Proceedings of the
12th International Conference on Applied Informatics
Eger, Hungary, 2023

Reinterpretation of FAIR Guidelines for Program Slicing Tools

Péter Attila Soha[a]

[a]University of Szeged,
psoha@inf.u-szeged.hu

Abstract. Program Slicing is one of the most important uses of source code analysis. Over the years, different variations of it have been implemented in many tools, depending on the goal to be achieved. However, what these programs have in common is that they are not intended for publication, and their subsequent use may be problematic. In this paper, we present a set of principles that builds on the FAIR guidelines and describes the basics and common criteria for publishing research software in a more accurate and informative way.

Keywords: Program Slicing, FAIR, Slicer Tools

AMS Subject Classification: ***D*.2.12 Software and its engineering** — *Interoperability*: ***D*.2.13 Software and its engineering** — *Reusable Software*

1. Introduction

Since the *Program Slicing* method had been originally published [8], it was interpreted and implemented in many different ways to solve various problems. The **FAIR** guidelines were published in 2016 by Mark Wilkinson et al. [9] with the aim of providing a unified framework for the publication of scientific databases. The FAIR principles mean that data and algorithms should be accessible, transparent, and reproducible to ensure equal access and responsible use. They are intended to help ensure that data are not only used by scientists but are also accessible to the general public and can be applied in scientific research and practice. FAIR defines four main principles covering the main criteria for data, namely **Findability**, **Accessibility**, **Interoperability**, and **Reusability**. However, the main drawback of the original guidelines was that it was made for data, not for programs, which (although can be considered as data) require a different point of view. This

recognition led Anna-Lena Lamprecht et al. in 2019 [7] and in 2022 [4] to modify the original principles to cover research software more precisely. In this paper, we present a FAIR interpretation that better separates the relevant properties of the tool itself from those that define the data that describe it, thus overcoming the drawback of the previous approaches about falsely failing principles.

2. Reinterpretation of the FAIR Guidelines

The main objection to the previous interpretations is that they follow the original structure too closely, so that in the event of an evaluation, the software may perform poorly even if it meets the requirements that directly affect its use. An example of this are the *R1-R1.3* points in the approach described in [7], where the declared criteria must be satisfied for both the software and its associated metadata. If 1 published tool has incomplete metadata, it cannot satisfy these points (for example, the GitHub interface does not allow us to edit the metadata of our software). From this point of view, our approach is based on clustering that separates metadata requirements from software-related ones. The two categories are labelled **Availability** and **Content**. In the former, we defined three subcategories covering *identifiers*, *metadata*, and *protocols* used for access, while under Content, we included properties such as *attachments*, *version control*, and *documentation*.

3. Results

To prove the usability of our approach, we focused on program slicing applications and evaluated them with the new method. In this case, we have evaluated three popular slicers that are currently in use. The first was *JavaSlicer* [3], which is a static slicer for the Java language, especially version 6. The second choice was also a static slicer, *JavaSDGSlicer* [6][5], which, however, supports newer language versions. Finally, the third choice was a dynamic slicer tool, namely *Slicer4J* [2][1]. The advantage of evaluating the approach is that all three tools are published on GitHub and have publications, except for JavaSlicer, which simplifies the process of testing the Availability. The existence of a repository automatically provides unique identification and a full version history, as well as independent metadata management. Positive results were found in these areas. However, gaps were found in the analysis of the Content, such as replication packages, external dependencies, or user documentation, which are not mandatory elements of a GitHub repository. For example, in the case of JavaSlicer, there were serious obstacles due to incomplete or outdated dependencies.
To summarize the results, our approach gives a more accurate picture of the real-world usability of a research tool than previously published solutions on this topic. We have achieved this by separating the principles that directly describe usability from those that define accessibility.

4. Conclusions and Future Work

In this article, we have examined a new approach to the FAIR guidelines, which aims to separate the principles that describe the availability of software from those that describe the content, giving a more accurate picture of usability. For the analysis we employed widely used software slicing tools, freely available on GitHub. The aim of this publication is to lay the foundations for a more extensive evaluation involving more software and for a more detailed guideline to achieve more precise results.

References

[1] K. Ahmed, M. Lis, J. Rubin: *resess/Slicer4J: Slicer4J is an accurate, low-overhead dynamic slicer for Java programs.* Accessed: 2022-09-18, URL: https://github.com/resess/Slicer4J.

[2] K. Ahmed, M. Lis, J. Rubin: *Slicer4J: A Dynamic Slicer for Java*, in: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2021, Athens, Greece: Association for Computing Machinery, 2021, pp. 1570–1574, ISBN: 9781450385626, DOI: 10.1145/3468264.3473123.

[3] C. Backes: *Backes/javaslicer: JavaSlicer is an open-source dynamic slicing tool developed at Saarland University*, Accessed: 2022-09-18, URL: https://github.com/backes/javaslicer.

[4] M. Barker, N. P. Chue Hong, D. S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, et al.: *Introducing the FAIR Principles for research software*, Scientific Data 9.1 (2022), pp. 1–6, DOI: https://doi.org/10.1038/s41597-022-01710-x.

[5] C. Galindo: *mistupv/JavaSlicer: A program slicer for Java, based on the system dependence graph (SDG).* Accessed: 2022-09-18, URL: https://github.com/mistupv/JavaSlicer.

[6] C. Galindo, S. Pérez, J. Silva: *Slicing Unconditional Jumps with Unnecessary Control Dependencies*, in: Logic-Based Program Synthesis and Transformation: 30th International Symposium, LOPSTR 2020, Bologna, Italy, September 7–9, 2020, Proceedings, Bologna, Italy: Springer-Verlag, 2020, pp. 293–308, ISBN: 978-3-030-68445-7, DOI: 10.1007/978-3-030-68446-4_15.

[7] A.-L. Lamprecht, L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. M. del Pico, V. D. D. Angel, S. van de Sandt, J. C. Ison, P. A. Martínez, P. McQuilton, A. Valencia, J. L. Harrow, F. Psomopoulos, J. L. Gelpi, N. P. C. Hong, C. A. Goble, S. Capella-Gutiérrez: *Towards FAIR principles for research software*, Data Sci. 3 (2020), pp. 37–59, DOI: DOI10.3233/DS-190026.

[8] M. Weiser: *Program Slicing*, IEEE Transactions on Software Engineering SE-10.4 (1984), pp. 352–357, DOI: 10.1109/TSE.1984.5010248.

[9] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al.: *The FAIR Guiding Principles for scientific data management and stewardship*, Scientific data 3.1 (2016), pp. 1–9, DOI: 10.1038/sdata.2016.18.