Fostering Collaboration of Academia and Industry by Open Source Software

David Baum[1], Pascal Kovacs[2], Richard Müller[3]

Abstract: In 2017 and 2018 we released two of our research prototypes as open source. We explain our motivation and concerns at that time and compare them with our actual experience. We also describe how open source releases enabled collaboration with industrial partners. Finally, we show how research projects can extend their funding through grants for open source software. We share our experiences with the initiative Google Summer of Code and show how we overcame bureaucratical hurdles and how our research has benefited from participating in this program.

Keywords: Open Source; Industry Collaboration; Technology Transfer

1 Introduction

For 12 years our research group *Visual Software Analytics* at Leipzig University has been combining findings and methods from the fields of software analytics, software visualization, data science, and empirical software engineering to extract, visualize, and analyze software-related data. The majority of our work are basic research and empirical evaluation which are usually not of direct interest to practitioners. However, we also develop software prototypes which form the basis of our empirical evaluations. The research project is financed only to a very small extent by third-party funds, but is driven primarily by budget funds and student participation. Therefore, the team consists of only four researchers working part-time for the project, two of them being university employees. The research is supported to some extent by the Institute for Applied Informatics (InfAI), which is an affiliated institute of Leipzig University to promote science and research in the areas of computer science and information systems. Within the research group, we discussed for a long time whether an open source release would be useful for us. Open soure releases have a strong tradition in computer science. Nevertheless, the benefits for basic research projects are not always obvious. For this reason, we would like to share our experiences with systems that we have published as open source. We will briefly present *Getaviz* and *jQAssistant Dashboard* and discuss

[1] Leipzig University, Information Systems Institute, Grimmaische Straße, 04109 Leipzig, Germany david.baum@uni-leipzig.de
[2] GISA GmbH, Consulting, Leipziger Chaussee 191A, 06112 Halle (Saale), Germany pascal.kovacs@gisa.de
[3] Leipzig University, Information Systems Institute, Grimmaische Straße, 04109 Leipzig, Germany rmueller@wifa.uni-leipzig.de

doi:10.18420/SE2020_52

advantages, disadvantages, and recommendations. Afterwards we will discuss opportunities for funding open source software development beyond conventional third-party funding.

2 Getaviz

With Getaviz[4] users can solve software engineering problems visually by exploring software artifacts. Getaviz visualizes the structure and runtime behavior of several programming languages, i.e, Java, Ruby, C#, JavaScript, and ABAP. It can enrich these visualizations with evolutionary information from git- and svn-repositories. Among other things it provides proper visualizations and an interactive user interface for identifying and refactoring architectural antipatterns, locating runtime bottlenecks, assessing software quality, and tracking code changes across multiple versions. Getaviz provides a variety of two- and three-dimensional visualizations, that can be explored via browser, HTC Vive, or Microsoft HoloLens. You can find a selection of showcases on our website[5]. Getaviz includes an evaluation server. Its main objective is to conduct empirical evaluations locally and remotely in an efficient and reproducible way. The complete codebase has about 200.000 lines of code in JavaScript, Java, and Ruby.

We did not develop Getaviz for a one-time research project. Instead, we planned from the beginning to create a long-living system that is developed continuously and gets reused across different research projects. Getaviz has been developed as closed source for about ten years from 2007 – 2017. At this time it was used only internally and there was no collaboration with industry so far, i.e., companies did not took part in the development of Getaviz and they did not use Getaviz in practice. Occasionally, Getaviz was made available to students, so that they could use it and extend it as part of their studies. Triggered by the publication of our tool at the most important conference in our research field [Ba17], we decided to release Getaviz as open source in this year. Doing this we pursued the following objectives:

1. Improve visibility of Getaviz and of our work in general within the scientific community
2. Simplify future collaboration with industry and other research groups
3. Facilitate reproducibility of empirical studies

However, there were also concerns about increased maintenance efforts, too many feature requests, and outflow of ideas. We assumed that it is hard for projects like these to build a large and lively community. So we did not expect any unpaid external contributors, i.e., people who contribute to open source in their leisure time since people prefer to contribute to software they use by themselves. We were also aware that Getaviz is a research prototype

[4] https://github.com/softvis-research/Getaviz
[5] http://home.uni-leipzig.de/svis/showcases/

and hence not ready for production and it will not find suddenly hundreds or thousands of users. Nevertheless, we decided to release Getaviz under Apache License 2.0. We also discussed more restrictive and viral licenses such as GNU General Public License 3 (GPL3) and even GNU Affero General Public License to ensure that modifications to the source code will be published under the original license. However, companies often shy away from this type of license. That is why we decided to use the more liberal Apache License 2.0 which is not viral, i.e., anyone can modify the source code without making the changes available again as open source.

Until its release, we had to overcome some obstacles. Since many contributions came from students as part of their theses, it was not clearly regulated how to publish this code. Therefore we had to contact all students being involved formerly and ask for their permission, which was a lot of work. Next, we had to review all used third-party libraries for their licenses. As a consequence, we had to replace some libraries due to license incompatibility, especially between Apache License 2.0 and GPL3. For this reason we have discarded the version history and started with a new and clean repository. Finally, we released Getaviz 1.0 in November 2017 under Apache License 2.0.

In 2019 GISA GmbH[6] and Visual Software Analytics started a joint initiative named "Visual SAP Analytic Process" (VISAP) based on Getaviz. GISA is an IT full service provider with focus on SAP systems for energy market, contracting authorities, and industrial as well as service companies. The goal of VISAP is to improve quality assessment of customer specific SAP code and to support the migration of custom code to the new SAP S/4HANA platform by using software visualizations. As part of VISAP, Getaviz is used to visualize the custom code written in ABAP, the programming language of SAP systems, which is extracted by an own tool for custom code life cycle management. Currently, GISA employs four people who work part-time to develop VISAP and have already provided many new features for Getaviz that will benefit everyone involved. These include numerous feature improvements as well as improvements in robustness and maintainability, with only a few functions of exclusive interest to GISA. Although the license used, Apache License 2.0, is not viral and GISA is not legally obliged to play back its changes on Getaviz, they do. From GISA's point of view the open source nature of Getaviz is an advantage, because there are no possible lock-in effects and a later usage as a commercial product is also possible.

Since Getaviz is a self-contained system, it is easy to deploy and provide running instances for reviewers and other researchers. We have observed further positive side effects. Previously, the number of installations was very limited and it was possible for us to monitor each installation process personally and provide useful tips. Since this approach does not work for users outside the research group, we had to revise the installation process as well as to improve documentation. Therefore, the quality of our documentation and setup routine has improved significantly since we released Getaviz as open source. The same applies to the code itself. Getaviz has become more robust since it is used on more computers with different

[6] http://www.gisa.de

setups. We also got contributions from external users, but - so far - limited to a manageable extent. In most cases these contributions have been bug reports and improvements of the documentation. We also noticed an increased interest from students in contributing to Getaviz as part of their studies, since they prefer to write code that is actually used.

3 jQAssistant Dashboard

Since 2017, our research group collaborates with BUSCHMAIS GbR[7], an IT consulting company with focus on application integration, deployment, scalability, and persistence. The company has developed *jQAssistant*[8], a quality assurance tool that is based on Neo4j[9]. It scans software artifacts' data, for example source code or test results, stores them as graphs in a Neo4j database, and provides means to analyze the graph data. jQAssistant was released in 2015 under GPL3 and has since formed a significant community.

Through this collaboration we have mutually benefited from each other. On the one hand, BUSCHMAIS was looking for a way to visualize the software data extracted with jQAssistant. On the other hand, our research group needed a unified data source for the visual analytics tools.

Hence, we have developed a dashboard on top of jQAssistant together with a master student. The *jQAssistant Dashboard*[10] supports project leaders and software architects in decision-making. It provides interactive views concerning architecture and dependencies as well as resource, risk, and quality management. Due to our experience with Getaviz, we developed the project as open source from the beginning. Therefore, we had not to consider subsequent license agreements and incompatible libraries and the whole process was much smoother. This successful collaboration resulted in a joint publication at VISSOFT 2018 [Mü18]. Even one year after finishing his master thesis, the student is still maintaining the project. The dashboard will be integrated into regular jQAssistant releases in the near future.

Furthermore, we are developing scanner plugins for jQAssistant licensed under GPL3 to support additional programming languages or data sources. Three plugins are particularly worth mentioning here. The first plugin[11] scans Java source code and was used to solve a feature location challenge at SPLC 2019 [ME19]. The second plugin[12] scans software traces to support application performance monitoring and architecture discovery. It was developed by a student as a seminar project and published at SSP 2019 [MF19]. The third plugin[13] scans data from Jira[14]. It was also developed by a student during Google Summer

[7] https://www.buschmais.de
[8] https://jqassistant.org
[9] https://neo4j.com
[10] https://github.com/softvis-research/jqa-dashboard
[11] https://github.com/softvis-research/jqa-javasrc-plugin
[12] https://github.com/softvis-research/jqa-kieker-plugin
[13] https://github.com/softvis-research/jqa-jira-plugin
[14] https://www.atlassian.com/de/software/jira

of Code (GSoC) 2019. As the jQAssistant Dashboard and the scanner plugins[15] are all open source, we get feature requests, bug reports, and contributions from academia as well as industry and our research can be easily replicated.

4 Open Source Funding

Another advantage of releasing research prototypes as open source is that additional funding sources can be used which exclusively address open source projects, be they scientific or not. Initiatives like GSoC[16], Google Season of Docs (GSoD)[17], and Outreachy[18] support students who would like to start contributing to open source projects. Especially for projects with limited budgets these initiatives are really attractive and a helpful complement to basic funding. However, the application process is different from conventional third-party funding and in our experience the organizational structure of universities is not designed for that. This is because of legal issues, e.g., transfer of rights, but also because of inexperience of the judicial department with US laws in general, which made it impossible for us to participate as Leipzig University in GSoC. To overcome these bureaucratic hurdles we applied for GSoC and GSoD under the umbrella of the InfAI and got accepted for GSoC in 2019. This means that we were able to award paid scholarships to students to further develop our software. Many students from all over the world were interested in our scholarships and applied for it. This is remarkable for such a small project given that our competitors were well-known open source projects with large communities, such as Linux Foundation, KDE, Mozilla, and many others. In our estimation, it is because Getaviz consists of a broad and modern technological basis and the fact that people find software visualizations in general fascinating. We have awarded two scholarships to international master students. Both worked independently for several months on a project based on Getaviz or jQAssistant. One student successfully completed his project and continued to contribute code afterwards. However, studies show that most students take part in GSoC for work experience, career building, and the stipends. Only a minority of the participants will actually become active contributors afterwards [Si19]. Therefore, GSoC only contributes to the building of an active and lively community to a limited extent. But even then open source funding and the contributions so far are very valuable. Taking part in GSoC helped us to increase visibility, to improve our documentation, and to increase the number of external contributions. And last but not least, students also benefit from participating in such programs.

5 Conclusions

For our research group it has been very beneficial to release our software as open source and we regret to have not done this earlier. Releasing our software as open source helped

[15] https://softvis-research.github.io/jqassistant-plugins
[16] https://summerofcode.withgoogle.com
[17] https://developers.google.com/season-of-docs
[18] https://www.outreachy.org

us to increase our visiblity, to provide reproducible software artifacts, and to collaborate with industry. We were even able to expand our funding. Noteworthy, we were not able to identify any negative effects so far.

Nevertheless, we have noticed that the process has its pitfalls and that many decisions have to be taken. Therefore we would like to give the following recommendations.

1. Consider open source release as early as possible in the research process.
2. External transfer institutes can be helpful to overcome limitations of organizational structures of universities and provide additional funding options.
3. Expect additional effort for documentation, code reviews, and support.
4. Limit your expectations with regards to community building.

We want to emphasize that these recommendations are based on our experience given our specific scenario and should be applied to other projects with care.

Acknowledgments

We would like to thank all contributors to our open source systems, especially Stefan Bechert, Matteo Fischer, Tino Mewes, Christina Sixtus, and Lisa Vogelsberg for their valuable and significant contributions. We also thank Ulrich Eisenecker for his comments that greatly improved the manuscript.

References

[Ba17] Baum, David; Schilbach, Jan; Kovacs, Pascal; Eisenecker, Ulrich; Müller, Richard: GETAVIZ: Generating Structural, Behavioral, and Evolutionary Views of Software Systems for Empirical Evaluation. In: IEEE VISSOFT. 2017.

[ME19] Müller, Richard; Eisenecker, Ulrich: A Graph-based Feature Location Approach Using Set Theory: [Challenge Solution]. In: Proc. 23rd Int. Syst. Softw. Prod. Line Conf. - Vol. A. SPLC '19, ACM, New York, NY, USA, pp. 161–165, 2019.

[MF19] Müller, Richard; Fischer, Matteo: Graph-Based Analysis and Visualization of Software Traces. In: 10th Symp. Softw. Perform. Jt. Dev. Community Meet. Descartes/Kieker/Palladio. Würzburg, Germany, 2019.

[Mü18] Müller, Richard; Mahler, Dirk; Hunger, Michael; Nerche, Jens; Harrer, Markus: Towards an Open Source Stack to Create a Unified Data Source for Software Analysis and Visualization. In: Proc. 6th IEEE Work. Conf. Softw. Vis. IEEE, Madrid, Spain, 2018.

[Si19] Silva, Jefferson O; Wiese, Igor; German, Daniel M; Treude, Christoph; Gerosa, Marco A; Steinmacher, Igor: Google Summer of Code: Student Motivations and Contributions. 2019.