# VisFork: Towards a Toolsuite for Visualizing Fork Ecosystems

Siyue Chen[a], Loek Cleophas[a,b], Sandro Schulze[c], Jacob Krüger[a,*]

[a]*Eindhoven University of Technology, The Netherlands*
[b]*Stellenbosch University, Republic of South Africa*
[c]*Anhalt University of Applied Sciences, Germany*

**Abstract**

In our previous work, we have developed and tested different visualizations that help analyze fork ecosystems. Our goal is to contribute analyses and tools that support developers as well as researchers in obtaining a better understanding of what happens within such ecosystems. In this article, we focus on the tool implementation of our most recent visualizations, which can help users to better understand the relations between and activities within forks. Since fork ecosystems are widely used in practice and well established research subjects, we hope that our tooling constitutes a helpful means for other researchers, too.

*Keywords:* Fork Ecosystems, Software Repositories, Visualizations, Variant-Rich Systems, Software Evolution

## Metadata

| **code metadata description** | |
|---|---|
| Current code version | v1 |
| Permanent link to code/repository used for this code version | `https://github.com/chensiyue98/visfork` |
| Permanent link to Reproducible Capsule | `https://zenodo.org/records/10462694` |
| Legal Code License | MIT |
| Code versioning system used | Git |
| Software code languages, tools, and services used | JavaScript, HTML, GitHub API, GitHub |
| Compilation requirements, operating environments and dependencies | Node.js, Next.js, Vercel |
| If available, link to developer documentation/manual | `https://github.com/chensiyue98/visfork/blob/main/README.md` |
| Support email for questions | `j.kruger@tue.nl` |

## 1. Motivation and Significance

Forking allows developers to copy a repository and work on that independent copy without impacting the original repository [7, 8]. Over time, forking has become widely

*Corresponding Author
*Email addresses:* `l.g.w.a.cleophas@tue.nl` (Loek Cleophas), `sandro.schulze@hs-anhalt.de` (Sandro Schulze), `j.kruger@tue.nl` (Jacob Krüger)

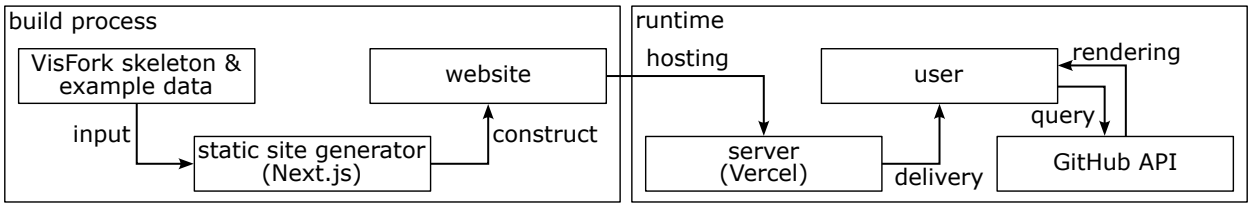


Figure 1: Structural overview of VisFork.

adopted and large fork ecosystems have emerged in industry and open-source development [10, 11, 12, 13, 18, 22]. While facilitating collaborative software development, forking challenges developers as well as researchers in keeping an overview of large fork ecosystems, such as the Linux Kernel (49,400 forks) or the Marlin 3D printer firmware (18,500 forks) [11, 12, 13, 15, 18, 20]. Due to their practical relevance, fork ecosystems have become a primary research subject with researchers working on various topics like identifying redundant or overlooked contributions [2, 6, 9, 15, 16, 21, 22]. GitHub has also introduced a Network Graph view that presents commits in the order of submission time. However, a graph of commits may not be expressive of the changes that occurred within a fork, since commits can be submitted with unreliable descriptions and can contain tangled or independent changes [1].

In our recent work [3, 4, 14], we have analyzed and proposed different visualizations aiming to support developers and researchers in understanding as well as comparing forks in ecosystems. Most prominently, we have designed and tested six visualizations in VisFork that we derived from 237 comments of GitHub developers on how to improve the GitHub Network Graph [4]. While subject to future improvements, we want to contribute this first prototype of our toolsuite VisFork to enable other researchers to use and extend it. To the best of our knowledge, we are the first to contribute such a reusable and extendable toolsuite, supporting researchers with visualizations to explore phenomena in large real-world systems. For instance, our user study with ten GitHub developers and seven graduate students [4] indicated that they already perceived most visualizations as helpful to tackle questions like: (i) *How did this fork evolve and how is it related to other forks?* (ii) *Which forks are still actively maintained?* (iii) *What (types of) work has been done within a fork?* Answering such questions is helpful for researchers to filter interesting forks and tackle novel research questions, thereby supporting a very active research area.

## 2. Software description and functionalities

Currently, VisFork works as a static website generator, following the structure we display in Figure 1. In the following, we provide a brief overview of VisFork before summarizing its two core functionalities: data retrieval and visualization.

**General Overview.** We decided to build on web technologies to have a smoother integration with GitHub, particularly making use of GitHub's robust Application Programming Interfaces (APIs)—including Representational State Transfer (REST) and GraphQL. For

our current prototype, we use the REST API. Additionally, web-based tools are accessible from any device with a web browser, without the need for specialized software installations. This ensures that our visualization tool can be accessed by a broad audience, regardless of their device or operating system. To implement our visualizations, we use D3.js.[1] We selected this library because our comparative analysis [3] of different tools has shown that it is most helpful when combining different visualizations. For developing the VisFork website, we chose Next.js[2] because this framework is renowned for its optimized performance. Using its support for Static Site Generation (SSG), we generate the HTML at build time, meaning the content is ready to be served to users as soon as they request it. This results in faster page load times, as there is no need to wait for server-side computations at runtime. To deal with dynamic data when querying GitHub's API on the user side, we utilize Client-Side Rendering (CSR) on the user query. So, data retrieving and visualization rendering are done on the user's browser, saving the cost of communicating with a backend system. By combining SSG and CSR, we can ensure that the bulk of the application is pre-rendered and served quickly, while dynamic data interactions remain fluid and responsive. This hybrid approach helps us to build fast initial page loads and interactive and real-time data visualization.

**Data Retrieval and Processing.** To query data, a user has to set up the GitHub authorization using the provided token. Then, VisFork fetches a specified repository's metadata, for which the number of forks to fetch and the sort order can be customized. For each fork, we fetch its branches and the commits of each branch within a specified date range. Please note that duplicate commits can occur in case the same commit is part of multiple branches.

**Data Visualizations.** We used D3.js as follows to implement visualizations:

**Data Transformation:** We mold the data fetched from GitHub's APIs into a D3-friendly format.

**Crafting SVG Elements:** We employ D3.js to generate SVG elements, such as nodes in a network graph or bars in a histogram.

**Data Binding:** We associate our refined data with the SVG elements, translating data points into visual representations.

**User Interaction:** We define specific responses to user actions, such as mouse clicks or hovers, enriching the user's engagement with the data.

Following the same process, other researchers can extend VisFork with visualizations based on their own requirements. Currently, VisFork supports the visualizations we summarize in the next section and display in Figure 2.

[1] `https://d3js.org`
[2] `https://nextjs.org`

## 3. Illustrative examples

In Figure 2, we provide an overview of the website generated by VisFork and its individual visualizations. We have used this small example as sample data for our previous user study [4]. Overall, we currently support six visualizations:

① A date range slider that (1) provides a general overview of development activity in a fork ecosystem and (2) allows users to select what data to look into in more detail.

② A commit timeline as a well-known visualization of commits, forks, and branches in an ecosystem. We have also developed a merged view, in which not all commits are displayed, but only those with deviations (i.e., forking, branching, merging). Again, the user can select those commits which are most relevant to them for the following visualizations. Note that the basic commit timeline is essentially the same visualization as GitHub's Network Graph, while all other visualizations are extensions.

③ A commit detail list, which is a tabular overview of the selected commits and their properties.

④ A word cloud that summarizes the most frequent terms occurring in the selected commits' messages.

⑤ A commit classification tab that uses keyword matching to assign each commit into one of the maintenance activities defined by Swanson [19] and visualizes these via a Sankey diagram.

⑥ A collaboration network history tab, which displays an evolving graph that illustrates which developers contribute to what forks over time through an animation.

For users who are unfamiliar with VisFork, we have designed an introductory tour that pops out when a user visits the website for the first time.

## 4. Impact

To the best of our knowledge, VisFork is the first attempt at systematizing the development of visualizations for fork ecosystems. We have conducted a user study [4] with ten GitHub developers and seven graduate students who were positive about our prototype compared to GitHub Network Graph, and provided critical feedback for improving it as well as individual visualizations. The participants were invited to conduct certain tasks and freely explore VisFork on any repositories of their interest. Subsequently, the participants completed an anonymous survey. The overall feedback from GitHub community members was positive, with several features appreciated, such as commit history and collaboration network history. We also received criticisms regarding certain scenarios, such as rule-based commit classifications [5], which can be improved by more advanced classification algorithms like natural-language processing with transfer learning [17]. Due to the positive feedback from the survey, we hope that we can expand on VisFork to also transfer it into a plugin

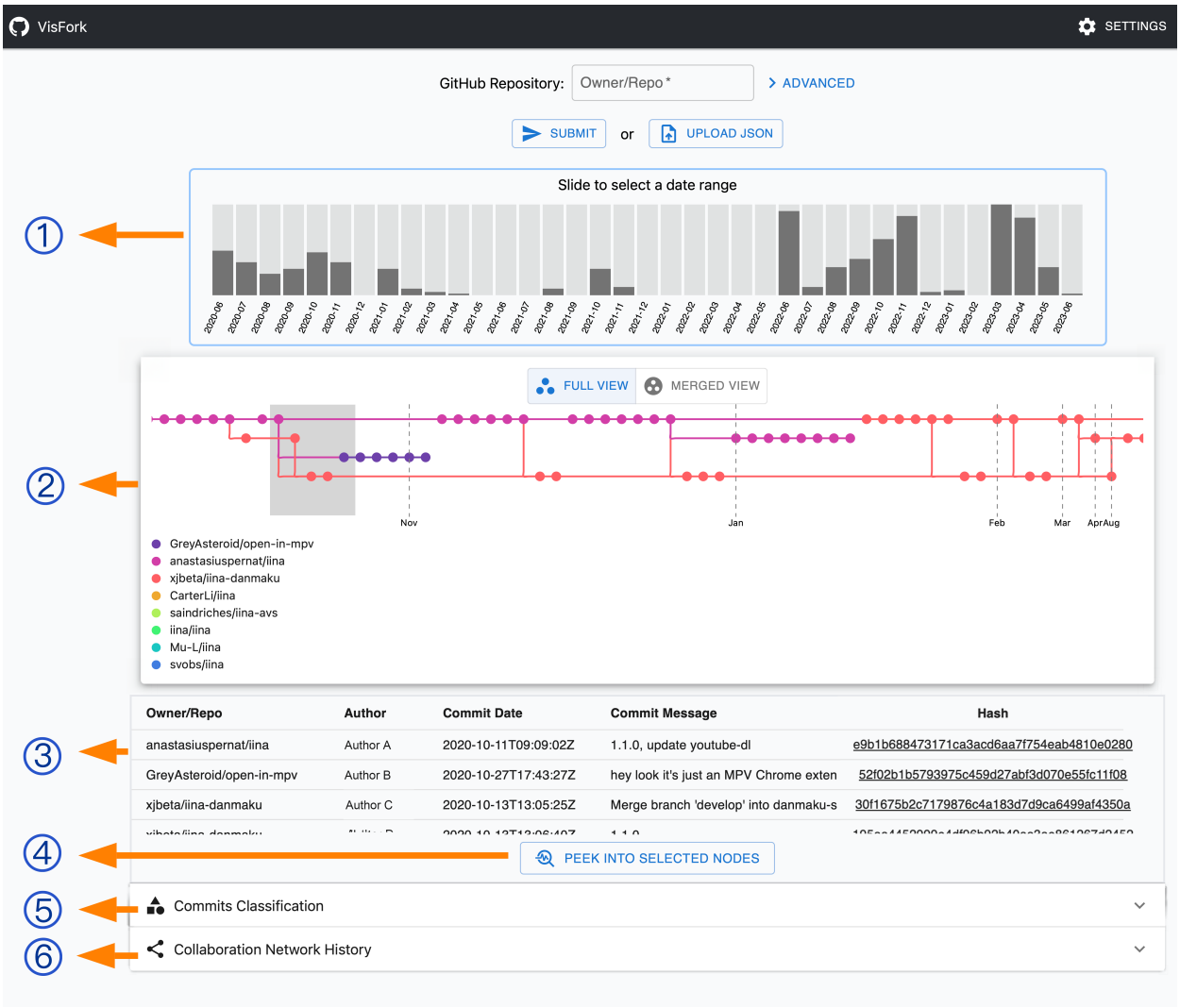


Figure 2: Overview screenshot from our previous work [4] of VisFork for a smaller, real-world fork ecosystem (author names are anonymized): ① date range slider; ② full commit timeline; ③ commit detail list; ④ word cloud; ⑤ commit classification tab; and ⑥ collaboration network history tab.

for GitHub and other social-coding platforms. At the moment, VisFork's visualizations are helpful for researchers to obtain a first understanding of a fork ecosystem to design their research. In particular, VisFork can facilitate existing research on fork ecosystems, such as identifying activity, stale forks, redundant work, code transplantation, bug propagation, or community interactions. These visualizations can also guide the development of new research directions, for instance, on software visualizations and uncovering reuse potential within fork ecosystems. So, we hope that sharing this initial and platform-independent prototype of visualizations can help other researchers working in these directions. We aim to extend and refine VisFork in the future with new functionalities and visualizations to develop it into a complete toolsuite.

## 5. Conclusions

In this article, we have detailed the implementation of VisFork, a first toolsuite aimed at providing visualizations for supporting developers and researchers in obtaining a better overview of fork ecosystems. VisFork has been perceived positively in a user evaluation [4]. However, our participants also noted limitations of the current visualizations and technical improvements we aim to tackle in the future:

1. It shows only direct forks of the original repository, but not forks of these forks.
2. It needs more elaborate visualizations for the relationships between forks and branches.
3. It does not yet allow to filter or search specific forks, making it somewhat difficult to navigate through large networks.
4. It does not provide a view to distinguish actively maintained forks from stale or abandoned ones, yet.

Besides improving VisFork along these dimensions, refining it with more visualizations and improving its performance are key. Furthermore, we need more user studies to evaluate the usefulness of VisFork and its visualizations. To this end we see the integration into social-coding platforms via plugins as an important engineering challenge to make VisFork more usable.

## Acknowledgements

We would like to thank everyone who has supported our research, for instance, by testing our prototypes or participating in our user studies.

## References

[1] Mike Barnett, Christian Bird, João Brunet, and Shuvendu K. Lahiri. Helping Developers Help Themselves: Automatic Decomposition of Code Review Changesets. In *International Conference on Software Engineering (ICSE)*, pages 134–144. IEEE, 2015. doi: 10.1109/icse.2015.35.

[2] John Businge, Moses Openja, Sarah Nadi, and Thorsten Berger. Reuse and Maintenance Practices among Divergent Forks in Three Software Ecosystems. *Empirical Software Engineering*, 27(2), 2022. doi: 10.1007/s10664-021-10078-2.

[3] Siyue Chen, Loek Cleophas, and Jacob Krüger. A Comparison of Visualization Concepts and Tools for Variant-Rich System Engineering. In *International Systems and Software Product Line Conference (SPLC)*, pages 153—-159. ACM, 2023. doi: 10.1145/3579027.3608986.

[4] Siyue Chen, Cleophas Loek, Sandro Schulze, and Jacob Krüger. Use the Forks, Look! Visualizations for Exploring Fork Ecosystems. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 993–1004. IEEE, 2024. doi: 10.1109/SANER60148.2024.00107.

[5] Leshem Choshen and Idan Amit. ComSum: Commit Messages Summarization and Meaning Preservation. *CoRR*, 2021. URL `https://doi.org/10.48550/arXiv.2108.10763`.

[6] Anh N. Duc, Audris Mockus, Randy Hackbarth, and John Palframan. Forking and Coordination in Multi-Platform Development. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 59:1–10. ACM, 2014. doi: 10.1145/2652524.2652546.

[7] Neil A. Ernst, Steve M. Easterbrook, and John Mylopoulos. Code Forking in Open-Source Software: A Requirements Perspective. *CoRR*, 2010. URL `https://doi.org/10.48550/arXiv.1004.2889`.

[8] Georgios Gousios, Martin Pinzger, and Arie van Deursen. An Exploratory Study of the Pull-Based Software Development Model. In *International Conference on Software Engineering (ICSE)*, pages 345–355. ACM, 2014. doi: 10.1145/2568225.2568260.

[9] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie van Deursen. Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective. In *International Conference on Software Engineering (ICSE)*, pages 358–368. IEEE, 2015. doi: 10.1109/icse.2015.55.

[10] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The Promises and Perils of Mining GitHub. In *International Working Conference on Mining Software Repositories (MSR)*, pages 92–101. ACM, 2014. doi: 10.1145/2597073.2597074.

[11] Sebastian Krieter, Jacob Krüger, Thomas Leich, and Gunter Saake. VariantInc: Automatically Pruning and Integrating Versioned Software Variants. In *International Systems and Software Product Line Conference (SPLC)*, pages 129–140. ACM, 2023. doi: 10.1145/3579027.3608984.

[12] Jacob Krüger and Thorsten Berger. An Empirical Analysis of the Costs of Clone- and Platform-Oriented Software Reuse. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 432–444. ACM, 2020. doi: 10.1145/3368089.3409684.

[13] Jacob Krüger, Mukelabai Mukelabai, Wanzi Gu, Hui Shen, Regina Hebig, and Thorsten Berger. Where is My Feature and What is it About? A Case Study on Recovering Feature Facets. *Journal of Systems and Software*, 152:239–253, 2019. doi: 10.1016/j.jss.2019.01.057.

[14] Jacob Krüger, Alex Mikulinski, Sandro Schulze, Thomas Leich, and Gunter Saake. DSDGen: Extracting Documentation to Comprehend Fork Merges. In *International Systems and Software Product Line Conference (SPLC)*, pages 47–56. ACM, 2023. doi: 10.1145/3579028.3609015.

[15] Mukelabai Mukelabai, Christoph Derks, Jacob Krüger, and Thorsten Berger. To Share, or Not to Share: Exploring Test-Case Reusability in Fork Ecosystems. In *International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. doi: 10.1109/ASE56229.2023.00191.

[16] Poedjadevie K. Ramkisoen, John Businge, Brent van Bladel, Alexandre Decan, Serge Demeyer, Coen De Roover, and Foutse Khomh. PaReco: Patched Clones and Missed Patches among the Divergent Variants of a Software Family. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 2022. doi: 10.1145/3540250.3549112.

[17] Muhammad U. Sarwar, Sarim Zafar, Mohamed W. Mkaouer, Gursimran S. Walia, and Muhammad Z. Malik. Multi-Label Classification of Commit Messages Using Transfer Learning. In *International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 37–42. IEEE, 2020. doi: 10.1109/ISSREW51248.2020.00034.

[18] Ștefan Stănciulescu, Sandro Schulze, and Andrzej Wąsowski. Forked and Integrated Variants in an Open-Source Firmware Project. In *International Conference on Software Maintenance and Evolution (ICSME)*, pages 151–160. IEEE, 2015. doi: 10.1109/icsm.2015.7332461.

[19] E. Burton Swanson. The Dimensions of Maintenance. In *International Conference on Software Engineering (ICSE)*, pages 492–497. IEEE, 1976.

[20] Shurui Zhou. Improving Collaboration Efficiency in Fork-Based Development. In *International Conference on Automated Software Engineering (ASE)*, pages 1218–1221. IEEE, 2019. doi: 10.1109/ASE.2019.00144.

[21] Shurui Zhou, Bogdan Vasilescu, and Christian Kästner. What the Fork: A Study of Inefficient and Efficient Forking Practices in Social Coding. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 350–361. ACM, 2019.

[22] Shurui Zhou, Bogdan Vasilescu, and Christian Kästner. How Has Forking Changed in the Last 20 Years? A Study of Hard Forks on GitHub. In *International Conference on Software Engineering (ICSE)*, pages 445–456. ACM, 2020. doi: 10.1145/3377811.3380412.