ScoutSL: An Open-source Simulink Search Engine

Sohil Lal Shrestha
Computer Science & Eng. Dept.
University of Texas at Arlington
Arlington, TX 76019, USA
sohil.shrestha@mavs.uta.edu

Alexander Boll
Software Engineering Group
University of Bern
3012 Bern, Switzerland
alexander.boll@inf.unibe.ch

Timo Kehrer
Software Engineering Group
University of Bern
3012 Bern, Switzerland
timo.kehrer@inf.unibe.ch

Christoph Csallner
Computer Science & Eng. Dept.
University of Texas at Arlington
Arlington, TX 76019, USA
csallner@uta.edu

***Abstract*—Simulink is one of the most widely used modelling languages in safety-critical industries. Most models created in industrial settings are valuable intellectual property to their companies and are thus often not publicly available. But to study Simulink software engineering processes or to develop new Simulink tools, access to models with relevant properties is vital for researchers. We conducted a community survey to find out what kind of models and model metrics are of interest to researchers. With these results, we implemented ScoutSL (http://scoutsl.net), a tool that gives researchers easy online access to over 100k open-source Simulink models from which they can select a subset according to their needs. A short video demonstration is available online at https://youtu.be/HwsHL8LrVCM**

***Index Terms*—Simulink, search engine, open-source**

I. Introduction

Searching for Simulink models presents challenges due to the absence of a convenient method for finding such models beyond text-based searches. Traditional textual programming language search attributes such as lines of code do not apply to Simulink models, which are developed via graphical block diagrams. So the requisite search attributes for Simulink models are not adequately addressed.

Despite the proliferation of open-source repositories that have accelerated empirical study of code [10], [24], there is a dearth of such studies on MATLAB/Simulink. This can be attributed to the lack of easily accessible model corpora and user-friendly tools that cater to novice users. Researchers have encountered difficulties in discovering third-party Simulink models suitable for utilization in their studies, particularly for stress testing their developed tools or validating the generalizability of novel techniques they are attempting to address. The absence of an easily accessible and user-friendly tool remains the primary hindrance [6], [19], [14], [22], [25], [21], [15].

Popular code hosting platforms such as GitHub and GitLab lack the capability to filter attributes specific to Simulink models, and the identification of Simulink projects is challenging as these platforms do not label projects with Simulink as a programming language. Moreover, utilizing the APIs of these platforms for research purposes is time-consuming due to API rate limits. For instance, GitHub's API restricts authenticated users to 30 search requests per minute, yielding 1k results per request and 5k other requests per hour. Considering that GitHub currently hosts over 330 million repositories, obtaining results, excluding downloading and further analysis, would require at least 330k requests (around 180 hours) [7], [23].

Few existing model-based search tools, like MAR [9], require a deep understanding of the metamodel, while others, such as ModelMine [12], offer a user-friendly search engine but rely on the GitHub API, which inherently imposes limitations on the number of search results it can retrieve. Furthermore, GitHub is not the sole source of Simulink projects. Simulink vendor MathWorks also provides a platform, which serves as a repository for community-developed projects.

Recent efforts on developing large collections of Simulink models have focused on carefully curating corpora of Simulink models manually [4] and later automatically [20] and maintaining metadata of commonly used attributes. Such corpora are either maintained in non-permanent locations though or packaged as a single non-divisible set, making them difficult to sample [3], [20]. Downloading a large corpus to sample a small subset of models is also often inconvenient.

To gain insight into attributes of Simulink models that are of interest to the research community, we conducted a survey involving researchers. The survey confirmed their struggles in getting suitable (e.g., size, publishable) models for their research as seen in Figure 2. Consequently, we developed a web-based search tool that allows users to easily sample models. Our tool, ScoutSL, expands upon the existing SLNET [20] and EvoSL [16] infrastructure to extract Simulink project attributes, collect model metrics and compute derived metrics. The tool offers advanced fine-grained filtering attributes, enabling users to efficiently sample desired models. We have indexed over 18k projects containing more than 100k Simulink models. To the best of our knowledge, ScoutSL is the first tool specifically designed for searching Simulink projects and models, offering filtering attributes not available through other search engines. To summarize, the paper makes the following major contributions.

- Our survey results show that researchers often struggle to get relevant models for their research and would likely benefit from a Simulink search engine.
- We developed a Simulink search engine deployed in a tool called ScoutSL whose search interface and ranking scheme are based on survey responses.
- The tool and all artifacts are open-source [17], [18].

- The search engine is accessible through its web component available online at http://scoutsl.net.

II. Background: Simulink, SLNET, and EvoSL

Simulink is a cyber-physical system (CPS) design and simulation tool that is a de-facto standard in many safety-critical industries. Engineers design a CPS as a model that contains interconnected blocks, where each block may accept data, perform some operation on the data, and transmit its output to other blocks, as depicted in Figure 1. Simulink provides an extensive library of blocks and toolboxes to design and simulate complex multi-domain systems.

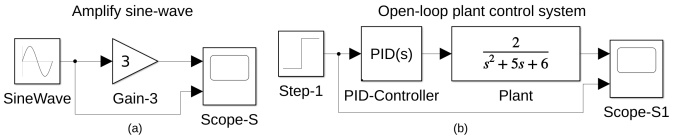

Fig. 1: Two tiny example Simulink models.

To enable empirical studies on Simulink models, researchers have curated large corpora of open-source Simulink models. The most extensive corpus available to date is SLNET [20], which contains Simulink models from two popular hosting sites. SLNET has 3k Simulink projects with their 8k Simulink models (excluding library and test harnesses), collectively featuring over 1M blocks. Boll et al. [1] confirmed large open source corpora to be suitable for empirical research. SLNET is complemented by mining and metric tools. However, as SLNET primarily consists of Simulink model snapshots, it does not support evolution studies.

To address this issue, EvoSL [16] extended SLNET-Miner and curated Simulink repositories from GitHub. EvoSL consists of 924 projects with over 140k default-branch commits. SLNET and EvoSL are self-contained and redistributable.

III. Survey of Simulink Users

We aimed to assess the potential need for a Simulink model search engine by asking Simulink researchers. We then used the survey results to develop ScoutSL.

From a literature review of Boll et al. [2] we extracted 215 academic papers' co-authors that report on Simulink tools and their empirical evaluation. In July and August 2022 we invited them to our Google Forms based anonymized online survey. 16 researchers participated in our survey, from which we discarded one participant (who responded "not applicable" to every question), leaving 15 participants. While all questions and responses are available online [18], we provide a brief summary of the questions and responses in the sequel.

In our first question of the survey, we asked the researchers, "what is the purpose of Simulink models in your research?" Of 15 participants, six reported that their main use-case for Simulink models was tool evaluation. Other use-cases like scalability evaluation, performance evaluation, model co-simulation, industrial process modelling, prototyping, test automation, testing, verification, code generation optimization, compilation, model deployment, and replication were mentioned by one participant each. All the following questions and their detailed results are shown in Figures 2 to 7.

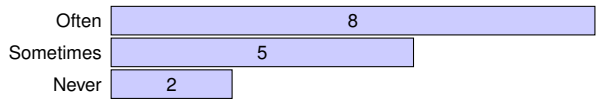

Fig. 2: Responses to "Do you have difficulties finding adequate Simulink models or projects for your research?"

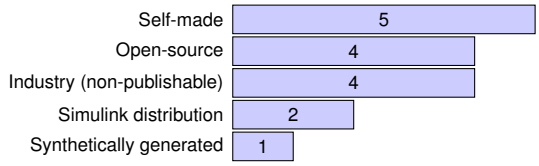

Fig. 3: Responses to "Where do you usually obtain your Simulink artifacts from?"

Over 85% of participants (13/15) faced difficulties finding appropriate models for their research projects (cf. Figure 2). When it came to acquiring models, one third of the participants created their own Simulink artifacts, followed by using closed-source and open-source models, see Figure 3. Figure 4 illustrates that the majority of participants said they require 20 or fewer models for their research—the unconventional ranges follow the responses. Figure 5 breaks down the model metrics of interest reported by the participants.

In response to our questions regarding the adoption of open-source Simulink models, participants showed overwhelming support for both potential usage of the dataset (cf. Figure 6) and the need for a search engine (cf. Figure 7).

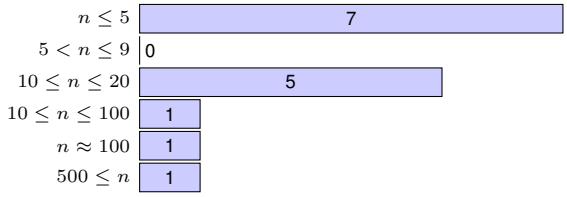

Fig. 4: "How many models would you need for your typical research project?"

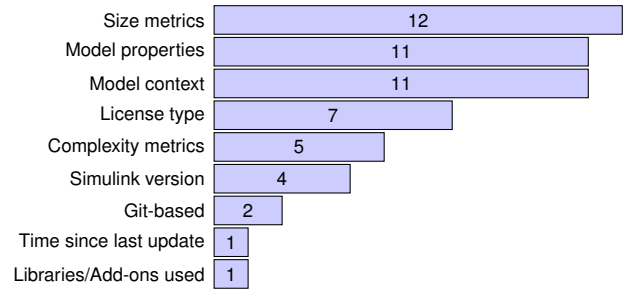

Fig. 5: Responses to "What are Simulink model metrics that are relevant for your research?"

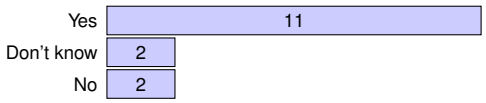

Fig. 6: Responses to "We collected 9,117 open-source models from GitHub. Intuitively, do you think this collection can provide you with suitable Simulink models for your research?"

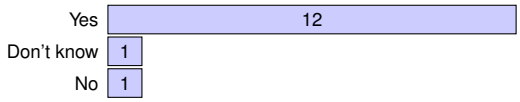

Fig. 7: Responses to "Would you use ScoutSL for your research, in the future?"

IV. Tool Architecture

Figure 8 illustrates ScoutSL's architecture, which consists of two main components: An (offline) mining component and an (online) web application component. The miner retrieves Simulink projects from the repository hosting sites and stores project, model, and commit metrics in a SQLite database.

An intermediate component queries the SQLite database, computes derived attributes such as a model's code generation capability, and calculates a "relevance" project score. A subset[1] of the primary and derived attributes are then stored in a cloud-hosted NoSQL database, as NoSQL databases typically have flexible data models and scale horizontally. The online web interface of ScoutSL facilitates user searches for Simulink projects, allowing filtering based on various attributes.

V. Mining Component

To mine from GitHub and MATLAB Central we use the existing SLNET [20] and EvoSL [16] infrastructure. Unlike SLNET or EvoSL, our focus is primarily on curating a comprehensive database of publicly available Simulink projects, and thus we do not prioritize the analysis of license files as the goal is to allow users to sample from all available open-source Simulink models. Our search yielded 18k projects comprising 109k Simulink models having 15M+ blocks ($\geq 15\times$ SLNET).

A. GitHub

Our extension of SLNET-Miner efficiently manages GitHub's API rate limits. Initially, we query for projects created within a specific time frame, such as *"q=simulink&created:2008-01-01..2009-01-01"*. To exhaustively search for projects using the GitHub API, we employ a divide-and-conquer strategy when the query returns 1k results. We split the time interval in half until the number of results returned is less than 1k.

From the query results we then iteratively download each project and check if it contains a Simulink model by checking for files with MDL or SLX extensions. We extract 80 attributes [18] from the projects' metadata, commits, issues, and pull requests. We only include each GitHub project's commits from its default-branch. Over a one-month period we thereby downloaded some 14k Simulink projects. To mitigate redundancy we currently do not collect metrics from forked projects (but plan to add this in the future).

[1]Due to unclear project licenses not all SQLite data are exposed.

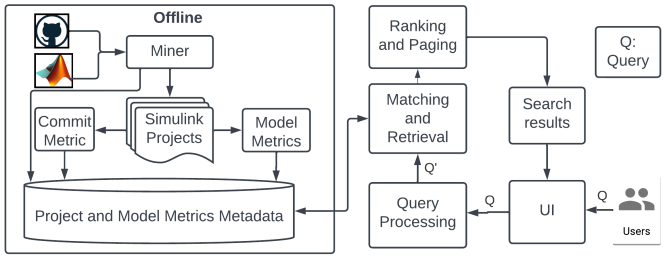

Fig. 8: Architecture of the ScoutSL tool.

B. MATLAB Central

As SLNET we parse MATLAB Central's RSS feed (which does not impose any parsing restrictions), but the feed does not offer a structured method for downloading projects. We extended SLNET-Miner to enhance its heuristic for constructing download links for MATLAB Central projects. Despite these improvements, 7.9k of 46k MATLAB Central projects remained inaccessible for automated download.

While the GitHub API exposes a project's license name, MATLAB Central projects are bundled with license files that require further analysis. To automate this process, we utilized an open-source library employed by GitHub [8]. Mining MATLAB Central took about two and a half days and yielded 4.2k projects with 18 attributes [18].

C. Model Metrics

To facilitate searching based on Simulink model metrics, we extended the existing SLNET-Metrics tool to add more metrics, including the presence of a TargetLink blocks, toolbox dependencies, and system target files. Responding to survey responses (which highlighted researchers' interest in filtering models based on block categories), we further enhanced the tool to support the categorization of block types into non-overlapping categories, as employed in our recent work [16]. We analyzed models on MATLAB R2022b and collected 39 model metrics [18] overall.

VI. User Interface

ScoutSL has simple and advanced search. The latter offers three distinct user interfaces: Simulink model search, repository search, and commit search (catering to various dataset research requirements, including model evolution studies and subject models for tool evaluations).

A. Simple Search

In the simple search users enter a text-based query, which ScoutSL matches with the project descriptions in the database. An example search of "turbine" produces 50+ project results. ScoutSL then sorts the results in descending order based on a ranking score. To compute the ranking score, we adapted a

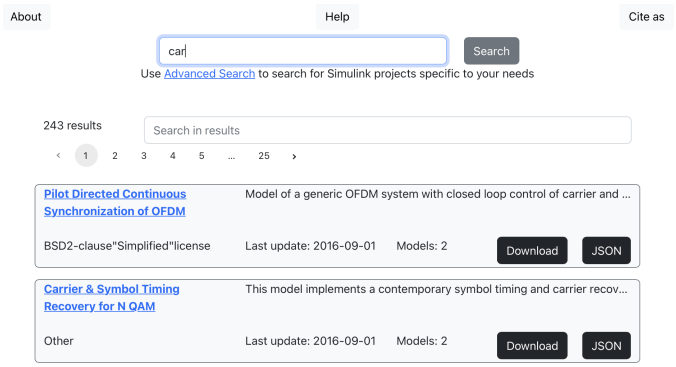

Fig. 9: Example simple Simulink project search for "car".

strategy inspired by previous work [11] that aimed to classify engineered and toy projects. We selected the common Table 1 attributes shared by GitHub and MATLAB Central projects and scored them based on survey responses. To prioritize Git-based attributes highlighted by the survey, we assigned lower weight scores to model revision and model contributors, while MATLAB Central projects received zero for these attributes.

Table 1: Project scoring scheme; CC = cyclomatic complexity.

Attribute	Survey	Weight	Scoring Scheme
Blocks	Size (12)	15	Continuous
Block types	Property (11)	15	Continuous
Code generation	Property (11)	15	Discrete (0,1)
Test harness	Property (11)	15	Discrete (0,1)
Documentation	Property (11)	15	Discrete (0,1)
License	License type (7)	10	Discrete (0,1)
CC	Complexity (5)	5	Continuous
Model revision	Git (2)	2	Continuous
Model contributors	Git (2)	2	Continuous
Toolbox	Add-ons (1)	1	Discrete (0,1)

We scored each attribute either via a binary (0 or 1) or a continuous scheme. For the latter we considered the distribution of data attributes, filtered out outliers, and normalized the scores to between 0 and 1. The final project score was calculated by summing the weighted Table 1 scores. While the scoring scheme is based on our survey responses, we do not make claims regarding the projects' engineering quality [11]. Evaluating and improving the scoring scheme is future work.

B. Advanced Search: Simulink Model

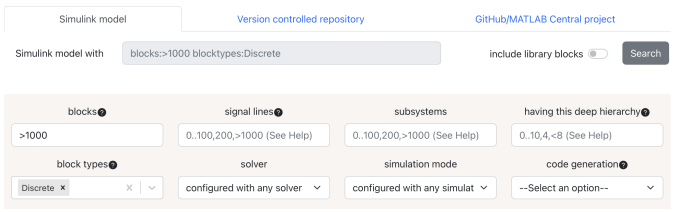

Fig. 10: Example search for Simulink models that contain over 1k blocks, including some discrete blocks.

To cater to the goal of facilitating Simulink model sampling, Figure 10 illustrates ScoutSL's model search UI. The page highlights the specific model metrics that users expressed interest in, as determined through our survey. Users can input numeric values or select attribute options from drop-down menus to refine their search criteria. For example, they can search for Simulink models with over 1k blocks having discrete blocks. This search query generates a list of over 650 projects as search results.

C. Advanced Search: Simulink GitHub Repository

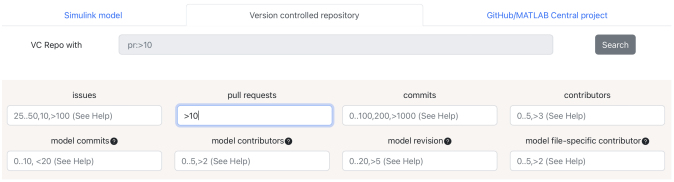

Fig. 11: Example search for Simulink GitHub repositories that have over 10 pull requests.

In order to support studies on model evolution and changes, ScoutSL incorporates GitHub-based selection criteria focused on version control and project management, as seen in Figure 11. Specifically, users can employ search criteria such as the number of project issues, pull requests, commits, and contributors. With our emphasis on Simulink models, users can also search for model-specific commits and contributions, which are subsets of project commits and contributors.

Additionally, ScoutSL enables users to filter projects based on a specific number of model revisions or model files with a certain number of authors. These model-specific search criteria are a unique ScoutSL feature not available on other online web-based tool. While other tools may offer project-level commit information and allow to search for commits per project [12], ScoutSL offers to search over the entire project database. As an example, researchers investigating model development can efficiently identify relevant projects by using a search query that filters for those with a significant number of model revisions. By using a search query for projects with more than 10 model revisions, ScoutSL yields over 500 relevant projects.

D. Advanced Search: Simulink Project

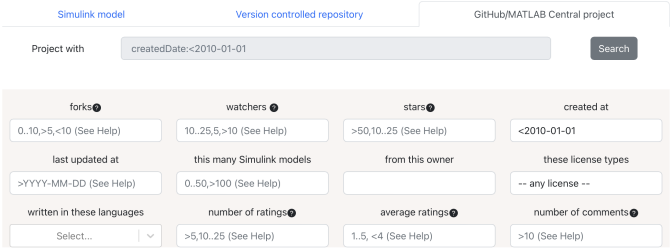

Fig. 12: Example search for pre-2010 Simulink projects.

A commonly employed strategy in mining software repositories for exploratory studies is to sample projects based on popularity metrics. ScoutSL provides the capability to filter

projects based on such metrics, as depicted in Figure 12. Additionally, users are able to query projects created within specific date ranges. Another feature we offer is the ability to filter projects based on license type, as it was identified as a requested feature in the survey responses. Since a Simulink project is often accompanied by complementary scripts written in other programming languages, users can also perform searches involving such criteria. For instance, researchers interested in studying projects Simulink projects with JAVA and C code can use ScoutSL to get 250+ relevant projects. While the most up-to-date project attributes may be available through the primary hosting sites, our database exclusively comprises Simulink projects, which are not easily sampled using existing tools or the primary hosting sites from which we mine our projects.

VII. Related Work

While several tools have been developed to facilitate sampling of open-source projects from platforms like GitHub, they primarily focus on textual programming languages [13], [5]. To obtain model artifacts, a web-based search tool, ModelMine [12], queries GitHub API to narrow down results with file extension. However, the tool mistakenly identifies ".simulink" as a Simulink model file extension and is inherently limited by GitHub API's 1k results per request limit.

A recent study examining forums of modeling tools including MATLAB/Simulink highlighted the potential benefits of model repositories, particularly for novice users who may encounter difficulties when attempting to model something specific [25]. The study emphasizes the importance of establishing and maintaining a diverse repository of example models. To that end, MAR [9] is a web-based search engine that maintains metamodels for various types of models, including UML models. For Simulink, the tool analyzes the precurated corpus to extract their metamodel using a third-party tool. As such, using MAR requires knowledge of modelling languages like EMF to get relevant models, and the search space is limited to 200 Simulink models.

VIII. Conclusions and Future Work

ScoutSL (http://scoutsl.net) is the first search engine geared towards Simulink users' needs. ScoutSL allows searching over 18k Simulink projects containing over 100k Simulink models.

Future works include extension of mining tool to enlarge and augment the dataset with new primary as well as derived project/model attributes such as project domain, Simulink model version. We intend to improve the search engine performance and conduct a thorough evaluation.

Acknowledgements

Christoph Csallner has a potential research conflict of interest due to a financial interest with Microsoft and The Trade Desk. A management plan has been created to preserve objectivity in research in accordance with UTA policy. This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1911017 and a gift from MathWorks.

References

[1] A. Boll, F. Brokhausen, T. Amorim, T. Kehrer, and A. Vogelsang, "Characteristics, potentials, and limitations of open-source simulink projects for empirical research," *Softw. Syst. Model.*, vol. 20, no. 6, pp. 2111–2130, 2021.

[2] A. Boll, N. Vieregg, and T. Kehrer, "Replicability of experimental tool evaluations in model-based software and systems engineering with MATLAB/Simulink," *Innov. Syst. Softw. Eng.*, pp. 1–16, 2022.

[3] S. A. Chowdhury, S. L. Shrestha, T. T. Johnson, and C. Csallner, "SLEMI: Equivalence modulo input (EMI) based mutation of CPS models for finding compiler bugs in Simulink," in *ICSE*. ACM, Jun. 2020, pp. 335–346.

[4] S. A. Chowdhury, L. S. Varghese, S. Mohian, T. T. Johnson, and C. Csallner, "A curated corpus of Simulink models for model-based empirical studies," in *SEsCPS*. ACM, May 2018, pp. 45–48.

[5] O. Dabic, E. Aghajani, and G. Bavota, "Sampling projects in GitHub for MSR studies," in *MSR*. IEEE, May 2021, pp. 560–564.

[6] Y. Dajsuren, M. G. J. van den Brand, A. Serebrenik, and S. A. Roubtsov, "Simulink models are also software: Modularity assessment," in *QoSA*, Jun. 2013, pp. 99–106.

[7] GitHub Inc, "About," 2023, accessed in Mar 2023. [Online]. Available: https://github.com/about

[8] Licensee, "licensee," 2023, accessed in July 2023. [Online]. Available: https://github.com/licensee/licensee

[9] J. A. H. López and J. S. Cuadrado, "An efficient and scalable search engine for models," *Softw. Syst. Model.*, vol. 21, pp. 1715–1737, 2022.

[10] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of UNIX utilities," *Commun. ACM*, vol. 33, no. 12, pp. 32–44, 1990.

[11] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating GitHub for engineered software projects," *Empir. Softw. Eng.*, vol. 22, no. 6, pp. 3219–3253, 2017.

[12] S. M. Reza, O. Badreddin, and R. Khandoker, "ModelMine: A tool to facilitate mining models from open source repositories," in *MODELS*. ACM, Oct. 2020, pp. 9:1–9:5.

[13] S. Romano, M. Caulo, M. Buompastore, L. Guerra, A. Mounsif, M. Telesca, M. T. Baldassarre, and G. Scanniello, "G-repo: A tool to support MSR studies on GitHub," in *SANER*. IEEE, Mar. 2021, pp. 551–555.

[14] S. L. Shrestha, "Automatic generation of Simulink models to find bugs in a cyber-physical system tool chain using deep learning," in *ICSE-C*. ACM, 2020, pp. 110–112.

[15] ——, "Harnessing large language models for simulink toolchain testing and developing diverse open-source corpora of simulink models for metric and evolution analysis," in *ISSTA*. ACM, 2023, pp. 1541–1545.

[16] S. L. Shrestha, A. Boll, S. A. Chowdhury, T. Kehrer, and C. Csallner, "EvoSL: a large open-source corpus of changes in Simulink models & projects," in *MODELS*. IEEE, 2023, to appear.

[17] S. L. Shrestha, A. Boll, T. Kehrer, and C. Csallner, "50417/ScoutSL: ScoutSL Simulink Search Engine," Aug. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8266234

[18] ——, "ScoutSL Artifacts," Aug. 2023. [Online]. Available: https://figshare.com/articles/dataset/ScoutSL_Artifacts/23717763

[19] S. L. Shrestha, S. A. Chowdhury, and C. Csallner, "DeepFuzzSL: Generating models with deep learning to find bugs in the Simulink toolchain," in *DeepTest*. ACM, May 2020.

[20] ——, "SLNET: A redistributable corpus of 3rd-party Simulink models," in *MSR*, 2022, pp. 01–05.

[21] ——, "Replicability study: Corpora for understanding simulink models & projects," in *ESEM*. IEEE, 2023, to appear.

[22] S. L. Shrestha and C. Csallner, "SLGPT: Using transfer learning to directly generate Simulink model files and find bugs in the Simulink toolchain," in *EASE*. ACM, 2021, pp. 260–265.

[23] T. Dohmke, "100 million developers and counting," 2023, accessed in Mar 2023. [Online]. Available: https://github.blog/2023-01-25-100-million-developers-and-counting/

[24] E. D. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, "The Qualitas corpus: A curated collection of Java code for empirical studies," in *APSEC*. IEEE, Nov. 2010, pp. 336–345.

[25] C. Vendome, E. J. Rapos, and N. DiGennaro, "How do I model my system?: A qualitative study on the challenges that modelers experience," in *ICPC*. ACM, May 2022, pp. 648–659.