Towards Modeling Process Mining for Graphical Editors

MohammadHadi Dehghani, Luca Berardinelli, and Manuel Wimmer
Institute of Business Informatics - Software Engineering & CDL-MINT, Johannes Kepler University, Linz, Austria
Email: mohammadhadi.dehghani@jku.at, luca.berardinelli@jku.at, manuel.wimmer@jku.at

***Abstract*—Engineering tools typically offer graphical environments to help modelers deal with the complexity of designing software-intensive systems. Collecting and analyzing how different users perform modeling actions is a valuable asset to improve the user experience, the quality of the modeled system, and the evolution of the modeling language and accompanying tool support.**

This tool paper presents a novel tool that captures user interaction events in graphical modeling editors and enables mining modeling processes. Modeling events from Sirius-based graphical editors and GLSP-compliant editors are saved in IEEE eXtensible Event Stream (XES) format and integrated with the open-source ProM process mining tool as one example. By importing modeling traces into process mining tools, analysts can visualize and gain insights into the underlying modeling processes, enabling informed decision-making for designing modeling language and tool improvements. Initial experimental results demonstrate the applicability of our tool in capturing and analyzing user interactions on desktop and Web editors in solo and collaborative modeling sessions.

***Index Terms*—Modeling Processes, Event Streams, Process Mining, Change Recording, Modeling Tools**

I. Introduction

Software and system development is an evolutionary process, and related artifacts undergo extensive changes in modern software engineering methodologies. These changes usually continue even after software deployment for maintenance purposes [1]. Models are no exception to this fact and are being used more than before in the software and system development processes as model-driven [2] and low-code [3] software development methodologies are rising to speed up the development cycle and support reusing software artifacts. It is, therefore, of high importance to understand and analyze the changes that occur in models and supporting tools throughout the different phases of development.

Understanding how users interact with graphical modeling editors can provide valuable insights for optimizing the software development process and identifying usability issues. To achieve this goal, this paper presents a *modeling process mining* approach for EMF-based models. The approach is supported by a *modeling event recorder* (MER) tool[1] to collect events from desktop and web-based graphical modeling editors implemented atop Sirius [4], an open-source framework that allows generating graphical modeling workbenches for specific domains from Ecore metamodels [5], and editors leveraging the Graphical Language Server Platform (GLSP) [6].

[1]https://marketplace.eclipse.org/content/mer

The proposed approach leverages the capabilities of the EMF Notification API [7] to record user interaction events in the form of *modeling traces*. These traces are saved in the widely-accepted and standardized IEEE eXtensible Event Stream (XES) format [8] that promotes integration with existing process mining tools. By collecting XES-compliant modeling event traces, off-the-shelf process mining tools, such as ProM Tools [9], can be integrated with MER to analyse the modeling activities performed via Sirius-based graphical editors. Analysts, language engineers, and tool developers can use our tool to understand how users navigate and interact with graphical modeling editors.

This tool paper presents the design and implementation of MER, the event logger tool, highlighting its capabilities in capturing and analyzing user interactions within desktop and Web editors based on Sirius [4] and GLSP frameworks [6].

The remainder of this paper is structured as follows. Section II provides a domain-independent overview of the proposed approach enabling modeling event recording and modeling process mining for graphical editors. Section III discusses the design of our tool for modeling process mining for desktop and Web editors. Section IV discusses related work, while Section V concludes the paper and suggests directions for future research.

II. Approach

The use case diagram in Fig. 1 depicts the main stakeholders and capabilities to enable a modeling process mining approach from a domain-independent standpoint. In the following, we describe each use case and we provide a short list of the most prominent underlying technologies.

A *modeler* performs modeling activities using a graphical modeling editor. Modeling events are recorded in logs. Thenceforth, when the user ends the modeling activities, the modeling events are serialized in a specific format to feed *modeling process mining* tasks, triggered by an *analyst*. A *graphical representation* can be generated from mined modeling processes to help analysts detect modeling patterns and potential bottlenecks during graphical modeling activities. In the following subsections, we detail each of the use cases.

A. Modeling

In MDE, models are the cornerstone machine-readable artifacts of the engineering process [2]. Graphical modeling helps beginners (e.g., citizen developers of LCDP platforms [3],

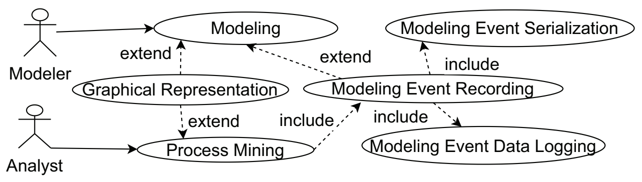

Fig. 1: Use Case Diagram

[10]) and expert modelers deal with complexity and improve communication. EMF [7] and Sirius [4] are two prominent technologies [11] to define modeling languages' abstract and concrete syntax through the Ecore metamodeling language and Sirius, respectively, and to automatically providing Eclipse-based graphical model editors by interpreting viewpoints descriptors.

B. Modeling Event Recording and Data Logging

Modern information systems record detailed events that occur within the system for later processing and analysis [12]. These systems record users' actions on the application, finally aggregating them into activities. The recorded data includes various details of single actions and overall activities at different, possibly customizable, granularity levels. It usually consists of the name, order, duration of the performed actions/activities and the resources or users involved in each activity. These records are called *event data* [13]. Event data logging is an essential capability, and event data logs are the primary input to enable *process mining*. Therefore, properly collecting and recording the event data is highly important.

TABLE I: Example event data from a graphical modeling editor application

Case-ID	Activity	TimeStamp	Resource	User
…	…	…	…	…
6824	Create EClass	13:48:21	A.xmi	U1
6824	Set EClass:name	13:48:56	A.xmi	U1
6825	Create EClass	13:49:30	B.xmi	U2
6824	Create EAttribute	13:49:53	B.xmi	U1
6824	Set EAttribute:name	13:50:21	B.xmi	U1
…	…	…	…	…

We are interested in recording and logging events from EMF-based graphical editors. Table I shows an example event log which includes a collection of modeling event data captured from EcoreTools [14], which is a graphical modeling editor for the Ecore metamodeling language [15]. Each row of this table corresponds to an event that indicates the execution of an *action* in a particular process instance or *case*, and each column corresponds to an event *attribute*. The set of rows, possibly including many cases, is a *trace*. For example, in the trace for case 6824, the user U1 creates an EClass, sets its name inside an XMI model file named "A.xmi", then creates an EAttribute, and changes its name in the file "B.xmi". In the meantime, as this user is modifying these files, user U2 creates an EClass in file "B.xmi". Notice that this action has a different Case ID as each user is performing a specific activity. All these events are sorted based on execution timestamps [13].

C. Modeling Event Serialization

Nowadays, information systems produce huge amounts of event data [16]. These applications require efficient serialization methods that save storage space and memory footprint. In addition, the data should be structured to facilitate the analysis and mining of the recorded events. In this regard, there are existing serialization methods that tackle this issue, like eXtensible Event Stream (XES) [8], Object-Centric Event Logs (OCEL)[2], Mining eXtensible Markup Language (MXML) [17], among others. Generally, process mining tools support one of these standards to process events.

The serialization format of the modeling event recording tool determines which process mining tools support analyzing the logged events. In particular, the XES standard is promoted by the IEEE. It offers an XML Schema that defines the structure of an XES event log/stream, plus extensions that define additional attributes to enrich the information of logged events (e.g., time extension). Eighteen process mining tools accept the XES format as input at the time of writing[3] .

D. Process Mining

According to v.d. Aalst et al. [16], [18], *process mining* refers to discovering, monitoring, and improving real processes by extracting knowledge from information systems' event logs to gain insight and optimize a process. Via process mining, usage patterns can be identified, and process bottlenecks can be detected and resolved. It is also possible to notice when users deviate from the modeled patterns.

In this paper, we focus on collecting modeling events that enable *modeling process mining*, i.e., the discovery, monitoring, and improvement of the modeling activity performed by a modeler creating prescriptive models manipulated in graphical model editors suitable for MDE processes. Therefore, process mining capabilities are inherited from available process mining tools, like PrOM.

III. Tool Support

This section presents a prototypical implementation of the modeling event recording and process mining capabilities depicted in Fig. 1. Modeling, modeling event recording, and process mining capabilities can be realized by integrating three components, i.e., (i) a graphical modeling editor, (ii) a modeling event recorder, and (iii) a process mining tool.

In this paper, we propose a concrete model-driven realization of a modeling process mining tool connecting (i) Sirius-based graphical modeling editors, (ii) MER, a new modeling event recorder for EMF models, and (iii) the open-source process mining tool ProM [9].

Fig. 2 depicts a possible interaction of such a tool. First, a modeler creates their desired modeling project and starts designing the models via a dedicated Sirius-based graphical editor (①). Then MER detects the beginning of a modeling session and starts listening to the events happening inside that

[2]https://ocel-standard.org/

[3]http://www.xes-standard.org/tools

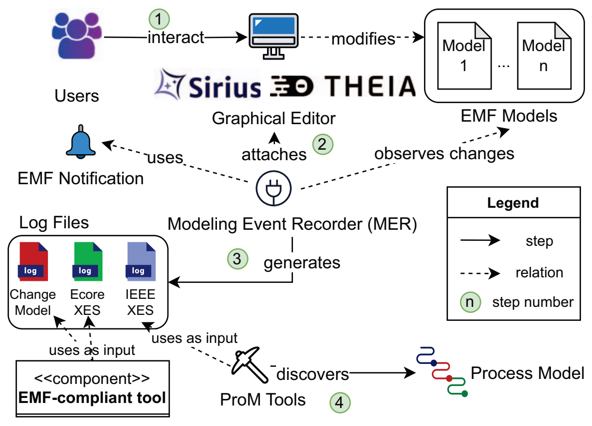

Fig. 2: Modeling process mining workflow.

session (②). The events are in the form of EMF Notifications, and they happen whenever the modeler modifies a model through the editor GUI. At the end of a modeling session, three log files are generated. Two of them represent the same trace log in two different serialization formats, i.e., a standard XES trace, conforming to published XES XML schema [8], and an EMF-based XES model conforming to a new XES metamodel. The third artifact is a Change Model conforming to the Change Metamodel introduced by EMF [7] (③).

The IEEE XES log file is manually fed to ProM and any IEEE XES compatible tool [8], while the equivalent XES model and Change Model are suitable for EMF-compliant tools. Thanks to ProM's mining algorithms, analysts can discover the process model describing the interaction of the modelers with the graphical editor (④).

In the following, we discuss the rationales, technical details, and related artifacts involved in the realization of the aforementioned components. The source code and additional material are available in [19].

A. Desktop and Web-based Editors: Sirius and Theia

When designing a model, it is convenient to have a graphical representation of the model, with the possibility of editing through a drag-and-drop editor [20]. Moreover, with the growth of cloud computing and the progress in web application development, the software industry is leveraging cloud environments and web user interfaces. This is particularly evident for Low Code Development Platforms (LCDP) [10] where the accidental complexity of model-driven techniques and practices [2], [11] are (partially) hidden to non-experts, a.k.a. citizen developers, by graphical editors and advanced services.

In this work, we leverage Sirius [4] and the GLSP-compliant Theia [21] frameworks and then offer the approach to desktop and Web graphical model editors. This enables us to conduct experimental case studies with different modeling languages and graphical editors.

1) Sirius-based graphical editors: Thanks to Sirius, users have a reusable way of defining graphical modeling editors extending the Eclipse IDE desktop application. Given an existing or new metamodel in Ecore, Sirius can automatically generate an editor that lets users graphically define models of different types [22]. In particular, Sirius specifies a Viewpoint Specification Project allowing different graphical concrete syntax options such as generic diagrams, edition tables, crosstables, trees, and sequence diagrams [5], taming the accidental technical complexity required for building graphical modeling editors from scratch [20].

Examples. The current integration activities regard enabling process mining for (i) AutomationML [23] using CAEX Modeling Workbench editor [24] (a demo video is available in [19]), (ii) SysML via Papyrus Sirius Integration [25], and (iii) Ecore via Eclipse Ecore Tools [14]. AutomationML and SysML process mining capabilities are part of a more extensive industrial case study with Volvo Construction Equipment [26] conducted within the AIDOaRt European project [27] where MER has been already applied.

2) Theia-based editors: GLSP [6] is available for creating graphical editors based on Web technologies. The Language Server Protocol (LSP) [28] and an extensible client and server frameworks are offered by GLSP. Moreover, GLSP makes it possible to create Web graphical editors with the server handling computation-intensive tasks. Eclipse Theia, VSCode, and Eclipse Desktop offer integration layers [6].

Theia [21] is a GLSP-compliant extensible platform that enables the creation of cloud-based and desktop IDE-like editors using Web technologies (HTML, CSS, and TypeScript) [29]. The Web-based editor consists of two components. A server application (back-end) and one or several client applications (front-end) that connect to the server. The client application contains no modeling logic. It is responsible for providing a user interface, sending all the actions to the server, and receiving updates from the server. Following the same rationale, the MER component for Web editors is deployed on top of the server application.

We have forked the Eclipse Theia Ecore Tools' GitHub repository [19] and integrated MER into the Theia Ecore Tools so that the corresponding event log is saved whenever a graphical model is saved. Theia Ecore Tools includes a client and a GLSP server, which has been modified on purpose.

Example. The GLSP-compliant Ecore editor, integrated into the Eclipse Theia IDE, provides a Web-based version of the popular Ecore tools. It allows its users to graphically create and modify Ecore Models via a diagram editor integrated into the web-based IDE Eclipse Theia [30].

B. Modeling Event Recording: MER

The MER component is the heart of the proposed approach. It detects any (potentially concurrent) modeling sessions and starts recording the events while the session is active. It leverages the EMF Notification API [7] to listen to model resource changes. Once a modeling session is concluded, three log files are generated: the IEEE XES log file, an XES model, and a Change Model. The IEEE XES log file is compatible with 18 different tools [8], including the chosen process

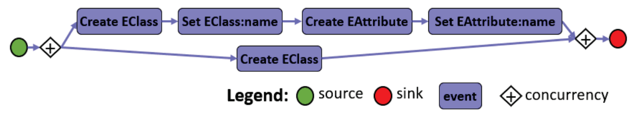

Fig. 3: Example process model created for Table I by ProM.

mining tool ProM. We modified [19] the OpenXES library for Java [8] to serialize the log traces in an efficient manner.

The same traces are also saved into an equivalent XES model conforming to a given XES metamodel (Fig. 4, further described later). The IEEE XES log file can be consumed by ProM and any IEEE XES compatible tool [8], while the equivalent XES model and Change Model are suitable for EMF-compliant tools.

As shown in Fig. 2, a Change Model is also generated according to an extended Change Metamodel [19] introduced in [7]. In particular, we extended the overall `ChangeDescription` by adding a *timestamp* attribute to `FeatureChange`. The benefit is twofold. It allows the collection of timed change descriptions while preserving compatibility thanks to such a non-breaking change [31] to the original Change Metamodel. It enables modeling event recording capability to desktop and Web graphical editors.

The MER component can be deployed as an Eclipse plugin for Sirius-based desktop editors and as a Java server application for Theia-based web editors.

C. Process Mining: ProM

Different tools support generic process mining tasks, such as ProM [9], Celonis[4] and Disco[5]. In this paper, we choose the ProM tool because it is a well-known and open process mining tool that has been used extensively in research. It offers several plugins for different mining purposes, supports common log formats as input (including XES), and generates process models in different formats. For example, the ProM tools contain some miner plugins that generate different types of process models as their output. Some of the most common miners of ProM are inductive visual miner, which produces process trees; Integer Linear Programming (ILP) miner, which produces Petri nets; and Heuristic miner, which produces C-nets. By integrating ProM as a process mining tool, we also inherit its capability to visualize mined process models. Fig. 3 shows a modeling process model of an Ecore metamodel, created by ProM using the inductive visual miner plugin. This is one of the several ways that a process model can be extracted from the same log shown in Table I via different algorithms and various configurations for each one

D. XES Metamodel

Our approach offers an XES metamodel to save modeling events as an EMF artifact to ease the integration with EMF-compliant tools. For example, in our previous work [26], the

[4]https://www.celonis.com/
[5]https://www.fluxicon.com/disco/

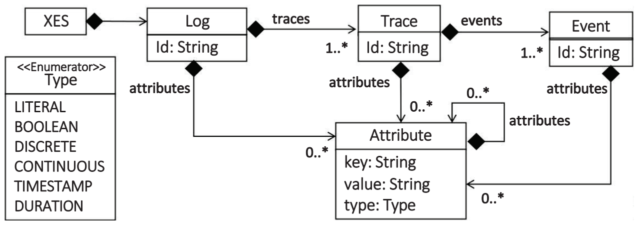

Fig. 4: XES metamodel based on the schema provided by [8].

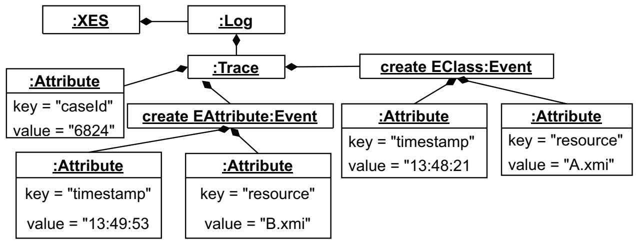

Fig. 5: Mapping two events of Table I to XES.

collected XES models are used to feed EMF-based model generators to train a model recommender [32].

Fig. 4 shows the metamodel of the XES format. As depicted, the root class is `XES`, which contains the rest of the elements. Each XES file may contain multiple `Logs`. Every `Log` should contain at least one and may contain several `Trace` elements. Each `Trace` includes a non-empty series of `Events` (activities within a process). The `Attribute` element defines the properties of `Logs`, `Traces` and `Events`. In addition, each `Attribute` has a key that is unique within the same `Log`, `Trace` and `Event` elements, i.e., one `Event` cannot have two or more attributes with the same key, but two different `Events` may have an `Attribute` with the same key.

As we leverage the EMF Notification API to observe the changes, but we need the XES format for serialization, we have developed a transformation to adapt the EMF Notification objects to XES objects. We show how we map the EMF events to XES via an object diagram in Fig.5. This object diagram partially represents two events from Table I.

IV. Related Work

Due to space limitations, we focus on the most relevant related work. In [33], Tinnes et al. present the OCKHAM tool that learns model transformations for mining edit operations from existing models in a model repository. Our approach delegates process mining to capabilities inherited by XES-compliant tools like ProM and works directly with recorded change logs. The work in [34] discusses a generic operation recorder for model evolution based on an operation metamodel. Like MER, it reuses EMF Notifications but ignores process mining capabilities and compatibility with standards like XES. The work in [35] outlines a research agenda to improve collaborative graphical modeling focusing on users and change history. It also proposes using Event Sourcing to support model versioning and collaboration. Our approach can contribute to collaborative graphical modeling via Theia-based Web editors by collecting edit events in XES. Please note that previously the integration of MDE techniques with process mining techniques has been proposed, but only for generated applications from models [36].

V. Conclusion and Future Work

This paper introduces a novel tool chain for capturing and analyzing user interaction events in graphical modeling editors. Our MER tool captures user interactions and saves them in the XES format (amongst others). These event logs can be imported into, e.g., the ProM process mining tool to visualize and analyze user processes. Moreover, to leverage the extensive EMF-based tool ecosystem in future work, we devised an XES metamodel in Ecore and generated compliant XES trace models.

This represents the basis for further analysis mechanisms that may improve user experiences. Future research may focus on enhancing the tool with event logging formats that also consider the models' contents, and expanding its compatibility with other editors and process mining tools.

Acknowledgment

This project has received funding by the AIDOaRt project, an ECSEL Joint Undertaking (JU) under grant agreement number 101007350, by the EU Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 81388, and by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development (CDG).

References

[1] J. Garcia and J. Cabot, "Stepwise Adoption of Continuous Delivery in Model-Driven Engineering," in *DEVOPS 2018*, pp. 19–32, Springer, 2019.

[2] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*. Synthesis Lectures on Software Engineering, Cham: Springer International Publishing, 2017.

[3] D. Di Ruscio, D. S. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-code development and model-driven engineering: Two sides of the same coin?," *Softw. Syst. Model.*, vol. 21, no. 2, pp. 437–446, 2022.

[4] "Sirius." https://eclipse.dev/sirius/overview.html. Accessed: 2023-07.

[5] V. Viyović, M. Maksimović, and B. Perisić, "Sirius: A rapid development of DSM graphical editor," in *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*, pp. 233–238, 2014.

[6] "Eclipse GLSP." https://eclipse.dev/glsp/. Accessed: 2023-07.

[7] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: Eclipse Modeling Framework*. Pearson Education, 2008.

[8] "IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams," standard, IEEE, 2016.

[9] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The ProM Framework: A New Era in Process Mining Tool Support," in *Applications and Theory of Petri Nets 2005*, pp. 444–454, Springer, 2005.

[10] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the Understanding and Comparison of Low-code Development Platforms," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 171–178, IEEE, 2020.

[11] A. Iung, J. Carbonell, L. Marchezan, E. Rodrigues, M. Bernardino, F. P. Basso, and B. Medeiros, "Systematic Mapping Study on Domain-specific Language Development Tools," *Empirical Software Engineering*, vol. 25, pp. 4205–4249, Sept. 2020.

[12] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated Discovery of Process Models from Event Logs: Review and Benchmark," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 4, pp. 686–705, 2019.

[13] D. Schuster, S. J. van Zelst, and W. M. van der Aalst, "Utilizing Domain Knowledge in Data-driven Process Discovery: A Literature Review," *Computers in Industry*, vol. 137, p. 103612, 2022.

[14] "Ecore Tools." https://projects.eclipse.org/projects/modeling.emft.ecoretools. Accessed: 2023-07.

[15] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: Eclipse Modeling Framework*. Pearson Education, 2008.

[16] W. M. P. van der Aalst, *Process Mining: Data Science in Action*. Springer, 2nd ed., 2016.

[17] B. F. van Dongen and W. M. Van der Aalst, "A Meta Model for Process Mining Data.," *EMOI-INTEROP*, vol. 160, p. 30, 2005.

[18] W. Van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying History on Process Models for Conformance Checking and Performance Analysis," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[19] "Modeling Process Mining." https://github.com/jku-win-se/modeling-process-mining. Accessed: 2023-07.

[20] F. Bedini, R. Maschotta, and A. Zimmermann, "A Generative Approach for Creating Eclipse Sirius Editors for Generic Systems," in *2021 IEEE International Systems Conference (SysCon)*, pp. 1–8, 2021.

[21] "Eclipse Theia." https://theia-ide.org/. Accessed: 2023-07.

[22] S. Jäger, R. Maschotta, T. Jungebloud, A. Wichmann, and A. Zimmermann, "Creation of Domain-Specific Languages for Executable System Models with the Eclipse Modeling Project," in *2016 Annual IEEE Systems Conference (SysCon)*, pp. 1–8, 2016.

[23] "AutomationML." https://www.automationml.org/. Accessed: 2023-07.

[24] "CAEX Modeling Workbench." https://github.com/amlModeling/caex-workbench. Accessed: 2023-07.

[25] "Papyrus Sirius Integration." https://git.eclipse.org/c/papyrus/org.eclipse.papyrus-sirius.git. Accessed: 2023-07.

[26] J. Bergelin, L. Berardinelli, D. Bilic, H. Bruneliere, A. Cicchetti, M. Dehghani, C. Di Sipio, J. Miranda, A. Rahimi, and R. Rubei, "Towards Automating the Industrial Design of CPS: The Experience of Volvo Construction Equipment." Availabe at SSRN: https://ssrn.com/abstract=4484021 or https://dx.doi.org/10.2139/ssrn.4484021, 2023.

[27] H. Bruneliere, V. Muttillo, R. Eramo, L. Berardinelli, A. Gomez, A. Bagnato, A. Sadovykh, and A. Cicchetti, "AIDOaRt: AI-augmented Automation for DevOps, a Model-Based Framework for Continuous Development in Cyber-Physical Systems," *Microprocessors and Microsystems*, vol. 94, p. 104672, 2022.

[28] "Language Server Protocol." https://microsoft.github.io/language-server-protocol/. Accessed: 2023-07.

[29] "Eclipse Theia." https://theia-ide.org/. Accessed: 2023-07.

[30] "EMF Cloud." https://eclipse.dev/emfcloud/. Accessed: 2023-07.

[31] A. Cicchetti, D. Di Ruscio, and A. Pierantonio, "Managing Dependent Changes in Coupled Evolution," in *International Conference on Theory and Practice of Model Transformations*, pp. 35–51, Springer, 2009.

[32] C. Di Sipio, J. Di Rocco, D. Di Ruscio, and P. T. Nguyen, "MORGAN: a Modeling Recommender System based on Graph Kernel," *Software and Systems Modeling*, pp. 1–23, 2023.

[33] C. Tinnes, T. Kehrer, M. Joblin, U. Hohenstein, A. Biesdorf, and S. Apel, "Mining domain-specific edit operations from model repositories with applications to semantic lifting of model differences and change profiling," *Automated Software Engineering*, vol. 30, no. 2, p. 17, 2023.

[34] M. Herrmannsdoerfer and M. Koegel, "Towards a generic operation recorder for model evolution," in *Proc. of the 1st International Workshop on Model Comparison in Practice*, pp. 76–81, 2010.

[35] J. Pietron, "Enhancing Collaborative Modeling," in *Proc. of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proc.*, pp. 1–6, 2020.

[36] A. Mazak and M. Wimmer, "On Marrying Model-Driven Engineering and Process Mining: A Case Study in Execution-based Model Profiling.," in *SIMPDA*, pp. 78–88, 2016.