Metrics computed every weeks (from 2024)

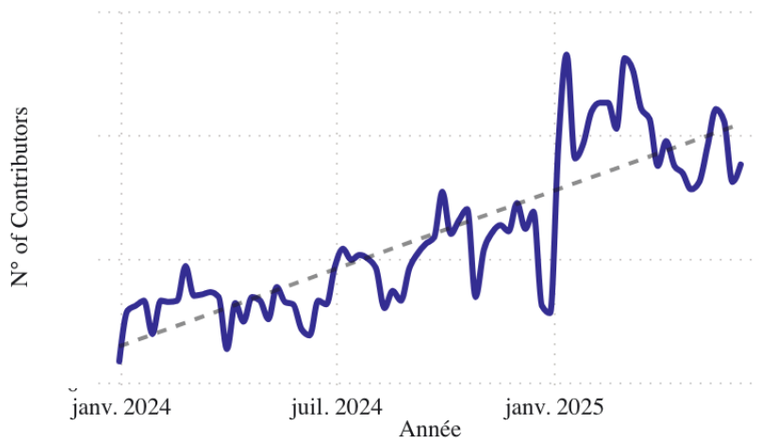
Tendency

N° commits



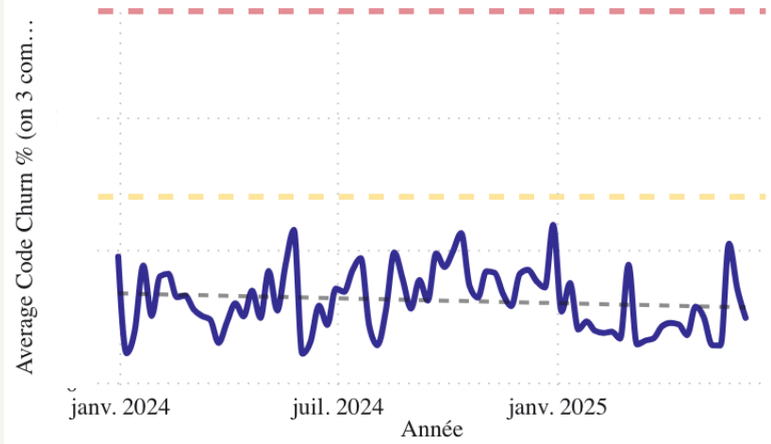
N° of commits should indicate whenever a developer is enough satisfy with a feature development progress
More commits **can** reflect more activity. However, developers can commits as much as they want, so this metric **can not be considered alone.**

N° of Contributors



The number of contributor during the week. It represents the number of unique contributor to the project for the selected period.

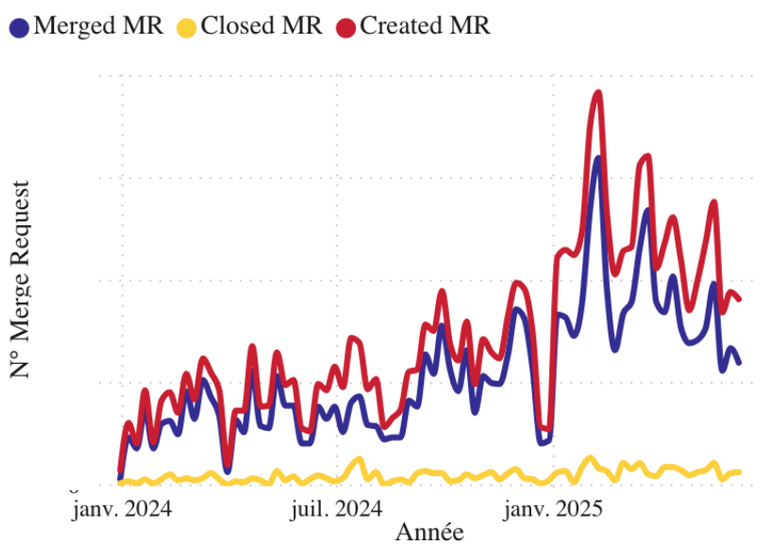
Code Churn



Code Churn represent stability of introduced change
Less is better
Expected less than 7% for old project
Less than 14% for new project

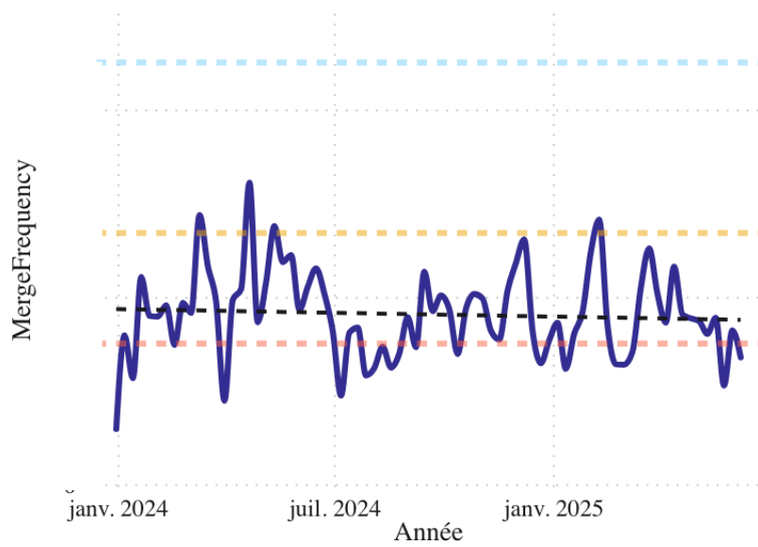
Merge Request metrics

Merge Request created / merged and close



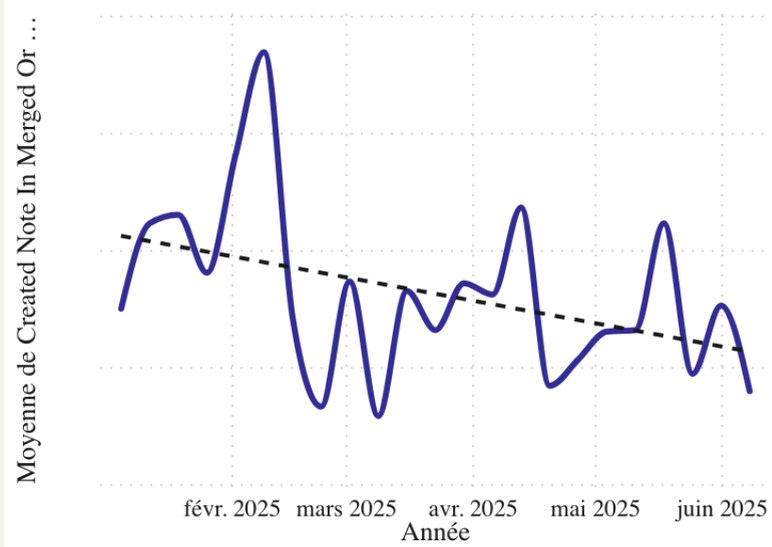
Close Merge Requests represent defective work. Lower the best.
Closed MR merged MR, represent ended work. Created MR represent started work.
When they are close one to the other, it represents that we do not keep lot of unfinished work.

Merge Request Frequency



Merge merged by dev - LinearB 2025 - Merge by dev by week
More is better

Average Merge Request Comments/Note



Average number of comments/notes by MR
A high number means a lot of discussion for many which is unexpected
0/1 means no comments are made, interrogation for code review practice

Conclusion & Perspectives

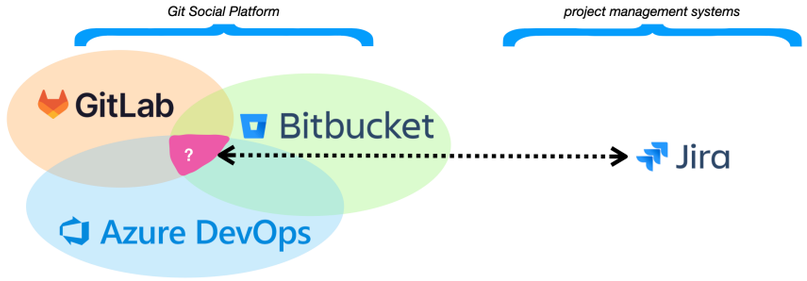
and end.

Conclusion

Recap

Industrial context

The specificities of Berger-Levrault



What tool to mine 3 different Git Social Platforms (GSP) ?

How can we implement GSP metrics efficiently ?

How can you connect GSP data to Jira efficiently ?

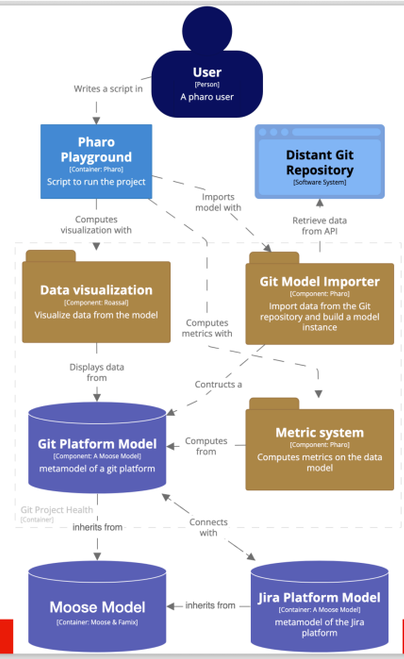
We use Model Driven Engineering with Pharo-Moose

Our MSR solution

GitProjectHealth

Key Features:

- **Data Extraction**:
  Extracts data from major social platforms.
- **Model Import**:
  Imports a model of specific Git entities.
- **Model Connection**:
  Connects models (e.g., Gitlab and Jira).
- **Visualization and Metrics**:
  Visualizes data and computes metrics.

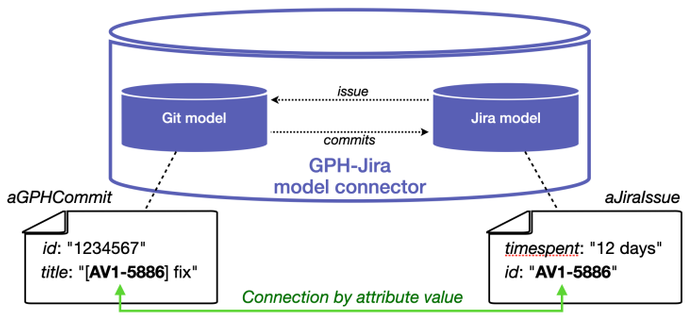


Our MSR Solution

Metamodel connection: Jira - Git Model

```
connectCommit: commit

   (jiraModel allWithType: JPIssue)
      detect: [ :issue | commit message includesSubstring: issue key ]
      ifFound: [ :issue | commit jiraIssue: issue ]
```



Using GPH at Berger-Levrault

Computing MSR Metrics

- We build a metric framework within GPH
  - They are either *Projet* or *User* centric
- For each Metric,
  - it **loads** entities from a time **period** (i.e., two dates)
  - it **calculates** the metric **over** a time windows (i.e. a Day, a Week, a Month, or a Year).
- **47 metrics** are implemented so far.

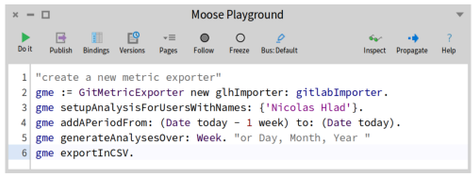


Fig - Running all metrics in GitProjectHealth from a playground (simplified)

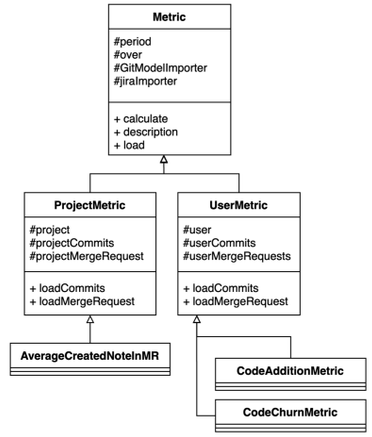


Fig - UML representation of the Metrics in GitProjectHealth

Future Works

- Addressing limitations
  - The difficulty of maintaining a global metamodel by investigating the generation of GSP submetamodels from their OpenAPI
  - Discuss the purpose of the measures and consider which measures correlate with a healthy project.
  - Evaluating GPH against existing tools (PyDriller, Git-Miner, etc)
- Evolution
  - From GPH model to source code model navigating from repository to Famix model
  - Build usable knowledge maps by detecting parts of the repository that have become unknown to developers.

Ressources

Links

GitProjectHealth https://github.com/moosetechnology/GitProjectHealth

Pharo Gitlab API https://github.com/Evref-BL/Gitlab-Pharo-API

Pharo BitBucket API https://github.com/Evref-BL/Bitbucket-Pharo-API

Pharo Jira API https://github.com/Evref-BL/Jira-Pharo-API

Example using GitProjectHealth:

Heatmaps https://github.com/Marpioux/Gitlab-HeatMap

Ressources

Bibliography

[Steffen et al. 2010] Steffen M. Olbrich, Daniela Cruzes, and Dag I. K. Sjùberg. 2010. Are all code smells harmful? A study of God Classes and Brain Classes in the evolution of three open source systems. In 26th IEEE International Conference on Software Maintenance (ICSM 2010), September 12-18, 2010, Timisoara, Romania. 1-10.

[Palomba et al. 2014] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Andrea De Lucia. 2014. Do They Really Smell Bad? A Study on Developers' Perception of Bad Code Smells. In Proc. of the 30th International Conference on Software Maintenance and Evolution. 101-110

[Bacchelli et al. 2013] Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review. In Proc. of the 35th International Conference on Software Engineering. 712-721

[Bacchelli et al. 2010] Alberto Bacchelli, Marco D'Ambros, and Michele Lanza. 2010. Are popular classes more defect prone?. In International Conference on Fundamental Approaches to Software Engineering. Springer, 59-73.

[Mockus et al. 2000] Audris Mockus, Roy T Fielding, and James Herbsleb. 2000. A case study of open source software development: the Apache server. In Proc. of the 22nd international conference on Software engineering. Acm, 263ś272.

[Hassan 2008] A. E. Hassan, "The road ahead for Mining Software Repositories," 2008 Frontiers of Software Maintenance, Beijing, China, 2008, pp. 48-57, doi: 10.1109/FOSM.2008.4659248.

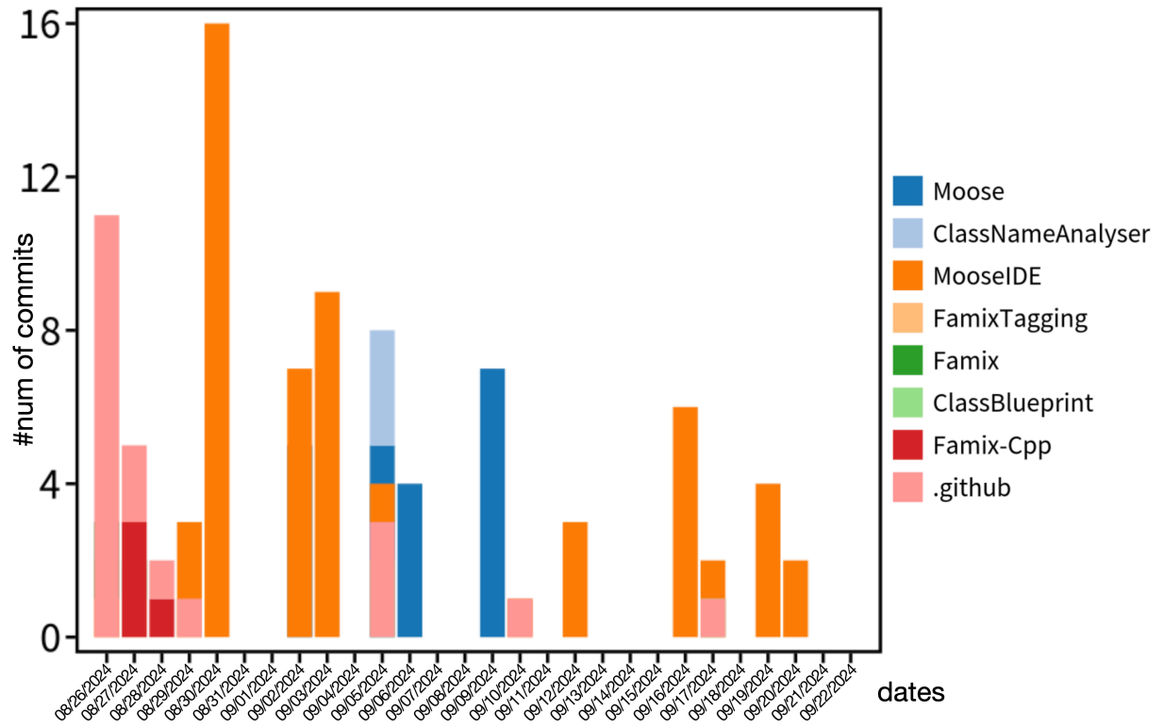
[Dabbish et al. 2012] Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012, February). Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work* (pp. 1277-1286).

Annexe

Texte du titre

Open-Source Analysis (github)

- Analyzed 30 days of activities from **Eclipse**, **Microsoft**, and **MooseTechnology**.
- Analyzed ~4457 entities over 30 days
- Visualizations: daily commit distributions, user activity



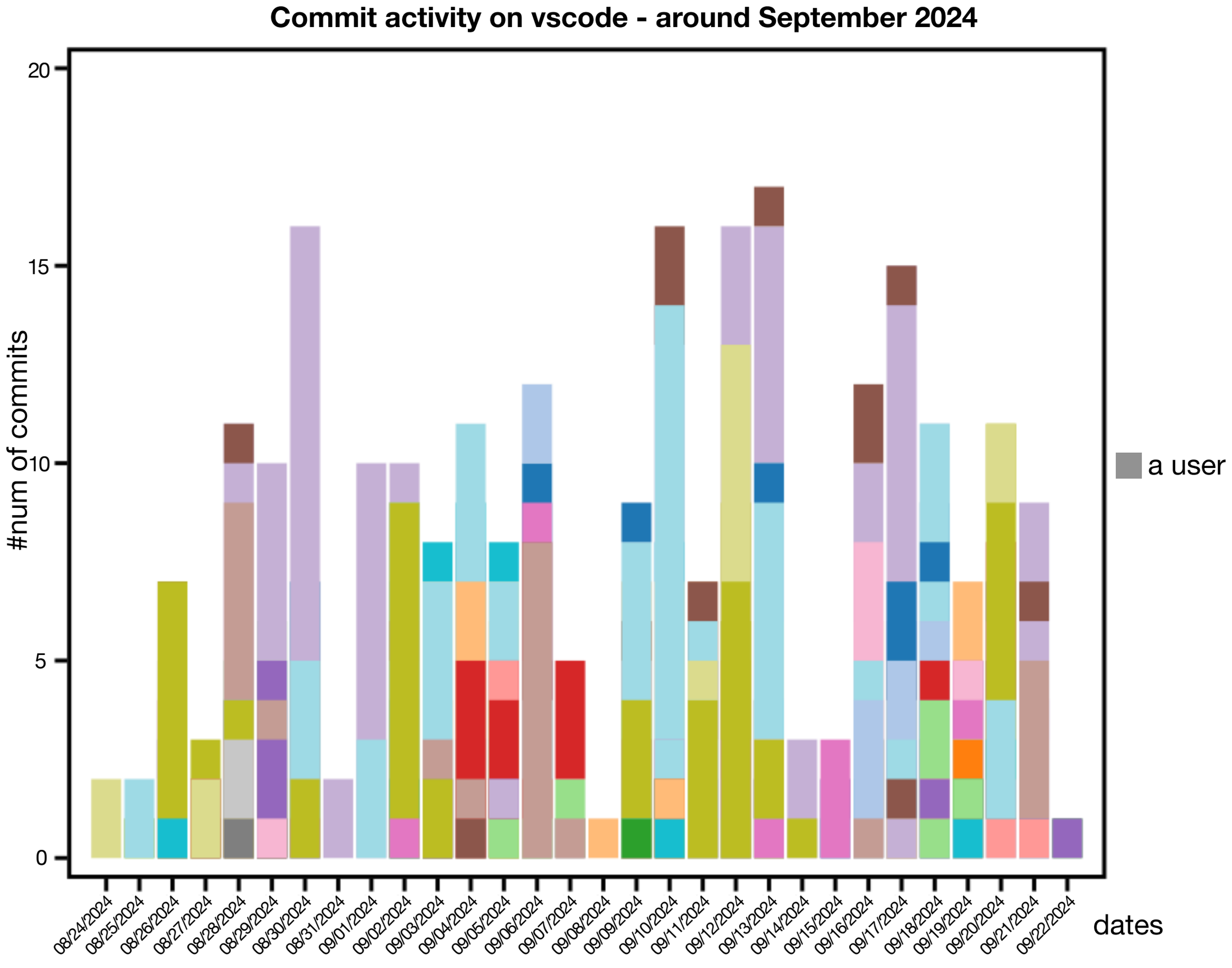
*A user's commit activity by day, during the month of September 2024, on **moosetechnology***

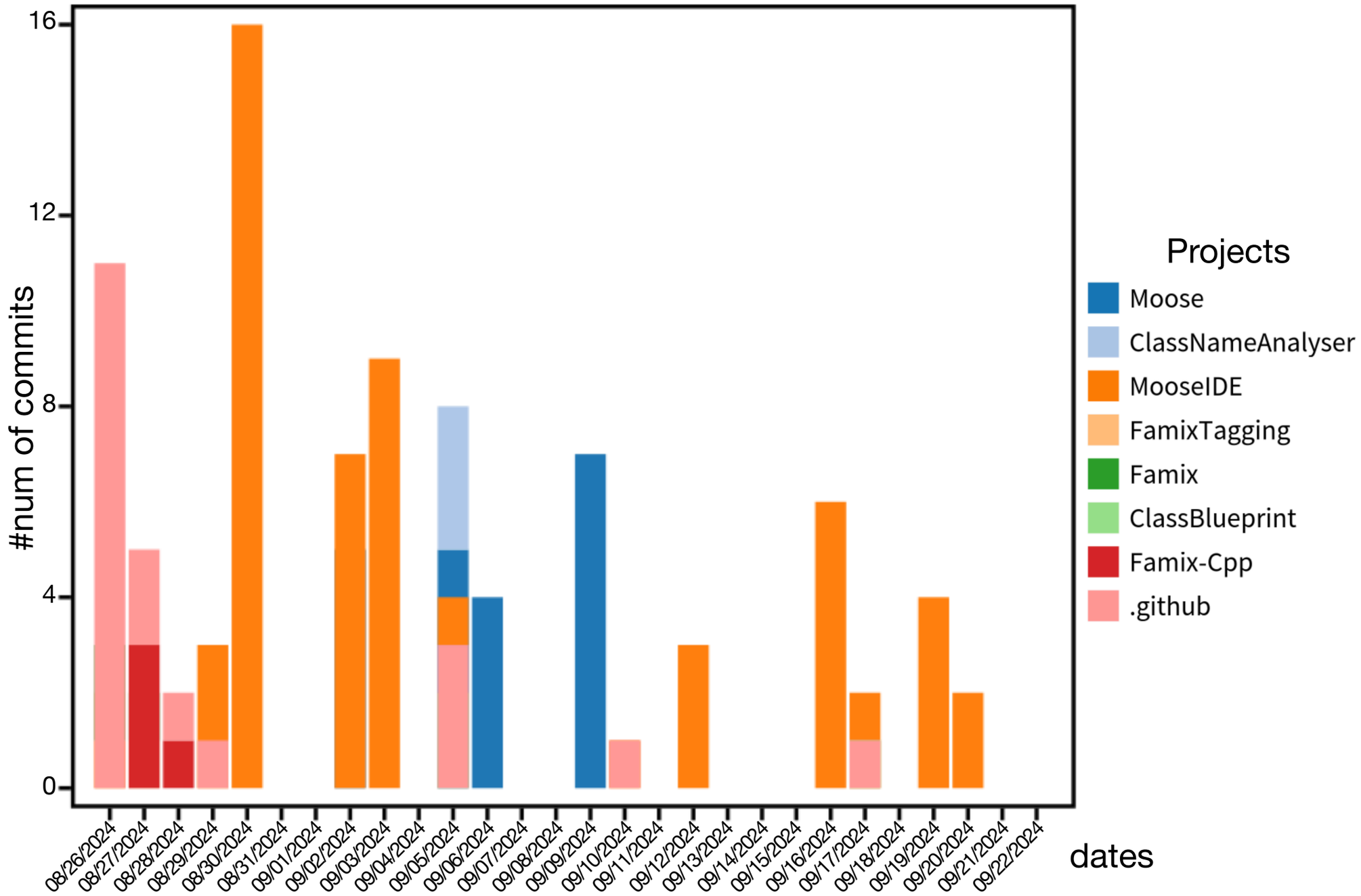
Industrial Case at Berger-Levrault (gitlab)
Connected Jira model with Git model to analyze merge request distributions across different issue types.

- 27% of Merge Requests linked to bug-related Jira issues.

Type	Occurrences	Occurrences %
Anomaly	19	27%
Task	21	30%
Epic	26	37%
None	4	6%
Total	**70**	**100%**

Issues occurrences in September 2024 for **WeHR**


Commit activity on vscode - around September 2024
#num of commits
20
15
10
5
0
08/24/2024
08/25/2024
08/26/2024
08/27/2024
08/28/2024
08/29/2024
08/30/2024
08/31/2024
09/01/2024
09/02/2024
09/03/2024
09/04/2024
09/05/2024
09/06/2024
09/07/2024
09/08/2024
09/09/2024
09/10/2024
09/11/2024
09/12/2024
09/13/2024
09/14/2024
09/15/2024
09/16/2024
09/17/2024
09/18/2024
09/19/2024
09/20/2024
09/21/2024
09/22/2024
dates
a user


16
12
8
4
0
#num of commits
Projects
Moose
ClassNameAnalyser
MooseIDE
FamixTagging
Famix
ClassBlueprint
Famix-Cpp
.github
08/26/2024
08/27/2024
08/28/2024
08/29/2024
08/30/2024
08/31/2024
09/01/2024
09/02/2024
09/03/2024
09/04/2024
09/05/2024
09/06/2024
09/07/2024
09/08/2024
09/09/2024
09/10/2024
09/11/2024
09/12/2024
09/13/2024
09/14/2024
09/15/2024
09/16/2024
09/17/2024
09/18/2024
09/19/2024
09/20/2024
09/21/2024
09/22/2024
dates

Sunburst: last activity on the code base

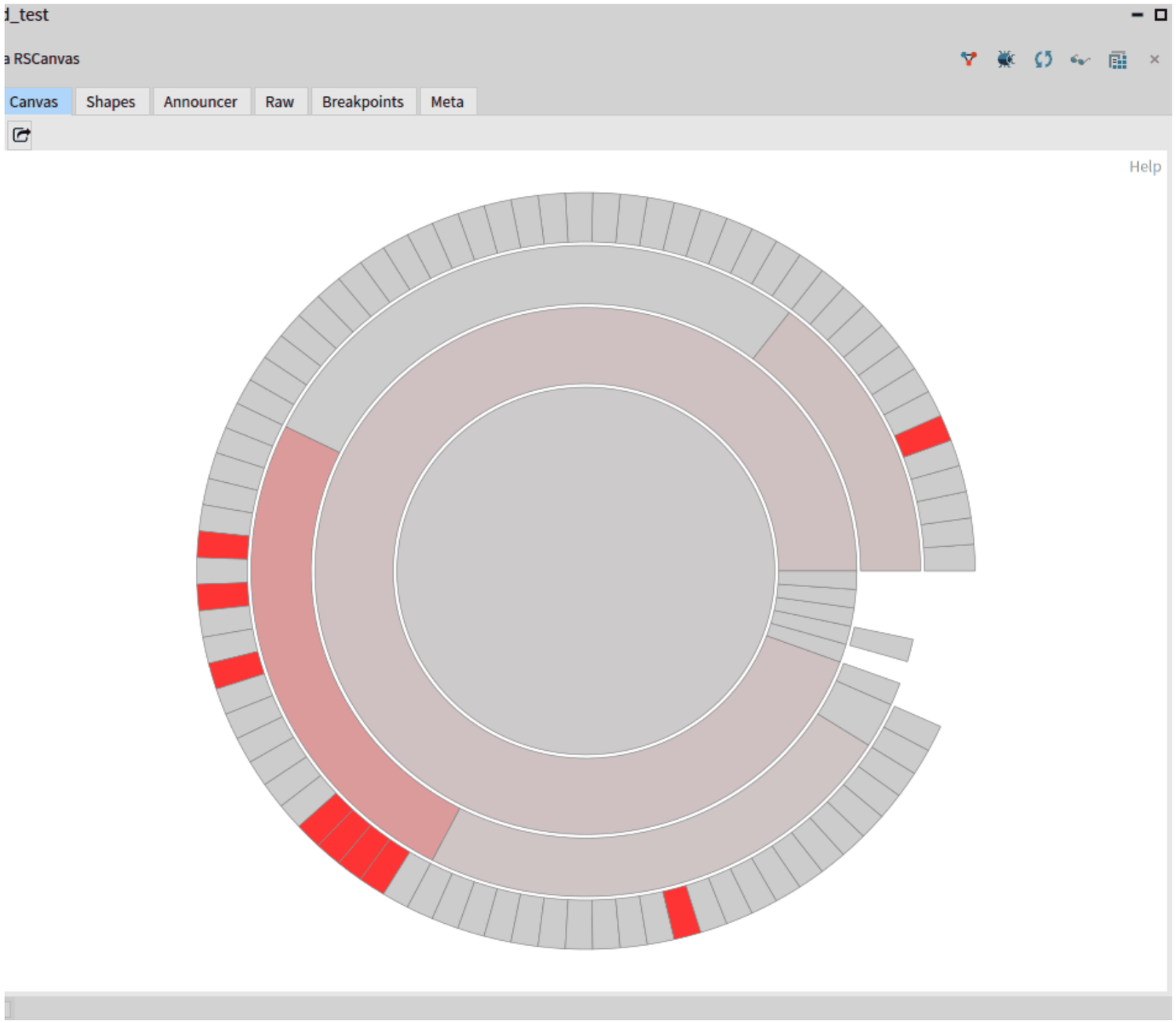


Fig — Sunburst representation of a developer activity in a project