A Multi-Modal Community Quest for Open Data and Reproducibility in Software Engineering

Abstract

Various research disciplines face their unique challenges in handling research data and making their research reproducible. In this talk, we present open data and reproducibility challenges, like trade secrets, copyright, and paywalls in the domain of software engineering, specifically its modeling subcommunity and the community's initiatives to overcome them.

Introduction

Research data is not at all monotonous. Across various disciplines of science, research data offers various peculiarities, challenges, and opportunities. One well-known challenge, for instance, is the proper anonymization of sensitive personal data in medical research; for archival purposes, but also to write about in a published article (van Ooijen, 2019). Our research discipline, software engineering research, offers several challenges that we view as quite rare and in their combination even unique and thus interesting to discuss. In fact, some of our challenges like trade secrets or copyright are described as "rabbit holes" for software and data curation with hurdles that should be avoided, for general solutions (Rios, p. 239, 2018). Nevertheless, our research discipline exists and is facing these challenges. Our community is thus incentivized to tackle these hurdles, and our talk will give an overview of our challenges and our efforts to overcome them.

Software engineering (Mall, 2018) is a discipline that aims to improve the process of creating software. As such, in software engineering research, often, the research data that is investigated is software (hereafter referred to as "programs"). In addition, the research software that analyzes programs is often highly specialized or unique (Rios et al., 2020). This means, that a reproduction package[1] of an article in software engineering usually contains the research software, the programs/data to be analyzed and documentation on how to reproduce the findings, cf. Figure 1.

However, in our specific subdomain of Simulink research, we encounter additional challenges that complicate conducting research or the creation and sharing of such reproduction packages. These challenges arise particularly due to the sensitivity of the data, licensing, and the legal constraints around sharing. Below, we outline the key challenges and problems (CPs) that we face in this context.

[1] Supplementary materials needed to reproduce a study's findings.

The 19th International Digital Curation Conference takes place on 17-19 February 2025 in The Hague, Netherlands

URL: https://www.dcc.ac.uk/events/idcc25

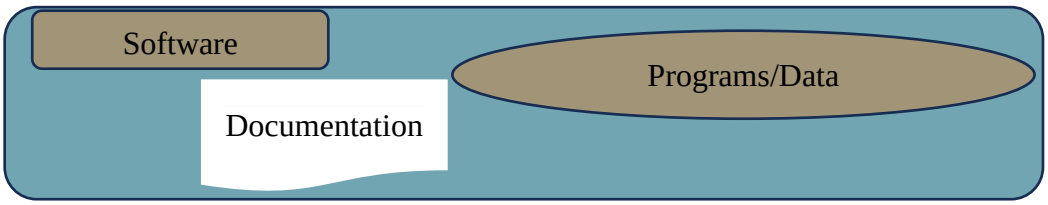

Figure 1 Standard elements of a software engineering reproduction package. We extend an adapted Figure from Rios (Rios, p. 241, 2018).

(CP1) The most interesting research data are "real-world" programs, that are owned by companies. The companies consider their programs to be sensitive intellectual property and usually do not share their programs with researchers or do so only under heavy restrictions. Consequently, researchers are left without necessary research subjects and cannot start their research project or are, e.g., not allowed to publish their data.

(CP2) Additionally, many alternative public programs, e.g., on the sharing website GitHub, are hard to locate or were created without a copyright license or a restrictive one. Such programs qualify as research subjects but publishing them in parts or as a whole is illegal.

(CP3) Simulink, the environment to execute the research software or programs, is proprietary, i.e., not freely available. For reproduction, i.e., when programs need to be executed, an expensive paywall needs to be overcome.

(CP4) Programs are data. While studying their static behavior they behave like normal data. However, when investigated dynamically (e.g., to study program correctness or execution time), the program is run and exhibits behavior that might be unsafe[2] in addition to other complex behavior.

(CP5) The (research) software is often highly specialized, or even unique. It thus needs to be included in the reproduction package and be properly documented – better yet the reproduction package workflow should be automated.

Our subcommunity particular focuses on (CP1), (CP2), (CP3), while the wider computer science community addresses (CP4), (CP5) (Barr et al., 2023; Goedicke & Lucke, 2022). The wider computer science community focuses less on open data (CP1), (CP2), and necessary software for reproduction (CP3) as they are more commonly accessible and free-to-use, for them.

Overall, the efforts, described in our talk help researchers to discover, acquire, freely study, publish, and archive programs or adequate alternatives as research subjects (CP1), (CP2), and to bypass reproduction paywalls (CP3). This reduces the necessary upfront investment for researchers or is the necessary precondition that makes research and its reproducibility possible in the first place.

Background: Simulink and Simulink Research

Simulink is a graphical programming language, often used for programs in engineering contexts, e.g., to control complex hardware in domains such as automotive (Palli et al., 2022), quadcopters (Yasar & Karakose, 2022), and energy (Badi et al., 2021). An

[2] Unchecked programs might crash the execution environment (computer) or may be outright malware, like computer viruses.

exemplary Simulink program is shown in Figure 2. Simulink programs consist of signal lines, that connect blocks. Signal lines transport values from block outputs to block inputs, while blocks compute an output from their input values. This enables complex, dynamic behavior.

In the Simulink integrated development environment, a developer can view the graphical representation of the program, simulate its behavior, and can translate it into a traditional, textual programming language and then deploy it, e.g., onto a chip of a car.

Research around Simulink focuses on the following three directions (Ds). We list their associated challenges and problems in brackets.

D1. Develop new Simulink programs to solve an engineering task (Norouzi et al., 2017). (CP3), (CP5)

D2. Develop a new software tool that helps Simulink developers and validate it on Simulink programs, e.g., a program checker (Nejati et al., 2019). (CP1), (CP2), (CP3), (CP5)

D3. Study Simulink programs themselves: statically, dynamically or the program evolution (Shrestha et al., 2023). (CP1), (CP2), (CP3), (CP4), (CP5)

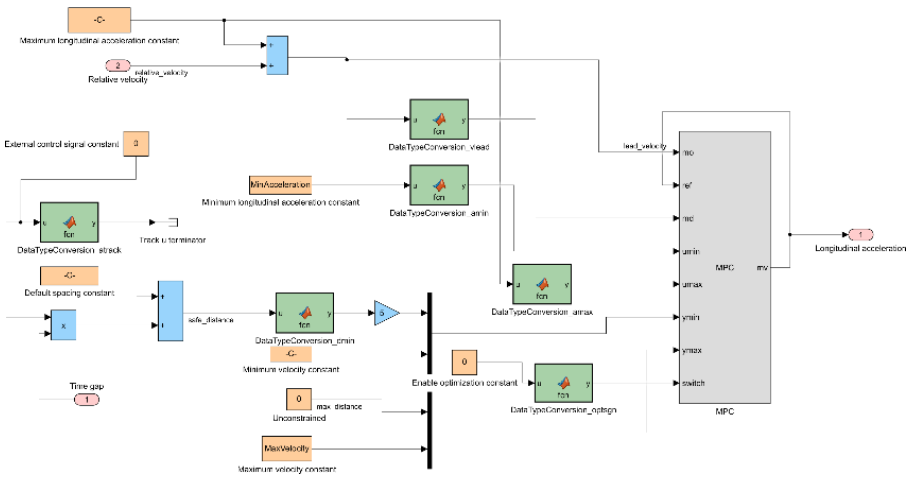

Figure 2: Adaption of a Simulink program of self-driving cars.

A Reproduction Crisis in the Community

The first step in tackling any problem is recognizing that it exists. Many Simulink researchers commented parenthetically on problems of acquiring the necessary programs for their research (CP1) or to publish them (CP1), (CP2) (Bertram et al., 2017; Chowdhury, 2018; Chowdhury et al., 2018; Hussain et al., 2019; Jiang et al., 2017; Rao et al., 2017; Tomita et al., 2019) – be it for the evaluation of their tool (D2) or for studying the programs themselves (D3). The full extent of this problem was only revealed in a systematic literature study (Boll et al., 2022). This study found a reproducibility crisis in empirical Simulink research: only 22% of the programs used as experimental subjects in Simulink studies were accessible in the reproduction package. Equally alarming was the fact, that only 31% of the software tools that were developed in these studies were shared. Combining these findings, only 9% of the studies provided both programs and tools, which is a necessary precondition for reproducibility, cf. Figure 1. Due to missing documentation

or a too cumbersome experimentation workflow, *none* of the investigated Simulink studies could be reproduced, fully.

Community Initiatives for Open Data & Reproducibility

When our community realized the extent of the reproduction crisis described in the prior section, we saw the necessity to act and developed multiple initiatives to mitigate (CP1), (CP2), (CP3):

(CP1) Companies don't share their programs to protect intellectual property (Boll et al., 2024) developed an obfuscator that anonymizes programs, selectively removing form or function of them, while keeping the program structure intact. Companies can now share anonymized programs, with valuable intellectual property removed. An anonymized version of the program of Figure 2 is depicted in Figure 3. We view this as a surprising parallel to anonymization of, e.g., personalized data. In both cases, parts of the data are removed, while keeping other parts intact that still have value.

Note: most initiatives mitigating (CP2) are also helping with (CP1).

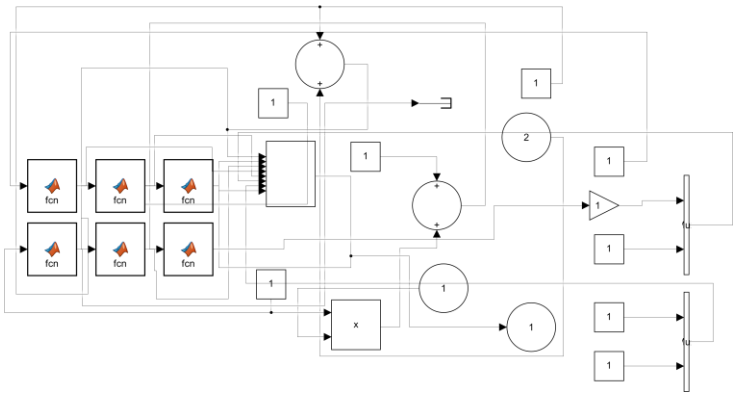

Figure 3 The program from Figure 2 after anonymization. Names and colors are removed, all positions of blocks are scrambled, while the overall structure is preserved. There are also additional invisible changes removing functionality.

(CP2) Public programs are hard to locate and often without permissive license The community started to curate larger and larger corpora of open-source programs from various public sources (Chowdhury, 2018; Chowdhury et al., 2018; Shrestha et al., 2022; Shrestha et al., 2023). These corpora were found to contain many Simulink programs that are adequate for empirical research (Boll et al., 2021). They thus offer a viable alternative, for many research methods, that would otherwise require confidential (non-shared) programs.

To facilitate the discovery of adequate programs within these corpora, ScoutSL, a search engine was established, to find programs based on general and Simulink-specific user criteria (Shrestha et al., 2023).

Additionally, several program mutators exist now, that can amplify useful research subjects by creating close variants of it (Bourbouh et al., 2020; Ceylan et al., 2023).

Using the curated corpora as a learning basis, programs can also be synthesized automatically (Chowdhury et al., 2018; Shrestha & Csallner, 2021). Currently, these artificial programs are quite small, though. However, since the advent of Large Language

Models, new AI assistants are trained on public corpora for semi-automatic program synthesis (Adhikari, 2021; Adhikari et al., 2024; Tinnes et al., 2024).

(CP3) The execution environment, Simulink, is behind an expensive paywall
There are two initiatives, here: 1. Translate Simulink programs into another programming language with a free execution environment, where Simulink is not needed, anymore (Meenakshi et al., 2006; Minopoli & Frehse, 2016; Sanchez et al., 2019). 2. The vendor of Simulink, MathWorks, started an offer that provides basic access to Simulink for free.[3] Some reproduction packages are executable, using this offer.

Success of initiatives
While there is some evidence of an obfuscator being used for publication (Jaskolka et al., 2020; Pantelic et al., 2018) (CP1), overall, evidence for the utilization of initiatives against (CP1) and (CP3) are rare.

Researchers often clearly state the origin of their research data for empirical evaluation, though. Apart from the usage of corpora for the above-mentioned synthesizers and mutators, we found many articles that used one of the curated corpora for their empirical study (Amorim et al., 2023; Boll et al., 2024; Schultheiß et al., 2023; Shrestha et al. 2023; Su et al., 2024). Furthermore, ScoutSL, the search engine for Simulink programs is visited by 50 people, daily, discovering adequate research subjects (CP2).

Please note, that the initiatives are just mitigatory and do not solve all challenges, yet. Especially (CP1) remains a hard challenge, still.

Conclusion

Our talk presented several interesting challenges for open data and reproducibility of our field. These challenges are mitigated by various initiatives from within our community. While these initiatives are not solving the challenges completely, our community is now in a much better position for a more open and reproducible research.

Our talk's discussion will focus on possible parallels of the presented techniques to other research areas. Those research disciplines facing similar challenges may also anonymize or mutate existing data, curate open alternatives, or synthesize adequate substitutes. On the other hand, we also seek feedback on what opportunities our community can learn from other communities and their ideas.

Acknowledgements

This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project number NFDI 52/1 501930651.

We used ChatGPT for better phrasing and to correct grammar mistakes.

[3] Certain restrictions apply for free users, such as time and storage space contingents.

Bibliography

Adhikari, B. (2021). *Intelligent Simulink Modeling Assistance via Model Clones and Machine Learning* [Masters Thesis]. Miami University.

Adhikari, B., Rapos, E. J., & Stephan, M. (2024). SimIMA: a virtual Simulink intelligent modeling assistant: Simulink intelligent modeling assistance through machine learning and model clones. *Software and Systems Modeling*, *23*(1). https://doi.org/10.1007/s10270-023-01093-6

Amorim, T., Boll, A., Bachman, F., Kehrer, T., Vogelsang, A., & Pohlheim, H. (2023). Simulink bus usage in practice: an empirical study. *Journal of Object Technology*, *22*(2). https://doi.org/10.5381/jot.2023.22.2.a12

Barr, E., Timperley, C., Bell, J., Hilton, M., & Mechtaev, S. (2023). Continuously Accelerating Research. *Proceedings - International Conference on Software Engineering*. https://doi.org/10.1109/ICSE-NIER58687.2023.00028

Bertram, V., Maoz, S., Ringert, J. O., Rumpe, B., & Von Wenckstern, M. (2017). Component and Connector Views in Practice: An Experience Report. *Proceedings - ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems, MODELS 2017*. https://doi.org/10.1109/MODELS.2017.29

Boll, A., Brokhausen, F., Amorim, T., Kehrer, T., & Vogelsang, A. (2021). Characteristics, potentials, and limitations of open-source Simulink projects for empirical research. *Software and Systems Modeling*, *20*(6). https://doi.org/10.1007/s10270-021-00883-0

Boll, A., Kehrer, T., & Goedicke, M. (2024). SMOKE: Simulink Model Obfuscator Keeping Structure. *MODELS (Accepted)*.

Boll, A., Rani, P., Schultheiß, A., & Kehrer, T. (2024). Beyond code: Is there a difference between comments in visual and textual languages? *Journal of Systems and Software*, *215*, 112087. https://doi.org/10.1016/j.jss.2024.112087

Boll, A., Vieregg, N., & Kehrer, T. (2022). Replicability of experimental tool evaluations in model-based software and systems engineering with MATLAB/Simulink. *Innovations in Systems and Software Engineering*. https://doi.org/10.1007/s11334-022-00442-w

Chowdhury, S. A. (2018). Understanding and improving cyber-physical system models and development tools. *Proceedings - International Conference on Software Engineering*. https://doi.org/10.1145/3183440.3183455

Chowdhury, S. A., Mohian, S., Mehra, S., Gawsane, S., Johnson, T. T., & Csallner, C. (2018). Automatically finding bugs in a commercial cyber-physical system development tool chain with SLforge. *Proceedings - International Conference on Software Engineering*. https://doi.org/10.1145/3180155.3180231

Chowdhury, S. A., Varghese, L. S., Mohian, S., Johnson, T. T., & Csallner, C. (2018). A curated corpus of simulink models for model-based empirical studies. *Proceedings - International Conference on Software Engineering*. https://doi.org/10.1145/3196478.3196484

Goedicke, M., & Lucke, U. (2022). Research Data Management in Computer Science - the NFDIxCS Approach. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft Fur Informatik (GI), P-326*. https://doi.org/10.18420/inf2022_112

Hussain, A., Sher, H. A., Murtaza, A. F., & Al-Haddad, K. (2019). Improved restricted control set model predictive control (iRCS-MPC) based maximum power point

tracking of photovoltaic module. *IEEE Access*, *7*. https://doi.org/10.1109/ACCESS.2019.2946747

Jaskolka, M., Pantelic, V., Wassyng, A., & Lawford, M. (2020). *Supporting Modularity in Simulink Models*.

Jiang, Z., Wu, X., Dong, Z., & Mu, M. (2017). Optimal Test Case Generation for Simulink Models Using Slicing. *Proceedings - 2017 IEEE International Conference on Software Quality, Reliability and Security Companion, QRS-C 2017*. https://doi.org/10.1109/QRS-C.2017.67

Mall, R. (2018). Fundamentals of Software Engineering, Fifth Edition. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Meenakshi, B., Bhatnagar, A., & Roy, S. (2006). Tool for translating simulink models into input language of a model checker. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4260 LNCS*. https://doi.org/10.1007/11901433_33

Minopoli, S., & Frehse, G. (2016). SL2SX translator: From simulink to SpaceEx models. *HSCC 2016 - Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. https://doi.org/10.1145/28838172883826

Nejati, S., Gaaloul, K., Menghi, C., Briand, L. C., Foster, S., & Wolfe, D. (2019). Evaluating model testing and model checking for finding requirements violations in Simulink models. *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. https://doi.org/10.1145/3338906.3340444

Norouzi, P., Kivanç, Ö. C., & Üstün, Ö. (2017). High performance position control of double sided air core linear brushless DC motor. *2017 10th International Conference on Electrical and Electronics Engineering, ELECO 2017*, *2018-January*.

Pantelic, V., Postma, S., Lawford, M., Jaskolka, M., Mackenzie, B., Korobkine, A., Bender, M., Ong, J., Marks, G., & Wassyng, A. (2018). Software engineering practices and Simulink: bridging the gap. *International Journal on Software Tools for Technology Transfer*, *20*(1). https://doi.org/10.1007/s10009-017-0450-9

Rao, A. C., Raouf, A., Dhadyalla, G., & Pasupuleti, V. (2017). Mutation testing based evaluation of formal verification tools. *Proceedings - 4th International Conference on Dependable Systems and Their Applications, DSA 2017*, *2018-January*. https://doi.org/10.1109/DSA.2017.10

Rios, F. (2018). Incorporating Software Curation into Research Data Management Services: Lessons Learned. *International Journal of Digital Curation*, *13*(1). https://doi.org/10.2218/ijdc.v13i1.608

Rios, F., Lassere, M., Ruggill, J. E., & McAllister, K. S. (2020). Sustaining Software Preservation Efforts Through Use and Communities of Practice. *International Journal of Digital Curation*, *15*(1). https://doi.org/10.2218/ijdc.v15i1.696

Sanchez, B. A., Zolotas, A., Hoyos Rodriguez, H., Kolovos, D. S., & Paige, R. F. (2019). On-the-Fly Translation and Execution of OCL-Like Queries on Simulink Models. *Proceedings - 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems, MODELS 2019*. https://doi.org/10.1109/MODELS.2019.000-1

Schultheiß, A., Bittner, P. M., Boll, A., Grunske, L., Thüm, T., & Kehrer, T. (2023). RaQuN: a generic and scalable n-way model matching algorithm. *Software and Systems Modeling*, *22*(5). https://doi.org/10.1007/s10270-022-01062-5

Shrestha, S. L., Boll, A., Chowdhury, S. A., Kehrer, T., & Csallner, C. (2023). EvoSL: A Large Open-Source Corpus of Changes in Simulink Models & Projects. *Proceedings - ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems, MODELS 2023*. https://doi.org/10.1109/MODELS58315.2023.00024

Shrestha, S. L., Boll, A., Kehrer, T., & Csallner, C. (2023). ScoutSL: An Open-Source Simulink Search Engine. *Proceedings - 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS-C 2023*. https://doi.org/10.1109/MODELS-C59198.2023.00022

Shrestha, S. L., Chowdhury, S. A., & Csallner, C. (2022). SLNET: A Redistributable Corpus of 3rd-party Simulink Models. *Proceedings - 2022 Mining Software Repositories Conference, MSR 2022*. https://doi.org/10.1145/3524842.3528001

Shrestha, S. L., Chowdhury, S. A., & Csallner, C. (2023). Replicability Study: Corpora For Understanding Simulink Models & Projects. *International Symposium on Empirical Software Engineering and Measurement*. https://doi.org/10.1109/ESEM56168.2023.10304867

Shrestha, S. L., & Csallner, C. (2021). SLGPT: Using transfer learning to directly generate simulink model files and find bugs in the simulink toolchain. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3463274.3463806

Su, Z., Yu, Z., Wang, D., Yang, Y., Wang, R., Chang, W., Cui, A., & Jiang, Y. (2024). HSTCG: State-Aware Simulink Model Test Case Generation with Heuristic Strategy. *IEEE Transactions on Software Engineering*, 1–17. https://doi.org/10.1109/TSE.2024.3428528

Tinnes, C., Welter, A., & Apel, S. (2024). *Leveraging Large Language Models for Software Model Completion: Results from Industrial and Public Datasets*.

Tomita, T., Ishii, D., Murakami, T., Takeuchi, S., & Aoki, T. (2019). A scalable Monte-Carlo test-case generation tool for large and complex simulink models. *Proceedings - 2019 IEEE/ACM 11th International Workshop on Modelling in Software Engineering, MiSE 2019*. https://doi.org/10.1109/MiSE.2019.00014

van Ooijen, P. M. A. (2019). Quality and curation of medical images and data. In *Artificial Intelligence in Medical Imaging: Opportunities, Applications and Risks*. https://doi.org/10.1007/978-3-319-94878-2_17