"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813884"

Project Number: 813884

Project Acronym: Lowcomote

Project title: Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms

Concepts for Mining Low-Code Engineering Repositories

Project GA: 813884

Project Acronym: Lowcomote

Project website: https://www.lowcomote.eu/

Project officer: Thomas Vyzikas

Work Package: WP4

Deliverable number: D4.2

Production date: 31 March 2022

Contractual date of delivery: 30 November 2020

Actual date of delivery: 31 March 2022

Dissemination level: Public

Lead beneficiary: Johannes Kepler University Linz

Authors: MohammadHadi Dehghani, Ilirian Ibrahimi

Contributors: Antonio Garmendia, Luca Berardinelli

Project Abstract

Low-code development platforms (LCDP) are software development platforms on the Cloud, provided through a Platform-as a-Service model, which allow users to build completely operational applications by interacting through dynamic graphical user interfaces, visual diagrams and declarative languages. They address the need of non-programmers to develop personalised software, and focus on their domain expertise instead of implementation requirements.

Lowcomote will train a generation of experts that will upgrade the current trend of LCPDs to a new paradigm, Low-code Engineering Platforms (LCEPs). LCEPs will be open, allowing to integrate heterogeneous engineering tools, interoperable, allowing for cross-platform engineering, scalable, supporting very large engineering models and social networks of developers, smart, simplifying the development for citizen developers by machine learning and recommendation techniques. This will be achieved by injecting in LCDPs the theoretical and technical framework defined by recent research in Model Driven Engineering (MDE), augmented with Cloud Computing and Machine Learning techniques. This is possible today thanks to recent breakthroughs in scalability of MDE performed in the EC FP7 research project MONDO, led by Lowcomote partners.

The 48-month Lowcomote project will train the first European generation of skilled professionals in LCEPs. The 15 future scientists will benefit from an original training and research programme merging competencies and knowledge from 5 highly recognised academic institutions and 9 large and small industries of several domains. Co-supervision from both sectors is a promising process to facilitate agility of our future professionals between the academic and industrial world.

Deliverable Abstract

This report presents a collection of concepts for mining Low-code engineering repositories. The entire mining process is divided into operational-based mining and state-based mining. The former outlines the concepts, requirements, and approaches for mining users' operational data produced within low-code repositories, while the latter outlines the concepts, requirements, and approaches of mining models themselves. This report also presents a preliminary design and results of an initial mapping study on different approaches for mining and reusing models. Finally, this report presents two approaches for operational-based mining and state-based mining, where the theoretical concepts, implementation options, and proof-of-concept prototypes are presented.

Contents

1 Introduction

Model-Driven Engineering (MDE) is a method in software engineering that focuses on (semi-)automatic development of software artifacts from high-level models [1]. Basically, this paradigm relies on the use of Domain-Specific Modelling Languages (DSMLs) to build such models by using domain concepts. As a result, the implementation effort is reduced due to the fact that the developer is isolated from the implementation details.

MDE methodology can be used to build platforms that produce actual software such as Low-code Development Platforms (LCDPs). For instance, tools such of Google AppSheet [2], Amazon Honeycode [3], among others [4] are becoming popular given their advantage for rapid development of full-stack applications without requiring prior programming knowledge. Therefore, some of these platforms targeting the so-called *citizen developers*, who have no programming skills. Typically, LCDPs provide a user-friendly interface (e.g., graphical editors to describe the different parts of the application), integration with third party systems and cloud development facilities. Due to Gartner [5], in 2024 low-code platforms will be responsible for the majority of the application development activities.

The EU project *Lowcomote* aims to enhance LCDPs by transferring knowledge from MDE to improve low-code development, actually to reach a more systematic low-code engineering. The main Research Objectives (ROs) of Lowcomote are:

- **RO1**: Enabling *Low-code Engineering of Large-Scale Heterogeneous Systems*, by smart development environments on the Cloud and precise integration of low-code languages with new domains.
- **RO2:** Developing a *Large-scale Repository and Services for Low-Code Engineering*, as a Cloud-based service able to handle a very large number of low-code artefacts, and automatically learn from them.
- **RO3:** Producing advancements in *Scalable Low-Code Artefact Management*, as new algorithms and reusable components.

This deliverable tackles RO2 that focuses on defining approaches for mining low-code engineering repositories to learn from existing models and interactions. The deliverable proposes *i)* an initial mapping of the state-of-the-art, *ii)* describes two novel Lowcomote approaches, namely state-based and operation-based approaches, related to mining interaction processes generated by modeling activities on LCDPs.

1.1 Problem Statement

The role of modeling in a LCDP is crucial for the development of the application. This is due to the fact that most likely all the resources are depicted by using diagrams serialized in the forms of models (e.g., implementation, architecture and deployment diagrams). Moreover, these diagrams are created collaboratively and interactively by citizen developers. To do this, these platforms provide multi-faceted set of User Interfaces (UIs) that are designed to support various tasks such as navigation, restructuring, debugging, and delegation for different user groups.

On the Need for Operation-based Mining in LCDPs. Usually, LCDPs generally lack mechanisms to customize the modelling environment. In this sense, we present an approach to adapt the environment to each user. Our approach is based on extracting information of the user's past behaviour in order to automate tasks that improve the usability [6]. Therefore, the motivation is to tame the accidental complexity of such environments by the customization of the modelling environment per user.

On the Need for State-based Mining in LCDPs. LCDPs also lack mechanisms to reuse fragments of already designed models. In this context, model reuse may help in reducing the effort to generate new models. By model reuse we refer to the search for relevant and valid models and model fragments.[1] Parts of such models can be integrated (and thus reused) in other models under construction. However, finding ways to search and reuse these model elements is still challenging. The aim of the proposed state-based mining is to facilitate model-level component discovery and reuse through automated identification of relevant low-code models.

1.2 Contribution

We now explain the three main contributions of this deliverable:

- **Initial State-of-the-Art Mapping:** We conducted an initial mapping study to collect the relevant approaches on operational-based mining and state-based mining. As a result, we collected a set of 97 relevant approaches. We present a summary for each approach, for which modeling support activities (e.g., modeling recommendations, model querying, model clone detection, etc.) they are used, the techniques they apply, the models they operate on, and their contribution type.

[1] In this report we consider as model fragments concrete modeling concepts such as classes, attributes, attributes datatypes, attributes relationships, processes, etc.

- **Operation-Based Mining Approach:** We first introduce basic concepts for capturing, recording, and using operational data generated by system executions and the recognized formats to abstract these executions as process models. Then, by presenting a prototypical tool for mining the operational data generated by a graphical modeling editor we show the feasibility of automatic collection of data from user interaction histories in LCDPs.
- **State-based Mining Approach**: We first present the concepts and steps required for state-based mining. The outlined approach enables modeling recommendations based on mining the state of models. The state of the models is represented as a weighted graph based on term frequencies and stored in a graph repository. A proof-of-concept implementation in an existing LCPD is also provided in this deliverable to demonstrate the feasibility.

1.3 Structure of the Deliverable

The remainder of this deliverable is structured as follows. Section 2 provides the basic background on operation-based and state-based mining, introduce a running example, and concludes with the initial results of an ongoing mapping study on model reuse. Section 4 discusses operation-based mining and state-based mining approaches in detail, respectively, together with a proof-of-concept prototypical implementation in the low-code engineering domain. Section 5 concludes the deliverable and outlines future work.

2 Background

This section gives first an introduction to operation-based and state-based mining in the context of low-code engineering, and subsequently, provides are first mapping of existing approaches which may be inputs for developing operation-based and state-based mining approaches for LCDPs.

2.1 Operation-based and State-based Mining in a Nutshell

All the activities during the work on an LCDP are related to models. Thus, we have separated the mining process in an LCDP to operational-based mining and state-based mining since these cover the domain of mining the work on a low-code repository.

Fig. 1 depicts a generic architecture of LCDP, inspired by [7]. It comprises three layers, namely *application*, *language*, and *model layers*.

The *application layer* provides access to LCDP's applications, e.g., model editors. In cloud-based LCDPs, it may be implemented as services accessed by LCPD users, once logged-in, via advanced Web UIs. The *language layer* is typically accessible by language engineers, i.e., skilled LCDP users that knows how to create or modify LCDP's underlying languages and processes. In model-driven LCDPs, we may expect such languages being defined by metamodels and processes as combination of valid models via model transformations. Finally, the *model layer* concerns the collection of design models, typically created via graphical editors provided as LCDP applications.

Operation-based and state-based mining workflows are defined over this generic architecture and detailed in the following.

Operation-Based Mining. In LCDPs and in their generated applications, a huge amount of user interaction data can be captured, recorded, and analyzed to validate or improve the development processes and the generated application. In this report, we focus on enhancing the user experience of applications generated by LCDPs. Thus, we refer to the user interaction with the application as *operation*. In Fig. 1, the operational data is collected in log files and transformed into runtime models (step 7) to be mapped back to the design model that defines the application (step 8).

State-Based Mining. With the term *state*, we refer to structural information of a model, typically provided by the associated metamodel. The state-based mining process is represented in the model layer in Fig. 1. The design models created by citizen developers (step 3) will be abstracted in a graph-based model and persisted in a graph repository. This graph serves as a knowledge base that retains all the structural information of design models and users' process logs. From the graph model repository, we can extract different information to support the modeling activity (e.g., model recommendations).

Workflow. Fig. 1 depicts the scenario that comprises all the steps of the development of an application in a low-code manner while leveraging both operation-based and state-based mining. The concepts and tools mentioned below will be introduced later in the following Sections 3-4.

1. The language engineer specifies the metamodels for modeling the application structure and behavior.
2. The language engineer defines the processes to generate the application code from models conforming to the aforementioned metamodels.

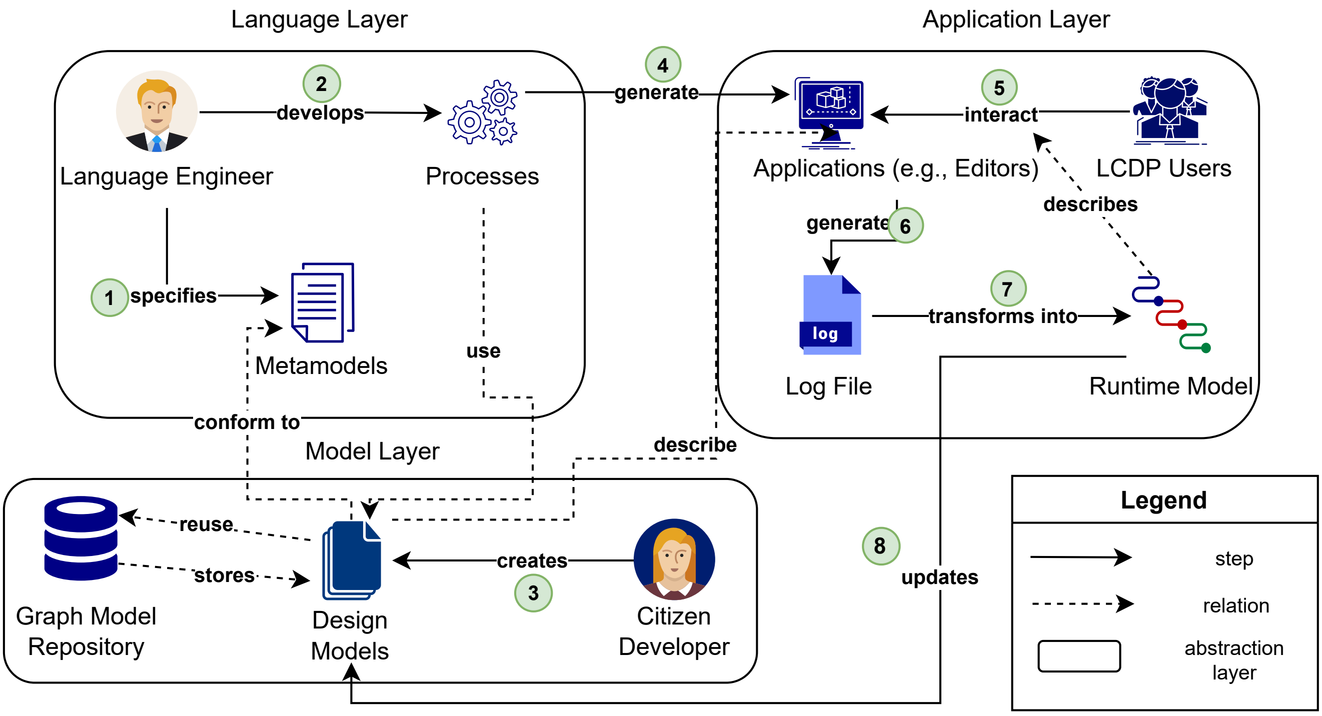


Figure 1: State-based and operation-based mining workflows in a layered LCDP architecture.

3. The citizen developers, who own the domain knowledge but have limited coding capabilities, creates their desired design models defining the desired application structure and behavior. The models must conform to the metamodels defined in Step 1. The design models will be abstracted in a graph-based model and persisted in a graph repository. This graph serves as a knowledge base that retains all the structural information of design models and users' process logs.
4. The processes are run with the design models of the previous step as input. The source-code of the application is generated automatically based on what was described in the design models.
5. The automatically generated application is ready for usage. The users can interact with the applications while the interaction history is being recorded.
6. The event data concerning the user interactions are recorded in a log file in XES [8] serialization format.
7. A process model is mined from the log file [9]. It is a runtime model showing how the users actually interact with the application and what specific actions they take to do their daily tasks in the application and in what order these actions are performed.
8. The runtime model is used to update the design models in order to enhance the user experience with the application based on what was observed from the actual user interaction with the application Finally, this step is combined again with step 3, mapping operational data mapped back to the design model. The result is a feedback loop that can be repeated until the application is tailored to the needs of each user.

2.2 Motivating Example

Fig. 2 sketches a motivating example where mining approaches can be beneficial for LCDPs.

A citizen developer (e.g., Alice) manipulates her design model(s) through a model editor provided as a software application by her preferred LCDP. Since the LCDP is logically organized as the layered architecture shown in Fig. 1, it can offer state-based and operation-based mining capabilities.

If an explicit consent is given, the operational data of each logged-in LCDP users can be collected. In particular, we are interested in modeling actions performed by citizen developers. In our example, Alice creates her own model with the LCDP graphical model editor and her actions are saved as operational data in log files. For the sake of illustration purposes, we show the editing actions on class diagram-like graphical model depicting concepts and relationships typically used in the educational domain (Person, Student, Professor) and logs are generated collecting the user id, the modeling action (e.g., create), and the associated model element (e.g., a class, an attribute, a method, or a relationship).

All these logs from citizen developers using available model editors are persisted in a repository leveraging a common graph-based representation for the sake of homogeneous processing by mining approaches.

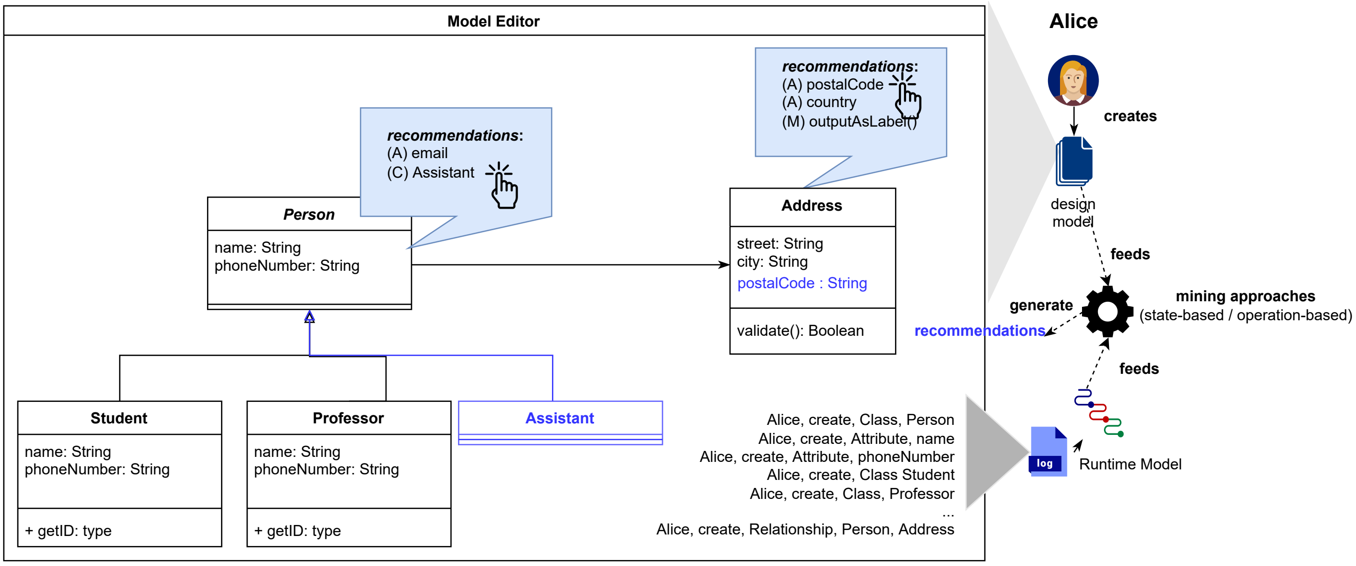


Figure 2: Motivating example on recommendations provided by mining approaches.

Thanks to the combination of different design models and logs representing the modification history of the models as performed by citizen developers, the LCPD can offer recommendation services to Alice. For example, attributes like postal code and country, as well as methods to generate post labels (outputAsLabel()) are likely to be associated to concepts like address. Similarly, the email attribute is likely to be associated to a person. Finally, more domain-specific recommendation can be provided. For example, since Alice's design model concerns the educational domain, the specialized class Assistant is suggested as specialization of the abstract class Person.

Alice can accept or ignore the suggestions provided by the model editor by clicking or not on the recommendation tips (blue notes in Fig. 2). An LCDP offering mining approaches can then leverage the knowledge of its users in the generation of its applications

2.3 Model Reuse: an Initial Mapping Study

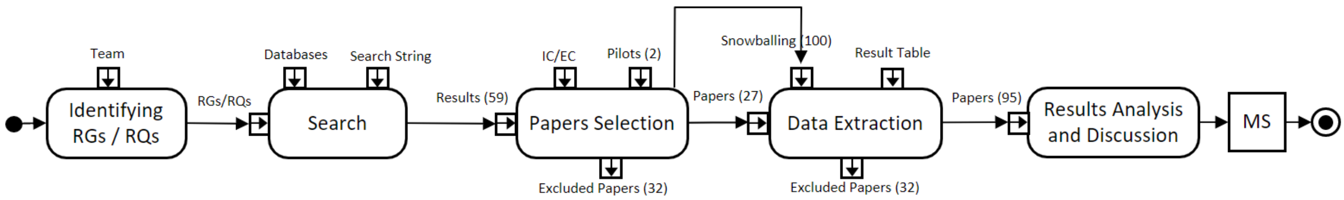


Figure 3: The mapping study protocol.

In order to collect and study the existing literature on model reuse, we present here a preliminary design and results of an ongoing mapping study (MS). We focus on model reuse as the main goals of both, operational and state-based mining is to provide some kind of model reuse features based on previous knowledge.

Fig. 3 shows the mapping study protocol workflow using an activity diagram like visualization. It is inspired by guidelines defined by Petersen et al. [10] for performing systematic mapping studies (MS) in the Software Engineering (SE) domain.

Purpose	To identify and classify existing solutions
Issue	leveraging model reuse
Object	to support citizen developers in the (model-driven) software engineering process based on LCDPs
Viewpoint	from the point of view of researchers.

Table 1: Research Goal.

It consists of the following steps:

1. **Identifying the Research Goals (RG) and the Research Questions (RQ)**. The RG is defined using the Goal-Question-Metric perspective [11], is given in Table 1.
   The preliminary research question (RQ) is:

RQ: *What are the approaches published in the literature concerning reuse of model artifacts?*

We expect to formulate more detailed RQs as soon as we proceed in the analysis of selected papers.

2. **Search**. It is conducted on selected bibliographic sources (ACM[2], SCOPUS[3], and IEEE Xplore[4]) via the execution of search queries.

   The search string is designed by identifying and then combining keywords in a logic formula. Two keyword sets S have been identified in particular:

   S1={"LCDP", "Low-code development platform", "Low-code platform", "No-code platform", "Model-Driven Engineering", "MDE", "Domain-specific modelling", "DSM", "Domain modeling"}

   S2={"Modeling assistance", "Modeling assistant", "Model reuse", "Model recommendation", "Model discovery", "Recommender system", "Model clone detection"}

   S1 aims to cover the concepts related to LCDP and domain modeling, while S2 tends to cover the tasks that enable the reuse of models.

   Each keyword k within the same S_j is combined in a OR proposition. Then S1 and S2 sets S are combined in a AND proposition resulting in the following search string:

   {"LCDP" OR "Low-code development platform" OR "Low-code platform" OR "No-code platform" OR "Model-Driven Engineering" OR "MDE" OR "Domain-specific modelling" OR "DSM" OR "Domain modeling"} AND {"Modeling assistance" OR "Modeling assistant" OR "Model reuse" OR "Model recommendation" OR "Model discovery" OR "Recommender system" OR "Model clone detection"}

   A total number of 59 papers have been collected from the bibliographic sources.

3. **Papers Selection**: Inclusion and Exclusion Criteria (IC/EC) are defined and applied to papers collected from databases, typically reading their titles and abstracts. The same IC/EC criteria are applied to set of collected pilot papers and the ones passing the selection are matched against the collected results to assess the effectiveness of the designed search string(s). Indeed, The higher is the number of matches, the better is/are the chosen search string(s). The two pilot papers [12, 13] have been found by the designed search string. Table 2 present the IC/EC criteria for our MS.

Task	Criteria
Inclusion	Conference, Journal, Tool papers
	Book chapters
	Thesis
Exclusion	Not in English
	Not accessible
	Survey
	Discussion paper

Table 2: Inclusion and exclusion criteria

   The application of IC/EC resulted in *27* selected studies.

4. **Snowballing.** We performed the snowballing on the selected papers, until no additional primary studies were identified and 100 additional relevant papers were added to the selected set to be reviewed.

5. **Data extraction** A Result table template has been used to uniformly collect papers' detailed information. Duplicates are removed from the set of collected papers (128) Typically, the application of IC/EC criteria continues on a full reading basis, and if needed, additional papers are excluded. The data extraction step has been performed on a total of 95 papers. The template of the result table is presented in Table 3

   .

6. **Results**. The preliminary result of the mapping study are collected in Table. 5 in Appendix A.

   In Fig. 4 we have outlined how the amount of studies in model reuse has increased throughout the years. There are studies on model reuse from 1995 and the focus on these studies has grown significantly afterwards. The models are mostly reused by recommendation systems and query

[2]https://www.acm.org/

[3]https://www.scopus.com/home.uri

[4]https://ieeexplore.ieee.org/Xplore/home.jsp

Approach	Summary	Task	Technique	Model type	Contribution type	Year
The reference to the approach	A summary that describes the approach	Which modeling task caused the reuse of models	Which techniques are applied to perform model reuse	What model types can be reused	What is the contribution type	Year of publication

Table 3: A template table for the LR results

facilities from 2006 and now. In the current state of our mapping study, we have included theses [14, 15, 16, 17, 18] and we excluded surveys [19, 20, 21, 22, 23] and discussion papers [24, 25].

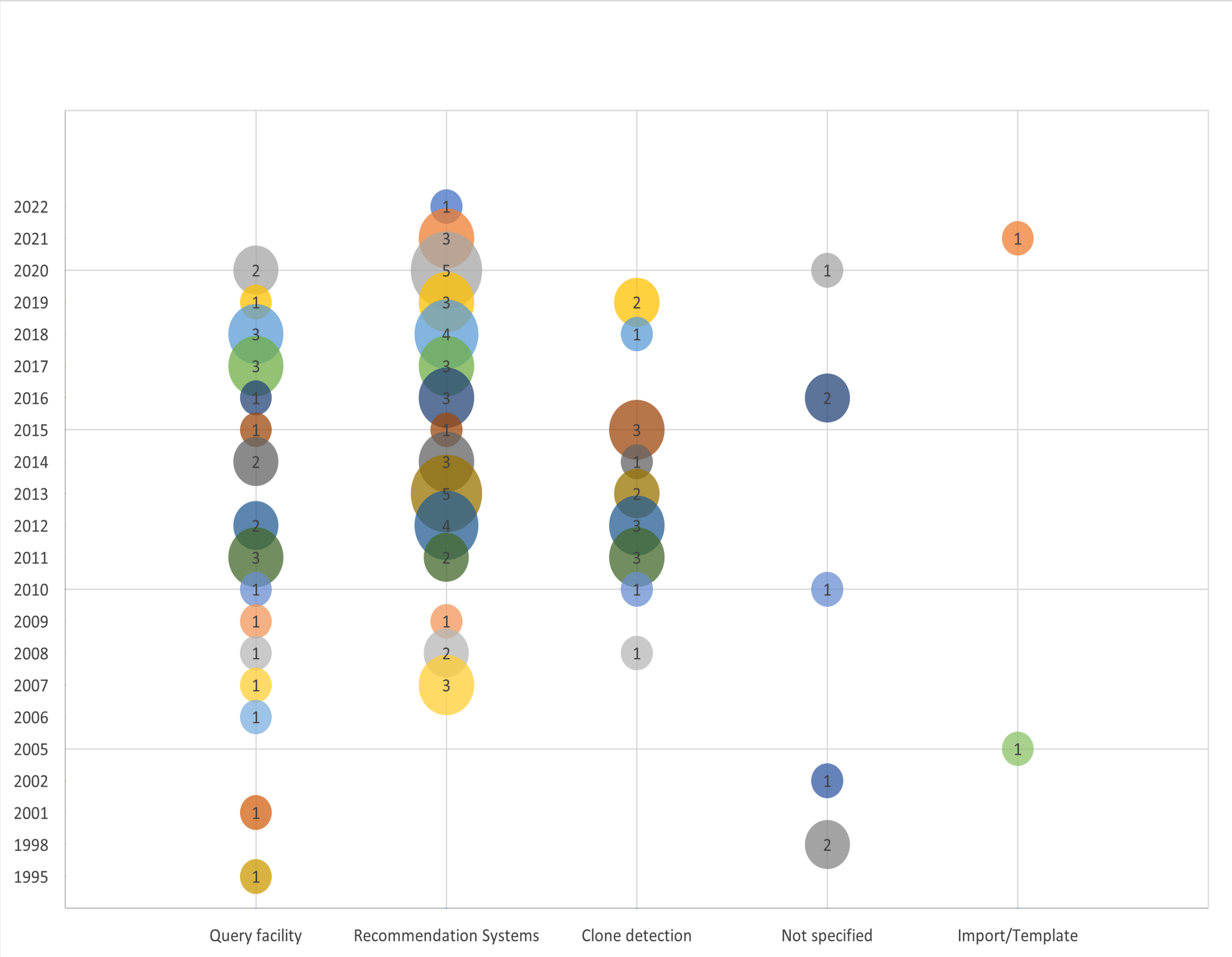


Figure 4: Model reuse studies throughout the years.

3 Operation-based Mining

This section presents operation-based mining which includes a collection of concepts for mining and mapping operational data back to design models. It also covers the requirements to abstract several executions of a system in runtime models along with presenting various types of process models used to describe system executions. Then we discuss automation possibilities for the capture, serialization, and analysis of operational data in LCDPs by presenting an implementation as proof of concept.

3.1 Basic Concepts

Capturing event data: Modern information systems record detailed events that occur within the system for later processing and analysis [26]. These systems record each activity (action) of the users on the application. The recorded data includes details of the performed activities. It usually includes the name, order, duration of the performed activity, and the resources or users involved in each activity, among others. These records are called *event data* [27]. The collection of event data is the core artifact of the mining process, therefore, it is of high importance to properly collect and record the event data.

Table 4: Example event data from a graphical modeling editor application

Case-ID	Activity	TimeStamp	Resource	User	...
⋮	⋮	⋮	⋮	⋮	...
6824	Create EClass	13:48:21	Ecore	U1	...
6824	Set EClass:name	13:48:56	Ecore	U1	...
6824	Create EAttribute	13:49:30	Ecore	U1	...
6824	Set EAttribute:name	13:49:53	Ecore	U1	...
6824	Set Bounds:x	13:50:10	Aird	U1	...
6824	Set Bounds:y	13:50:10	Aird	U1	...
6824	Create EReference	13:50:48	Ecore	U1	...
6824	Set DEdge:target	13:50:48	Aird	U1	...
6824	Set EReference:upperBound	13:51:15	Ecore	U1	...
6825	Create EClass	14:04:06	Ecore	U2	...
⋮	⋮	⋮	⋮	⋮	...

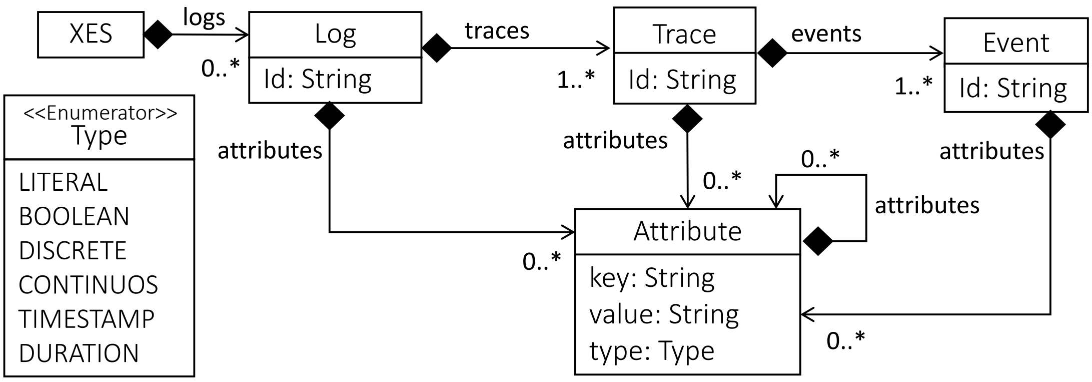


Figure 5: XES metamodel

Table 4 shows an example event log which includes a collection of event data captured from EcoreTools [28], which is a graphical modeling editor for the Ecore metamodeling language [29]. This language was proposed by the Eclipse Modeling Framework (EMF) to design metamodels taking as a reference Object-Oriented Programming (OOP) concepts. For instance, a class is called `EClass` in Ecore, an attribute is called `EAttribute` and a reference is called `EReference`. Each row of this table corresponds to an event that indicates the execution of an activity in a particular process instance which is called a *case*, and each column, corresponds to an event attribute. For example, the first event with the `Case-ID` of `6824` was performed by user `U1` on the `Ecore` resource at `13:48:21`. The set of rows comprises a `trace`. For example, in the trace for case `6824` the user `U1` as a first step creates a class, after that, the user sets the name of the class, then, creates an attribute and changes the name as well. After these actions, the user continues by doing some cosmetics on the diagram by changing the width and height of the rectangle that represents the class (`Set Bounds:x` and `Set Bounds:y`). Lastly, the user creates a reference, sets the target, and updates its `upperbound`. The events within logs are usually sorted based on execution timestamp [27].

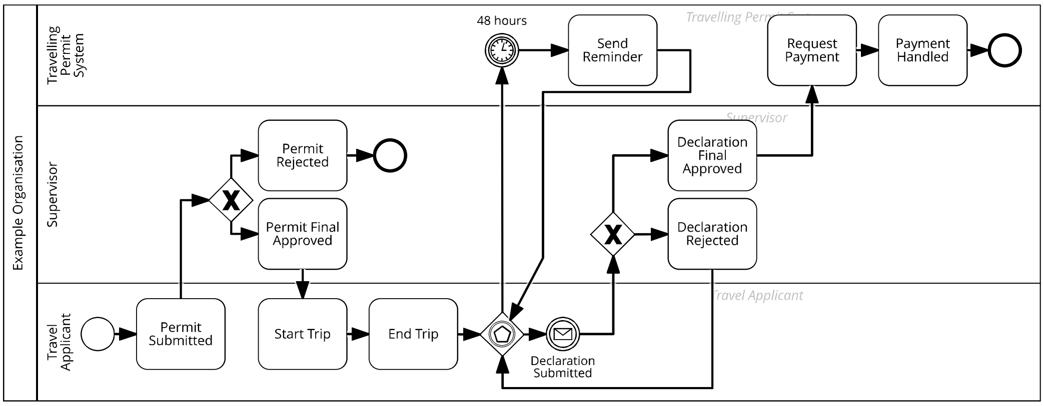
Serializing event logs: Nowadays, the information systems produce huge amounts of event data [30]. These applications require efficient serialization methods, i.e., that saves storage space and memory footprint. In addition, the data should be structured, to facilitate analysis and mine the recorded events. In this regards, there are existing serialization methods that tackle this issue, like XES (eXtensible Event Stream) [31], Object-Centric Event Logs (OCEL) [32], Mining eXtensible Markup Language (MXML) [33], among others. As a general rule, process mining tools support one of these standards to process the events.

One of the most prominent standards adopted by IEEE is *XES*. According to its website [8], up to the time this deliverable was written there are 18 process mining tools that accept XES format as input. Fig. 5 shows the metamodel of the XES format. As it is depicted, the root class is `XES` that contains the rest of 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 define the properties of `Logs`, `Traces` and `Events`. In addition, each `Attribute` has a key which 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.

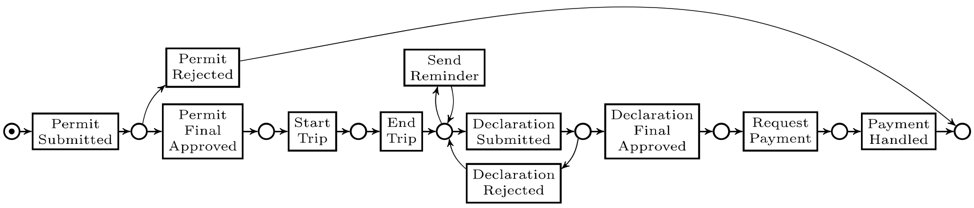
Process mining: A widespread approach to improving an organization's processes is to analyze event data generated during execution. Process mining [30] provides techniques, tools, and methods for systematically analyzing event data to gain insight into how a process is performed. These findings are used to optimize the process which is the main goal of process mining. Process mining refers to discovering, monitoring, and improving real processes by extracting knowledge from information systems' event logs [34].

Via process mining, organizations can get an insight into the usage patterns. By analyzing these patterns the company can detect bottlenecks in the process and resolve them. It is also possible to notice when users deviate from the normal patterns [30]. As the normal patterns are usually the most efficient way to use software and they provide the best user experience, the organizations try to encourage the users to interact with the software based on these patterns.

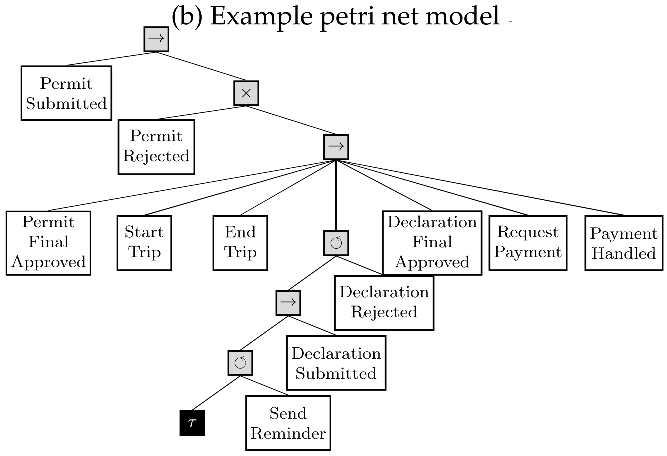
There are more than 25 commercial tools that support process mining tasks [30] such as ProM [35], Celenois [36] and Disco [37]. In this derivable, we would like to highlight the ProM [35] tool, because it is a well-known process mining tool that has been used in many process mining related research studies. It contains many plugins for different mining purposes, supports common log formats as input, and generates process models in different formats .



(a) Example BPMN model



(b) Example petri net model



(c) Example process tree model

Figure 6: Three process models describing the same travel permit process taken from [27]

Representation format: Process models make it possible to graphically depict and specify the execution of a process. There are many recognized process model formats, for example, Business Process Modeling Notation (BPMN) [38], Petri Net [39], and Process Tree [40]. Fig. 6 shows three different process model formats that describe the same process. Indeed all the models in Fig. 6 describe the same control flow of activities, but the expressiveness of the three formalisms is not the same. For example, the BPMN model specifies that a reminder is sent after 48 hours if no declaration is submitted. Further, the model indicates which resources are involved in the execution of the different process activities. This kind of information cannot be presented in the other two formalisms. Most process model formalisms used in process discovery focus on the control flow perspective, i.e., which activities are present in a process and how they relate to each other [27]. In addition to control flow perspectives, some process model formats

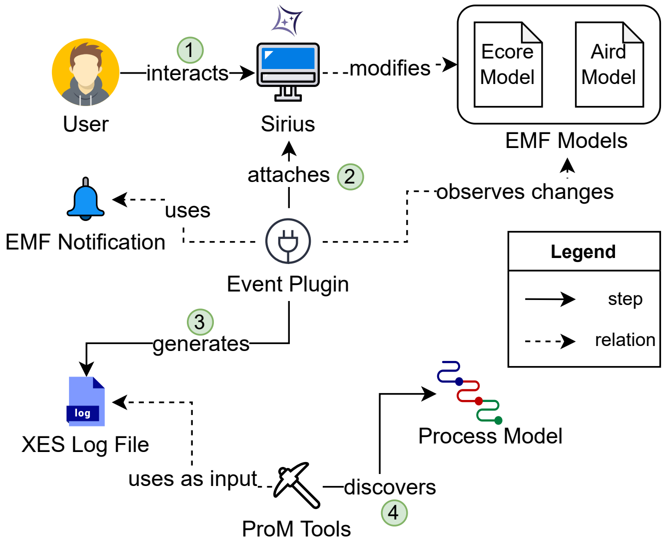


Figure 7: Process mining in Sirius graphical editor

include data perspectives or organizational / resource perspectives [41]. For example, BPMN provides modeling elements (known as swim lanes) to specify the resources involved in performing an activity, as well as elements to model the flow of data artifacts. However, most process detection algorithms only present processes in terms of control flow [27]. In addition to the process model format and its graphical representation, many process detection algorithms limit the classes of models that can be detected within a particular format. Therefore, these algorithms limit the presentation format of the discovered process model. For example, some important subclasses of Petri nets play important roles in process discovery, e.g., Workflow nets (WF-net), sound WF-nets, free-choice WF-nets, and block-structured WF-nets [42, 40].

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.

3.2 Proof of Concept

As proof of concept, we have implemented a mechanism to capture, record, and mine the event data generated by Sirius graphical modeling editor. We have developed an Eclipse plugin that connects to Sirius graphical modeling editors and collects event data when the user interacts with the editor. The source code of the implemented plugin is available on GitHub[5].

Fig.7 depicts the workflow of the implemented plugin. First, the users create their desired modeling project and start designing the models via a graphical modeling editor which is based on Sirius (label 1). Then our implemented plugin detects the start of a Sirius session and starts listening to the events that are happening inside that session (label 2). The events are in the form of EMF Notifications and they happen whenever The Ecore or the Aird models are modified by Sirius. The Ecore model contains the structural data of the model being designed by the graphical editor and the Aird model contains the graphical data for the corresponding visual elements on the canvas. After that, the captured events are converted to XES format and are serialized into an XES log file at the end of the session (label 3). In the end, we load the generated XES log file into ProM tools and by means of the provided mining algorithms, we discover a process model describing the interaction of the users with the graphical editor (label 4).

Fig. 8 shows a mined process model from the EcoreTools modeling editor which is an editor based on Sirius. The log file is captured by our plugin containing multiple user interaction histories when they are designing the Ecore model shown in Fig. 9. The log file is filtered to only contain the main events related to the Ecore model and the names of the events have been altered to make them more understandable. This process model is generated with the inductive visual miner from ProM tools and is in process tree format but is presented in BPMN style to show the flow of the process with similar notations as BPMN but without the complex concepts that are available in BPMN. Each node of this model depicts a change happening in a particular object in the Ecore model which in this example is the creation of that object. If the input log file contained only a trace from one user, we would get a straight line process tree with no concurrency. The only observed behavior would be the sequential execution of 7 object creation events in the exact order that the user has created them. But as we have multiple traces for the same process instance in our log file, the inductive miner detects that some events can happen in different orders and generates a concurrent process

[5]https://github.com/lowcomote/sirius.process.mining

tree.

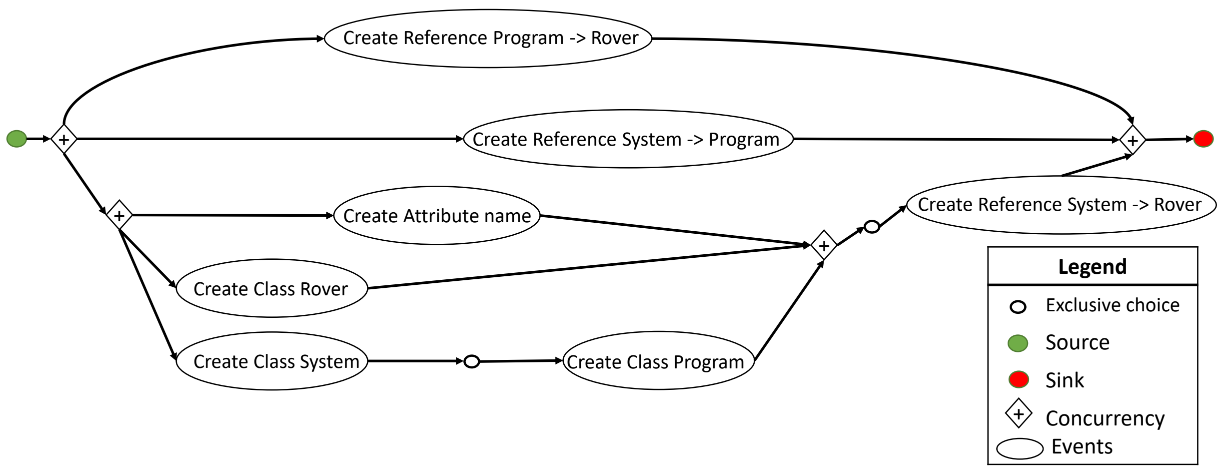


Figure 8: Mined process model from EcoreTools user interaction log captured by our tool

When designing a model, it is more convenient to have a graphical representation of the model, with the possibility of editing through a drag and drop editor [43]. Sirius is an Eclipse project that allows the creation of such graphical editors for the Eclipse Modeling Framework (EMF) models [44]. Thanks to it, users have a reusable way of defining editors, for the IDE Eclipse or the web, by specifying a Viewpoint Specification Project, without needing to be an expert in programming languages, the Eclipse environment, or its plugin system [43]. Sirius provides quick development of an editor which lets users graphically define models of different types, either using an existing metamodel or after specifying one for a certain domain [45]. It would be possible to specify representations of viewpoints of various natures with Sirius, such as generic diagrams, edition tables, crosstables, trees, and sequence diagrams [46].

These characteristics make Sirius a good candidate for a domain-specific graphical modeling editor to be used in LCDPs. Fig.9 depicts a screenshot of the EcoreTools graphical modeling editor which is based on Sirius. The user interface consists of a canvas that shows the graphical representation of the model, a palette that contains commands and graphical elements that can be added to the canvas, and the properties window that let the users modify the details of each element of the model by selecting the corresponding element on the canvas. As Sirius can generate customized modeling editors from viewpoint specification models, we can consider the generated editors as low-code applications and apply our operation-based mining technique to them.

As we leverage EMF Notification API for the observation of the changes but we need the XES format for serialization, we have developed some code to adapt the EMF Notification objects to XES objects. Listing 1 depicts a Java class that adapts the FeatureChange class from EMF Notification to Event class from XES. The output of the implemented Eclipse plugin is a log file in XES format. Listing 2 shows a small piece of one of the generated log files that is then fed to the ProM tools to extract a process model from it.

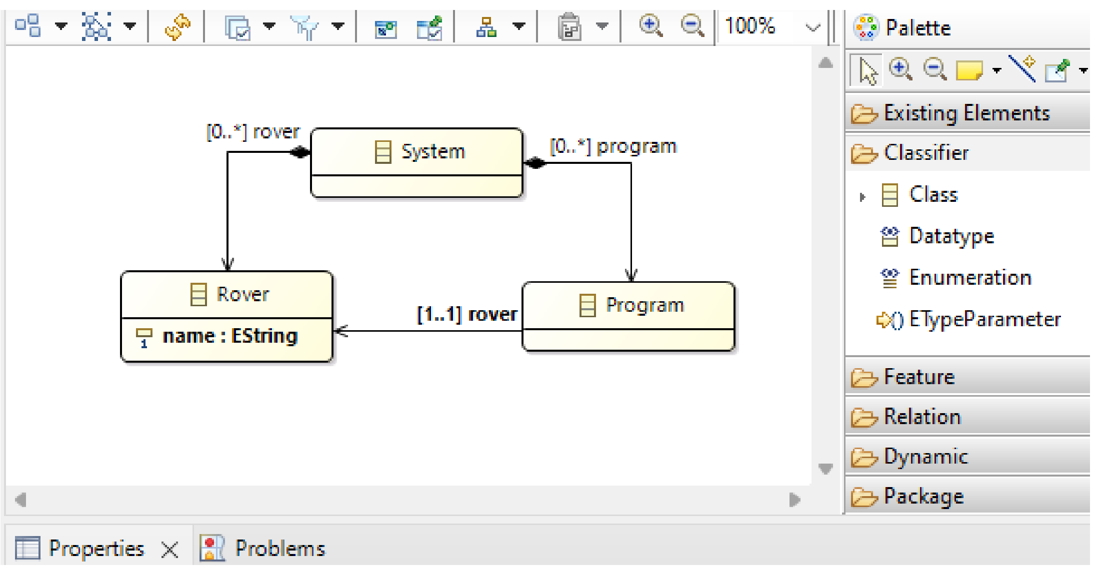


Figure 9: EcoreTools: graphical modelling editor based on Sirius

```java
public class FeatureChangeToEvent extends XEventImpl {

  public FeatureChangeToEvent(EObject reference, FeatureChange featureChange,
     XFactory factory) {
    super();
    put("type", "featureChange");
    putTime(featureChange.getTimeStamp());
    String className = reference.eClass().getName();
    String javaClass = reference.eClass().getInstanceClassName();
    put("class", className);
    put("javaClass", javaClass);
    put("featureName", featureChange.getFeatureName());
    putName(className + ":" + featureChange.getFeatureName());
    put("set", featureChange.isSet());
    String dataValue = featureChange.getDataValue();
    ...
  }

}
```

Listing 1: A java class to adapt EMF FeatureChange objects to XES event objects

```xml
<log xes.version="2.0" xes.features="-" openxes.version="2.27">
  ...
  <trace>
    ...
    <event>
      ...
      <string key="featureName" value="eStructuralFeatures"/>
      <string key="resource" value="ecore"/>
      <string key="javaClass" value="org.eclipse.emf.ecore.EClass"/>
      <string key="type" value="featureChange"/>
      <date key="time:timestamp" value="2022-01-05T14:23:06.361+01:00"/>
      <string key="proxy"
         value="platform:/resource/roverml/model/roverml.ecore#//System"/>
      <string key="concept:name" value="EClass:eStructuralFeatures"/>
      ...
    </event>
    ...
  ...
  </trace>
</log>
```

Listing 2: A piece of the generated XES log file

4 State-based Mining

This section presents the state-based mining process which includes a collection of concepts for mining and reusing the state of the models, i.e., the relevant structural data. It also covers the steps and provides an approach for reusing models mined based on their structural data. Finally, in this section the integration of the approach in a LCDP as a proof of concept is presented.

4.1 Basic Concepts

Abstracting the Structural data. One of the main concepts for mining the state of a model is capturing its relevant structural data and abstracting them in a specific format so they can be accessed afterwards. The abstraction format will serve as a knowledge base for the entire *state-based mining process*.

Graph-based repository. The continually increasing amount of information generated by state-based mining processes poses scalability issues. It is necessary to conveniently persist and access this information and efficiently access potentially large models.

In [47] Chen conducted a performance comparison between relational databases and graph databases in handling large-scale social data and demonstrated that graph databases perform better on large-scale data and have some advantages over relational databases. Also, the comparative analysis of relational and non-relational databases in the context of performance in Web applications conducted by Franczek et al. [48] concluded that non-relational databases perform better when reading data. Thus, graph repositories are a promising technologies for LCDP.

Mining approach Once a convenient solution for data storage is found, the approach consists of the following three main steps: *i) model clone detection*, for finding exact or similar models from a repository to a given model, *ii) model querying*, for searching for a specific model or model fragment in the repository, and *iii) model recommendations*, for supporting the citizen developer with modeling suggestion during the modeling process.

4.2 Prototypical Implementation

This part presents the concepts and steps of an approach that enables model recommendations based on model state mining. The approach starts by converting heterogeneous models into graph-based models and merges them to a single graph-based model which serves as a knowledge graph for the approach. This knowledge graph is persisted in a graph-based model repository as RDF graphs [49].

During the modeling activity performed by users (e.g., the citizen developer in Fig. 2), the approach queries the graph repository to compare the models under construction with any model within the graph repository. The prediction algorithm [50] predicts what could be the next modeling step and recommends the corresponding modeling steps to the user in a ranked order, e.g., based on term frequencies. The overview of this approach is given in Fig. 10, showing the steps realizing the model discovery and reuse based on model-state mining.

As proposed in [51], the model reuse cycle consists in four steps, namely *abstraction*, *selection*, *specialization*, and *integration*.

Abstraction. Abstraction is one of the key elements of the reuse cycle [52]. Thus, the abstraction step is performed by converting heterogeneous models to graph-based models.

It is worth noting that this step has to be suitably implemented for the adopted modeling technology. In our prototypical implementation, we provided abstraction support for Ecore models[6], zAppDev models[7], MoDisco reverse engineered models[8], Microsoft Common Data Model [9].

For instance, zAppDev models, a.k.a business objects, are checked (e.g., by adding all necessary URIs) and then transformed into graph-based models (see step 1 in Fig. 10). Then all zAppDev-based RDF graphs are merged into a single knowledge graph (step 2), which serves as knowledge base for our approach and is persisted into a graph database (step 3) for the sake of efficient data management. In order to enable recommendation algorithms, the knowledge graph is weighted (step 4) based on term occurrence frequency, by counting how many times relationships occur between pairs of classes. Once the weighting step is completed, the weighted graph is persisted back, via updates, to the graph repository (step 5).

All the remaining model reuse steps: Selection, Specification and Integration are incorporated in step 6 in Fig. 10.

[6]https://www.eclipse.org/modeling/emf/
[7]https://zappdev.com
[8]https://www.eclipse.org/MoDisco/
[9]https://docs.microsoft.com/en-us/common-data-model/

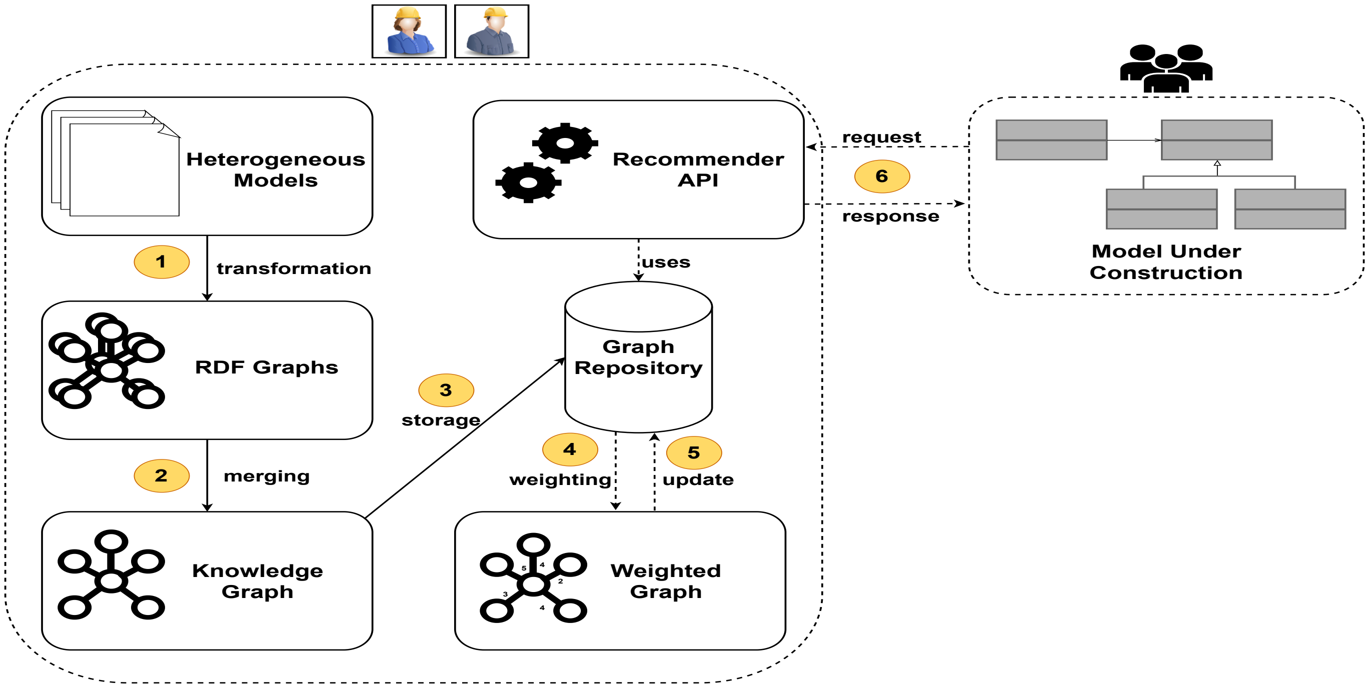


Figure 10: State-based mining approach for model recommendations

Selection. The first step of the selection step is getting the data from the model under construction, as depicted in Fig. 10 step 6 "request" part. If the approach is triggered for recommendations for model elements (e.g., the user clicks any classes or space on the modeling canvas, see Fig. 2) then the approach gets this data extracted from the model under construction and use it as a query input for getting relevant recommendations.

It is worth noting that the approach is currently focusing on *class recommendations*: a recommended class can optionally includes frequently associated attributes and methods. Class recommendations can be *domain-specific* and *domain-independent*. Indeed the same classes within different domains may have different relationships, e.g., a class `Manager` in a bank domain may be related to classes like `Bank`, `Client`, `Account`, etc, while the same class `Manager` may be connected to classes like `Hospital`, `Department`, `Doctor`, or `Nurse` in the healthcare domain. To automatically determine the domain of the model under construction, two or more classes are extracted from the model under construction and used to query the repository for related model names, which in turn help in identifying the model's domain. If the classes taken from the model under construction belong to any model name in the repository, then the latter is consider relevant to suggest recommendations for the model under construction

In case of domain-independent query then the approach searches for class recommendations in the entire repository rather than only in models with relevant model names.

Once the relevant domain has been determined, we proceed with finding potential model elements to be reused. Inspired by the work of Agt-Rickauer et al. [53] N-grams have been chosen to realize the prediction algorithm. In our approach, we used 1-grams, 2-grams, and 3-grams. All of them are in domain-relevant and domain-independent recommendation scenarios. When user triggers the recommendations for a class (request part at step 6 Fig. 10), the approach first perform domain-specific and then domain-independent queries. A list of all related classes are returned, suitably ranked based on their frequency.

Fig. 11 shows the recommendation step in action. The model consists of two classes `Trip` and `Leg` as created by the user. If the recommendation is triggered from the `Trip` or `Leg` classes, the approach checks for any `Trip` or `Leg` class, respectively. Since these classes are connected in the same model, the result is the same regardless from which of these class we asked for recommendations. After finding the `Trip` (or `Leg`) class, the approach generates N-grams to get all the related classes to Trip - Leg (2-grams in this case). The approach finds all connected classes with the relationship Trip - Leg within the repository and returns a list of those classes ranked based on their frequencies. Among them, class `Fare` has the highest weight and is ranked/shown first in the recommendation list.

Concerning class attributes, the approach can find relevant attributes for a given class. First, the approach queries the repository for the owning class, all attributes are collected and returned. Existing attributes are ignored while the others are offered as recommendations to the user.

Specialization. Models in the graph-based repository are agnostic from specific technologies adopted by the specific modeling editor and, more in general, LCDP platform. The specialization step consists in translating the recommendation(s), if accepted, from their graph-based representation back to the original technical space [54] adopted by the model editor offered by the LCDPs.

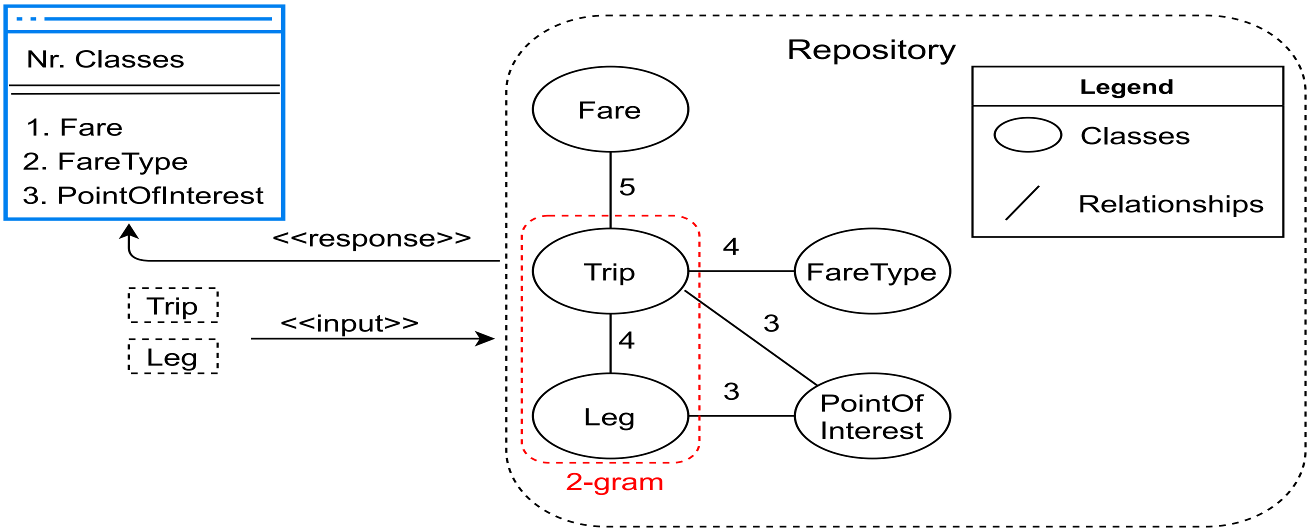


Figure 11: Extracting N-grams from the repository

Integration. After the approach has determined possible model elements that can be recommended and reused based on the inputs received from the model under construction state, the approach proceeds with the integration part. At the time of writing, the integration step is available on the zAppDev[10] LCDP. However, the integration step is exposed as a REST API to allow other model editors, as those used in LCDPs, to access its endpoints to get recommendations during the modeling activity. In Fig. 10 the integration part corresponds to step 6.

The approach and its model reuse cycle (abstraction, selection, specialization, integration) has been implemented in a Java-based proof-of-concept tool called BORA (Business Object Reuse Approach). BORA adopts RDF as graph-based model format, the TDB[11] is a component of Apache Jena for RDF storage and query, and SPARQL [55, 56] as query language.

BORA can manage zAppDev models, Ecore models, and MoDisco reverse engineered models [57]. Such models can be abstracted and stored in BORA's repository and can be reused. It is worth noting that any other graph-based model can be integrated with BORA[12]

Due to industrial property rights, zAppDev documents are not publicly available. However, 543 Ecore models have been collected from the Maven repository[13], converted to RDF, merged, and weighted. The RDF weighted graph obtained from the collected Ecore models can be used for testing recommendations via BORA.

In order to let other tools access our approach, we exposed BORA as a REST API by using Spring Boot. By using swagger[14] we outlined all endpoints that trigger the functionalities of BORA in a JSON file. So far, we have integrated BORA to zAppDev LCDP. In order to let zAppDev access BORA (or any other tool), we had to host BORA encoded as a jar file and also BORAs' repository in a Linux server.

From the zAppDev perspective, when the users want to use BORA for getting modeling recommendations (see Fig. 12) they have just to click the "Business Object Suggester" button on the modeling canvas. BORA then provides recommendations to the user. Once the user has selected the desired recommended classes/attributes, they will be automatically integrated into the zAppDev platform. In the class integration part, not only the classes will be integrated, but also the respective class attributes, connection type, connection name, and also the multiplicity that the recommended and selected class has to the class selected for recommendations. An overview of BORA used on zAppDev for class recommendations is depicted in Fig. 12. Up to three classes[15] from the current state of the model under construction (a) can be simultaneously selected for recommendations. In b) BORA provides recommendations from which user can perform a multiple selection. The classes selected for recommendation are then integrated together with all the information available in the repository and shown in the modeling canvas pressing the OK button (c). Depending on how many classes a user selects to ask for recommendations, the respective N-gram endpoint is triggered, i.e., one selected class triggers 1-gram endpoint will be triggered, two or three selected

[10]https://zappdev.com
[11]https://jena.apache.org/documentation/tdb/
[12]https://github.com/iliriani/BORA_Ecore.
[13]https://mvnrepository.com/
[14]https://swagger.io/
[15]BORA provides 1-gram, 2grams, and 3-grams at the time of writing.

classes trigger the 2-gram or 3-gram endpoints, respectively. If the user asks for recommendations without selecting any class, she receives recommendations for *island classes*, i.e., disconnected from any other class, within the respective domain. And finally, if the user presses the ”Suggest attributes” button next to the ”Business Object Suggester”, she receives recommendations for adding attributes within that class.

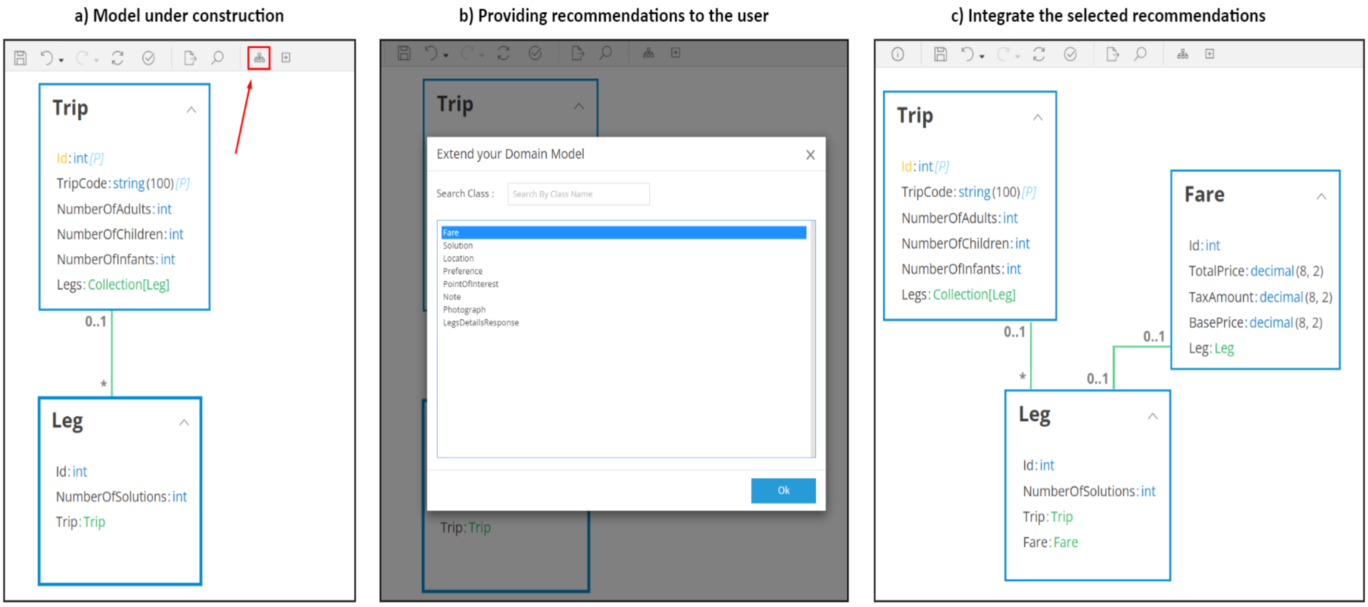


Figure 12: A screenshot of BORA’s integration with zAppDev

5 Conclusion and Future Work

This deliverable represents the concepts for mining low-code engineering repositories. We have divided the mining process into two main parts: operation-based mining and state-based mining. We have outlined in this deliverable the concepts for these parts, we have outlined the state-of-the-art conducting a preliminary mapping study on model reuse, and finally, we outlined our approaches related to these parts and their respective proof of concept implementations. Based on the performance of the approaches after the integration part, we believe that these approaches are promising for future studies on process-based and state-based mining.

As future work, we aim to combine operation-based mining and state-based mining in order to provide tailored-made suggestions to the user during the modeling process by considering the users’ logs and the state of the models. We aim to generate probabilistic graph models that abstract the structural changes done on a model as operational data for each user.

Moreover, we aim at investigating the potential contributions and challenges of *process mining-as-s-service*. It may represent a potential horizontal lightweight integration mechanism across knowledge bases collected from heterogeneous LCDPs, both in term of targeted application domains and (modeling) technologies. Nevertheless, process mining-as-a-service, by collecting and analyzing user-related data, is expected to raise privacy and intellectual property challenges.

References

[1] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-Driven Software Engineering in Practice: Second Edition. *Synthesis Lectures on Software Engineering*, 2017.

[2] Google. Google AppSheet. https://about.appsheet.com/, 2022.

[3] Amazon. Amazon Honeycode. https://www.honeycode.aws/, 2022.

[4] Davide Di Ruscio, Dimitris Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi, and Manuel Wimmer. Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 2022.

[5] Paul Vincent, Kimihiko Iijima, Mark Driver, Jason Wong, and Yefim Natis. Licensed for Distribution Magic Quadrant for Enterprise Low-Code Application Platforms. pages 1–34, 2019.

[6] Volodymyr Leno, Artem Polyvyanyy, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. Robotic process mining: vision and challenges. *Business & Information Systems Engineering*, 63(3):301–314, 2021.

[7] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. Supporting the understanding and comparison of low-code development platforms. *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, (August):171–178, 2020.

[8] IEEE 1849-2016 xes standard. https://www.xes-standard.org/, Aug 2021. Accessed: 2022-03-03.

[9] Prom tools. https://www.promtools.org/. Accessed: 2022-03-03.

[10] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, page 68–77, Swindon, GBR, 2008. BCS Learning & Development Ltd.

[11] Victor R Basili, Gianluigi Caldiera, and H Dieter Rombach. The goal question metric approach. *Encyclopedia of Software Engineering*, 2:528–532, 1994.

[12] Ángel Mora Segura and Juan de Lara. EXTREMO: An Eclipse plugin for modelling and meta-modelling assistance. *Science of Computer Programming*, 2019.

[13] Önder Babur, Loek Cleophas, and Mark van den Brand. Metamodel clone detection with SAMOS, 2019.

[14] Yanis Hattab. Extending Concern-Oriented Reuse to Existing Modelling Languages. (April), 2020.

[15] Tanumoy Pati, Dennis C. Feiock, and James H. Hill. Proactive modeling: Auto-generating models from their semantics and constraints. *SPLASH 2012: DSM 2012 - Proceedings of the 2012 ACM Workshop on Domain-Specific Modeling*, pages 7–12, 2012.

[16] Arvind Nair. Integrating Recommender Systems into Domain Specific Modeling Tools. *ProQuest Dissertations and Theses*, (May):120, 2017.

[17] Henning Agt-Rickauer. Supporting Domain Modeling with Automated Knowledge Acquisition and Modeling Recommendation. (November 2019):184, 2020.

[18] Gwendal Daniel. *Efficient persistence, query, and transformation of large models*. Theses, Ecole nationale supérieure Mines-Télécom Atlantique, November 2017.

[19] Lissette Almonte, Esther Guerra, Iván Cantador, and Juan de Lara. Recommender systems in model-driven engineering. *Software and Systems Modeling*, 21(1):249–280, 2022.

[20] Michael Fellmann, Novica Zarvic, Dirk Metzger, and Agnes Koschmider. Requirements catalog for business process modeling recommender systems (extended abstract). *CEUR Workshop Proceedings*, 1701:44–47, 2016.

[21] Abdullah Imad Abdullah Elkindy. Survey of business process modeling recommender systems. masterthesis, Universität Koblenz-Landau, Universitätsbibliothek, 2019.

[22] Krzysztof Kluza, Mateusz Baran, Szymon Bobek, and Grzegorz Nalepa. Overview of recommendation techniques in business process modeling. volume 1070, pages 46–57, 01 2013.

[23] A. Kusel, J. Schönböck, M. Wimmer, G. Kappel, W. Retschitzegger, and W. Schwinger. Reuse in model-to-model transformation languages: are we there yet? *Software & Systems Modeling*, 14(2):537–572, May 2015.

[24] Garcia-Dominguez A. and Bencomo N. Non-human modelers: Challenges and roadmap for reusable self-explanation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10748 LNCS:161 – 171, 2018. Cited by: 4; All Open Access, Green Open Access.

[25] Angela Barriga, Adrian Rutle, and Rogardt Heldal. AI-powered model repair: an experience report–lessons learned, challenges, and opportunities. *Software and Systems Modeling*, 2022.

[26] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Andrea Marrella, Massimo Mecella, and Allar Soo. Automated discovery of process models from event logs: Review and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):686–705, 2019.

[27] Daniel Schuster, Sebastiaan J. van Zelst, and Wil M.P. van der Aalst. Utilizing domain knowledge in data-driven process discovery: A literature review. *Computers in Industry*, 137:103612, 2022.

[28] Ecoretools - graphical modeling for ecore. https://www.eclipse.org/ecoretools/doc/, 2022. Accessed: 2022-03-31.

[29] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[30] Willibrordus Martinus Pancratius van der Aalst. *Process Mining: Data Science in Action*. Springer, 2nd edition, 2016.

[31] IEEE standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. Standard, IEEE, 2016.

[32] Ocel standard. http://ocel-standard.org/, 2022. Accessed: 2022-03-31.

[33] Boudewijn F van Dongen and Wil MP Van der Aalst. A meta model for process mining data. *EMOI-INTEROP*, 160:30, 2005.

[34] Wil Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[35] 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 Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets 2005*, pages 444–454, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[36] Process mining and execution management software celonis. https://www.celonis.com/, 2022. Accessed: 2022-03-31.

[37] Process mining and automated process discovery software for professionals - fluxicon disco. https://www.fluxicon.com/disco/, 2022. Accessed: 2022-03-31.

[38] Michele Chinosi and Alberto Trombetta. Bpmn: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134, 2012.

[39] W. M. P. VAN DER AALST. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 08(01):21–66, 1998.

[40] S.J.J. Leemans. *Robust process mining with guarantees*. PhD thesis, Mathematics and Computer Science, May 2017. Proefschrift.

[41] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, 1 edition, 2013.

[42] van der Aalst W.M.P. Structural characterizations of sound workflow nets. *Computing Science Reports*, (23):18 – 22, 1996.

[43] Francesco Bedini, Ralph Maschotta, and Armin Zimmermann. A generative approach for creating eclipse sirius editors for generic systems. In *2021 IEEE International Systems Conference (SysCon)*, pages 1–8, 2021.

[44] Eclipse sirius documentation. https://www.eclipse.org/sirius/doc/. Accessed: 2022-03-23.

[45] Sven Jäger, Ralph Maschotta, Tino Jungebloud, Alexander Wichmann, and Armin Zimmermann. Creation of domain-specific languages for executable system models with the eclipse modeling project. pages 1–8, 04 2016.

[46] Vladimir Viyović, Mirjam Maksimović, and Branko Perisić. Sirius: A rapid development of dsm graphical editor. In *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*, pages 233–238, 2014.

[47] Yaowen Chen. Comparison of Graph Databases and Relational Databases When Handling Large-Scale Social Data. page 82, 2016.

[48] Adam Grzech, Leszek Borzemski, Jerzy Świątek, and Zofia Wilimowska. Preface. *Advances in Intelligent Systems and Computing*, 430(October):V–vi, 2016.

[49] Ora Lassila and Ralph R Swick. Resource description framework (RDF) model and syntax specification. World Wide Web Consortium Recommendation. (October), 1999.

[50] Jimmy Lin. N-Gram Language Models N-Gram Language Models. 2009.

[51] A. Kusel, J. Schönböck, M. Wimmer, G. Kappel, W. Retschitzegger, and W. Schwinger. Reuse in model-to-model transformation languages: are we there yet? *Software and Systems Modeling*, 2015.

[52] Charles W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2):131–183, jun 1992.

[53] Henning Agt-Rickauer, Ralf Detlef Kutsche, and Harald Sack. DoMoRe – A recommender system for domain modeling. *MODELSWARD 2018 - Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, 2018-Janua(January):71–82, 2018.

[54] Jean Bézivin, Ivan Kurtev, et al. Model-based technology integration with the technical space concept. In *Metainformatics Symposium*, volume 20, pages 44–49. Citeseer, 2005.

[55] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. 01 2007.

[56] Bob DuCharme. *Learning SPARQL (free chapters)*. 2013.

[57] Hugo Bruneliere, Jordi Cabot, Grégoire Dupé, and Frédéric Madiot. MoDisco: a Model Driven Reverse Engineering Framework. *Information and Software Technology*, 56(8):1012–1032, August 2014.

A Selected Studies

Table 5: Outline of the 97 selected approaches after conducting the MS for finding approaches for enabling and supporting model reuse.

Approach	Summary	Model reused by	Technique	Model type	Contribution type	Year
[1]	Persists all models in a single model, and uses a query catalog to facilitate the search and reuse of model artifacts.	Query facility	Abstraction, OCL	*	Tool	2018
[2]	Presents a meta-model assistant that is capable to persist different source models and query them in a uniform way.	Query facility	Abstraction, OCL	*	Tool	2016
[3]	Presents Extremo - an Eclipe plugin for heterogeneous model assistence	Query facility	Abstraction, OCL	*	Tool	2019
[4]	Presents Moogle, a metamodel-based model search engine	Query facility	Metamodel information, Apache Lucene	*	Tool	2012
[5]	Presents SAMOS, a clone detection approach by using N-grams, VSM and clustering	Clone detection	n-grams, VSM, Clustering	Ecore	Tool	2018
[6]	Presents an extension of SAMOS (Statistical Analysis of MOdelS) to clone detection on Ecore metamodels.	Clone detection	n-grams, VSM, Clustering	Ecore	Tool	2019
[7]	Uses semantically related words to predict recommendations.	Recommendation system	n-grams	Ecore, UML	Tool	2018
[8]	Present DoMoRe, a domain model recommendation approach based on N-grams.	Recommendation system	n-grams	Ecore, UML	Tool	2018
[9]	Uses a DSL to configure the recommender and encodes models to a utility matrix for enabling recommendations.	Recommendation system	Abstraction, Utility matrix	Ecore	Algorithm	2020
[10]	Provides proactive modeling by combining six different recommendation parameters.	Recommendation system	OCL, action presence-based recommendation	UML	Tool	2021
[11]	Explains how to automatically foresees model assistance based on meta-model constraints	Recommendation system	OCL	UML	Tool	2012
[12]	Uses a predefined catalog of modeling activities.	Recommendation system	Predefined catalog of modeling activities	UML	Algorithm	2013
[13]	Presents RapMOD, a model auto-completion tool that transforms model activities into complex-event processing (CEP) rules, performs partial recognition and pattern comparison.	Recommendation system	Complex event patterns rules, Partial pattern recognition, Pattern comparison	UML	Tool	2021
[14]	Uses business process mining and distance metrics.	Recommendation system	Patterns mining and persisting on DB, Pattern comparison	BPM	Algorithm	2017
[15]	Uses business process mining and distance metrics.	Recommendation system	Graph mining, Graph edit distance (GED)	BPM	Algorithm	2012
[16]	Uses business process mining and distance metrics.	Recommendation system	Process mining, DFS, String edit distance (SED)	BPM	Algorithm	2014
[17]	Uses a design science method and multi-criteria-based ranking.	Recommendation system	Process execution logs	BPM	Tool	2011
[18]	Predicts recommendations from the meta-models and their constraints.	Recommendation system	ALLOY constraints, CNF formulas	*	Tool	2009

Continued on next page

Table 5 – *Continued from previous page*

Approach	Summary	Model reused by	Technique	Model type	Contribution Type	Year
[19]	Algorithm to predict recommendations from the meta-models and their constraints	Recommendation system	Constraint Logic Programming (CLP)	FSM model	Algorithm	2007
[20]	Using UML hierarchical-clustering with CACB and according to semantic similarity provide recommendations.	Recommendation system	Ant-based hierarchical clustering, semantic similarity	UML	Algorithm	2016
[21]	Search for UML sequence diagram to be reused based on users' preferences.	Recommendation system	CB and BoW algorithm	UML	Algorithm	2016
[22]	Uses a predefined catalog of modeling activities to match and provide recommendations.	Recommendation system	Activity patterns	UML	Tool	2014
[23]	Uses a combination of three techniques and a repository of processes for recommendations.	Recommendation system	Process context-based analysis, pre-and post-condition analysis, non-functional property analysis	BPM	Algorithm	2008
[24]	Uses a business process library to persist all previous processes (SQL) and match the new ones with the BPL for the recommendation process.	Recommendation system	SQL, BPL	BPM	Tool	2011
[25]	Applies process mining by looking at a process log and partial execution to predict future processes.	Recommendation system	Process mining of process log, Partial execution	BPM	Tool	2007
[26]	A road-map on how to implement three different ML techniques for process recommendations for business processes.	Recommendation system	Semantic repository Machine Learning	BPM	Tool	2019
[27]	Presents a representation-based learning method to train the extracted relations and predicts recommendations for business process models.	Recommendation system	Representation learning, Vector space	BPM	Tool	2018
[28]	Provides model recommendations based on the meta-model constraints - proactive modeling technique.	Recommendation system	Metamodel syntax OCL	UML, EMF	Tool	2017
[22]	Predicts recommendations from the meta-models and their constraints.	Recommendation system	Prolog constraint solver	FSM model	Tool	2007
[30]	Provides business process recommendations based on their keywords and by adding term frequency and structural correctness frequency.	Query facility	Keywords, Term frequency, Structural correction frequency	BPM	Tool	2008
[31]	Provides a heterogeneous model discovery framework for building business process models.	Query facility	Enterprise ontology, Graph theory - digraphs	*	Tool	2007
[32]	Provides a set of business process patterns that can be reused through some algorithms.	Recommendation system	Process patterns	BPM	Tool	2013
[33]	Describes a business process modeling editor, which assists users in purpose-oriented modeling of processes. It uses queries and process tagging for recommendations	Query facility	User Profiles Process tagging	BPM	Tool	2011
[34]	Uses Bayesian Networks to provide recommendations on business process modeling	Recommendation system	Bayesian networks	BPM	Algorithm	2013

Continued on next page

Table 5 – *Continued from previous page*

Approach	Summary	Model reused by	Technique	Model type	Contribution Type	Year
[35]	Provides a data model for abstracting business process data for using recommendation systems.	Recommendation system	Data model	BPM	Tool	2016
[36]	Provides an extensible framework for model recommendations.	Recommendation system	Integration of Rec. strategies, WordWeb	EMF, UML	Tool	2013
[37]	Provides a framework for model recommendations, the requirements, architecture and tool support.	Recommendation system	Integration of Rec. strategies	EMF, UML	Tool	2014
[38]	Provides a repository for model-driven development.	Query facility	Drupal content management platform	*	Tool	2012
[15]	Provides a proactive modeling technique by using the model semantics and constraints	Recommendation system	OCL	EMF, UML	Tool	2012
[40]	Describes how OCL can be adapted and leveraged to guide users toward correct solutions using visual cues.	Recommendation system	OCL	Batch processes	Tool	2008
[16]	Provides the integration of recommendation systems into the domain-specific modeling tool.	Recommendation system	OCL	EMF, UML	Tool	2017
[42]	Presents an approach based on a pre-trained language model for providing metamodel concepts recommendations.	Recommendation system	Pre-trained lamguage model	Ecore	Approach	2022
[43]	A structure-based search engine for heterogeneous models.	Query facility	Indexing, Bag of paths	*	Tool	2020
[44]	Presents an approach for recommending Simulink libraries elements for DSL. It uses data mining techniques and compares them: association rules and collaborative filtering.	Recommendation system	Data mining	MATLAB/Simulink	Algorithm	2012
[45]	Uses graph theory for automatic detection of clones in large models. Here the approach is implemented for MATLAB/Simulink model clone detection.	Clone detection	Graph theory, Similarity heuristics	MATLAB/Simulink	Tool	2008
[46]	A clone detection tool for MATLAB/Simulink models. The core of ModelCD is based on two novel graph-based clone detection algorithms: eScan and aScan.	Clone detection	Graph theory, sScan, aScan	MATLAB/Simulink	Tool	2009
[47]	A graph-based approach for UML clone detection	Clone detection	Similarity heuristics.	UML	Tool	2011
[48]	A tool based on graph theory for Simulink clone detection inspection.	Clone detection	Graph theory	MATLAB/Simulink	Tool	2010
[49]	Presents an indexing structure for business process model clone detection. It is based on graph canonization and string matching techniques.	Clone detection	Indexing, Graph Canonization	BPM	Algorithm	2011
[50]	Uses graph-theory techniques for parsing business models to graphs and detecting clones.	Clone detection	Graph theory	BM	Algorithm	2012

Continued on next page

Table 5 – *Continued from previous page*

Approach	Summary	Model reused by	Technique	Model type	Contribution Type	Year
[51]	Propose an indexing structure(RPSDAG) for clone detection in process model repositories with model decomposition based on graph canonization and string matching.	Clone detection	Decompose into SESE Graph canonization	BPM	Algorithm	2013
[52]	An approach for UML clone detection. Converts UML to XMI. The similarity between the two fragments is reported as a clone.	Clone detection	Subtree comparison	UML	Algorithm	2012
[53]	Presents a clone detection algorithm for UML domain models based on similarity heuristics.	Clone detection	Similarity heuristics	UML	Tool	2015
[54]	Proposes an optimized path-based model clone detection algorithm (OPMCD) for MATLAB/SIMULINK models.	Clone detection	Graph theory	MATLAB/SIMULINK	Algorithm	2014
[55]	A knowledge-based recommender system based on property graphs and meta graphs. It also provides a schema for model recommendation production for operation-based model recommenders	Recommendation system	Property graphs	UML, EMF	Tool	2018
[56]	Present the design and construction of a model reuse repository	Query facility	Domain-specific software architecture	Not specified	Tool	1995
[57]	Presents a framework on how to achieve disciplined reuse and evolution of UML model components by reuse contracts.	Not specified	Disciplined reuse /Reuse contract	UML	Concept	1998
[58]	Propose a metamodel package assembly called reuse and generalize based on MOF.	Import	MOF	UML	Concept	2005
[59]	Explains how to provide context-aware domain model recommendations based on the semantic relation of terms.	Recommendation system	N-grams, Term Semantic related network	EMF	Approach	2020
[60]	Depicts the idea of a reference framework for building an intelligent modeling assistant - RF-IMA.	Recommendation system	Not specified	*	Concept	2020
[61]	Present a detailed, level-wise definition for the properties of RF-IMA to enable a better understanding, comparison, and selection of existing and future IMAs.	Recommendation system	Not specified	*	Concept	2020
[62]	Proposes an approach to seamlessly integrate micro machine learning units into domain modeling, expressed in a single type of model, based on one modeling language.	Recommendation system	ML	Own modeling language	Algorithm	2019
[63]	Presents a plan for developing an modeling assistant and investigating its support to the modeler.	Recommendation system	AI	Not specified	Concept	2019
[64]	Presents the integration of Hawk with ECL.	Query facility	ECL, Hawk	EMF	Tool	2014
[65]	Provides an outline of the main scalability challenges in MDE, including scalable repositories and queries.	Query facility	ECL, Hawk	EMF, UML	Concept	2015

Continued on next page

Table 5 – *Continued from previous page*

Approach	Summary	Model reused by	Technique	Model type	Contribution Type	Year
[66]	Provided a model persistence framework that enables model storage into multiple data sources.	Query facility	ECL, Hawk	EMF	Tool	2017
[18]	Presents a scalable modeling framework by providing a persistence framework, an efficient query approach, and a model transformation solution	Query facility	OCL, NeoEmf	EMF, UML	Algorithm	2017
[68]	Presents a tool SYSML2RSHP that abstracts SysML model to the relationship model (XML), indexes them, and is capable to query them.	Query facility	CAKE Framework	SysML	Tool	2017
[69]	Examines two different techniques for indexing and searching model repositories, with a focus on Web development projects encoded in a DSL.	Query facility	Indexing, TF/IDF, Graph theory/ A*	WebML/XML	Algorithm	2014
[70]	Explore the use of graph-based similarity search as a tool for expressing queries over model repositories, uniformly represented as collections of labeled graphs.	Query facility	Graph theory/A*	WebML/XML	Algorithm	2011
[71]	Outlines an approach on how to query UML model using OCL.	Query facility	OCL	UML	Concept	2001
[72]	Investigates the use of classical Information Retrieval techniques for easing the discovery of useful information from past projects.	Query facility	Indexing IR techniques	WebML/XML	Algorithm	2010
[73]	Studies the problem of querying UML class diagrams.	Query facility	Datalog	UML	Concept	2012
[74]	Present CORE - collaborative ontology reuse and evaluation tool.	Query facility	NLP techniques	OWL	Tool	2006
[75]	Outlines a novel approach to model search that leverages the repository structure into a megamodel.	Query facility	MDEForge, Lucene, OCL, Megamodel	EMF	Tool	2018
[76]	Presents a DSL for model querying and manipulation - MoScript.	Query facility	OCL, Megamodel	*	Tool	2011
[77]	Uses LinkedIn profiles to provide recommendations for BP modeling.	Recommendation system	LinkedIn, ATL	BP	Algorithm	2020
[78]	Presents the challenges for reuse in collaborative modeling environment.	Query facility	Not specified	Not specified	Concept	2018
[79]	Presents a canonical set of reuse interfaces applicable for different artifacts.	Not specified	Not specified	Not specified	Concept	2016
[14]	Presents how to improve the support for modeling reuse and language tailoring by extending the Concern Oriented Reuse (CORE) modeling framework.	Query facility	Concern oriented	EMF	Tool	2020
[81]	Presents SIMONE-an adaptation of the mature text-based code clone detector NICAD to the efficient identification of structurally meaningful near-miss subsystem clones in graphical models.	Clone detection	NICAD	MATLAB/SIMULINK	Tool	2012

Continued on next page

Table 5 – *Continued from previous page*

Approach	Summary	Model reused by	Technique	Model type	Contribution Type	Year
[82]	Presents a clone detection approach for embedded systems relying on the semantic equivalence (not syntactic) based on graph transformation.	Clone detection	Graph transformation	MATLAB/SIMULINK, ASCET-SD, Esterel, Lustre	Algorithm	2011
[83]	Provides techniques for approximate clone detection based on clustering algorithms.	Clone detection	DBSCAN, HAC	BPM	Approach	2015
[84]	Presents an approach for identifying near-miss interaction clones in reverse-engineered UML behavioural models.	Clone detection	NICAD	UML, BM	Algorithm	2013
[85]	Presents a model clone approach used for code clone detection. This approach uses the CK Metrics suite for clone detection.	Clone detection	CK Metrics Suite	Not specified	Approach	2015
[86]	Provides an extensible framework for model recommendations. Explains the architecture on how to integrate different UIs and recommendations strategies.	Recommendation system	WordWeb	Ecore	Tool	2013
[87]	Provides a model library approach to improve the quality of models so they can be reused.	Recommendation system	Graphs, Quality dimensions	UML	Tool	2015
[88]	Explains the integration of ”reuse contracts” to enable model reuse and evolution.	Not specified	Reuse contract	UML	Concept	1998
[89]	Introduces the concept of model Library to store the reused model artifacts.	Not specified	Library concept	Not specified	Concept	2010
[90]	Provides an approach for isolating and reusing template instances of UML.	Not specified	Templates, OCL	UML	Tool	2016
[91]	Present an approach of reusing models of different overlapping concepts but which should have the same meta-model.	Not specified	Model transformation	UML	Approach	2002
[92]	Outlines the concept of building a virtual modeling assistant based on clone detection and matching.	Clone detection	ML Simone clone detection	MATLAB/SIMULINK	Concept	2019
[93]	Presents the main properties that have to be taken into account for assessing and comparing IMAs.	Not specified	Not specified	*	Concept	2020
[94]	A GNN-based recommender system for facilitating the modeling process by assisting the specification of metamodels and models.	Recommendation system	ML, Graph-kernel	*	Approach	2021
[95]	Presents an approach for reusing type-safe templates on a low-code platform (Outsystems).	Screen templates	Type safe template language.	Outsytem mod.	Approach	2021

Appendix References

[1] Mora Segura Ángel, Juan de Lara, Patrick Neubauer, and Manuel Wimmer. Automated modelling assistance by integrating heterogeneous information sources. *Computer Languages, Systems & Structures*, 53:90–120, 2018.

[2] Mora Segura A., Pescador A., De Lara J., and Wimmer M. An extensible meta-modelling assistant. page 79 – 88, 2016. Cited by: 11.

[3] Ángel Mora Segura and Juan de Lara. EXTREMO: An Eclipse plugin for modelling and meta-modelling assistance. *Science of Computer Programming*, 2019.

[4] Daniel Lucrédio, Renata P. de M. Fortes, and Jon Whittle. MOOGLE: a metamodel-based model search engine. *Software & Systems Modeling*, 11(2):183–208, May 2012.

[5] Önder Babur. Clone Detection for Ecore Metamodels using N-grams:. In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, pages 411–419, Funchal, Madeira, Portugal, 2018. SCITEPRESS - Science and Technology Publications.

[6] Önder Babur, Loek Cleophas, and Mark van den Brand. Metamodel clone detection with SAMOS, 2019.

[7] Henning Agt-Rickauer, Ralf-Detlef Kutsche, and Harald Sack. Automated recommendation of related model elements for domain models. In *MODELSWARD*, 2018.

[8] Henning Agt-Rickauer, Ralf-Detlef Kutsche, and Harald Sack. DoMoRe – A Recommender System for Domain Modeling:. In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, pages 71–82, Funchal, Madeira, Portugal, 2018. SCITEPRESS - Science and Technology Publications.

[9] Lissette Almonte, Iván Cantador, Esther Guerra, and Juan de Lara. Towards automating the construction of recommender systems for low-code development platforms. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 1–10, Virtual Event Canada, October 2020. ACM.

[10] Arvind Nair, Xia Ning, and James H. Hill. Using recommender systems to improve proactive modeling. *Software and Systems Modeling*, 20(4):1159–1181, August 2021.

[11] Tanumoy Pati, Dennis C. Feiock, and James H. Hill. Proactive modeling: auto-generating models from their semantics and constraints. In *Proceedings of the 2012 workshop on Domain-specific modeling - DSM '12*, page 7, Tucson, Arizona, USA, 2012. ACM Press.

[12] Tobias Kuschke and Patrick Mäder. Pattern-based auto-completion of UML modeling activities. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 551–556, Vasteras Sweden, September 2014. ACM.

[13] Patrick Mader, Tobias Kuschke, and Mario Janke. Reactive Auto-Completion of Modeling Activities. *IEEE Transactions on Software Engineering*, 47(7):1431–1451, July 2021.

[14] Shuiguang Deng, Dongjing Wang, Ying Li, Bin Cao, Jianwei Yin, Zhaohui Wu, and Mengchu Zhou. A Recommendation System to Facilitate Business Process Modeling. *IEEE Transactions on Cybernetics*, 47(6):1380–1394, June 2017.

[15] Bin Cao, Jianwei Yin, Shuiguang Deng, Dongjing Wang, and Zhaohui Wu. Graph-based workflow recommendation: on improving business process modeling. In *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, page 1527, Maui, Hawaii, USA, 2012. ACM Press.

[16] Ying Li, Bin Cao, Lida Xu, Jianwei Yin, Shuiguang Deng, Yuyu Yin, and Zhaohui Wu. An efficient recommendation method for improving business process modeling. *IEEE Transactions on Industrial Informatics*, 10(1):502–513, 2014.

[17] Mukhammad Andri Setiawan, Shazia Wasim Sadiq, and Ryan Kirkman. Facilitating business process improvement through personalized recommendation. In *BIS*, 2011.

[18] Sagar Sen, Benoit Baudry, and Doina Precup. Partial model completion in model driven engineering using constraint logic programming. In *International Conference on the Applications of Declarative Programming*, 2007.

[19] Sagar Sen, Benoît Baudry, and Doina Precup. Partial model completion in model driven engineering using constraint logic programming. 2007.

[20] Akil Elkamel, Mariem Gzara, and Hanene Ben-Abdallah. An UML class recommender system for software design. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8, Agadir, Morocco, November 2016. IEEE.

[21] Thaciana G. O. Cerqueira, Franklin Ramalho, and Leandro Balby Marinho. A Content-Based Approach for Recommending UML Sequence Diagrams. pages 644–649, July 2016.

[22] Tobias Kuschke and Patrick Mäder. Pattern-based auto-completion of uml modeling activities. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, page 551–556, New York, NY, USA, 2014. Association for Computing Machinery.

[23] Matthias Born, Christian Brelage, Ivan Markovic, Daniel Pfeiffer, and Ingo Weber. Auto-completion for executable business process models. *Lecture Notes in Business Information Processing*, 17 LNBIP:510–515, 2009.

[24] Karol Wieloch, Agata Filipowska, and Monika Kaczmarek. Autocompletion for business process modelling. In Witold Abramowicz, Leszek Maciaszek, and Krzysztof Wkecel, editors, *Business Information Systems Workshops*, pages 30–40, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[25] Helen Schonenberg, Barbara Weber, Boudewijn Van Dongen, and Wil Van Der Aalst. Supporting flexible processes through recommendations based on history. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5240 LNCS(2007):51–66, 2008.

[26] Emanuele Laurenzi, Knut Hinkelmann, Stephan Jüngling, Devid Montecchiari, Charuta Pande, and Andreas Martin. Towards an Assistive and Pattern Learning-driven Process Modeling Approach. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, page 6, 2019.

[27] Huaqing Wang, Lijie Wen, Li Lin, and Jianmin Wang. Rl-recommender: A representation-learning-based recommendation method for business process modeling. In Claus Pahl, Maja Vukovic, Jianwei Yin, and Qi Yu, editors, *Service-Oriented Computing*, pages 478–486, Cham, 2018. Springer International Publishing.

[28] Tanumoy Pati, Sowmya Kolli, and James H. Hill. Proactive modeling: a new model intelligence technique. *Software & Systems Modeling*, 16(2):499–521, May 2017.

[29] Sagar Sen, Benoit Baudry, and Hans Vangheluwe. Domain-specific model editors with model completion. volume 5002, pages 259–270, 09 2007.

[30] Thomas Hornung, Agnes Koschmider, and Georg Lausen. Recommendation Based Process Modeling Support: Method and User Experience. In Qing Li, Stefano Spaccapietra, Eric Yu, and Antoni Olivé, editors, *Conceptual Modeling - ER 2008*, pages 265–278, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[31] Aditya Ghose, George Koliadis, and Arthur Chueng. Process Discovery from Model and Text Artefacts. In *2007 IEEE Congress on Services (Services 2007)*, pages 167–174, Salt Lake City, UT, USA, July 2007. IEEE.

[32] Agnes Koschmider and Hajo A. Reijers. Improving the process of process modelling by the use of domain process patterns. *Enterprise Information Systems*, 9(1):29–57, January 2015.

[33] Agnes Koschmider, Thomas Hornung, and Andreas Oberweis. Recommendation-based editor for business process modeling. *Data & Knowledge Engineering*, 70(6):483–503, June 2011.

[34] Szymon Bobek, Mateusz Baran, Krzysztof Kluza, and Grzegorz J Nalepa. Application of Bayesian Networks to Recommendations in Business Process Modeling. In *AIBP@ AI* IA*, pages 41–50, 2013.

[35] Michael Fellmann, Dirk Metzger, and Oliver Thomas. Data model development for process modeling recommender systems. *Lecture Notes in Business Information Processing*, 267:87–101, 2016.

[36] Andrej Dyck, Andreas Ganser, and Horst Lichter. Enabling Model Recommenders for Command-Enabled Editors. In *MDEBE@MoDELS*, 2013.

[37] Andrej Dyck, Andreas Ganser, and Horst Lichter. A framework for model recommenders requirements, architecture and tool support. In *MODELSWARD 2014 - Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, 2014.

[38] Robert B. France, James M. Bieman, Sai Pradeep Mandalaparty, Betty H. C. Cheng, and Adam Jensen. Repository for Model Driven Development (ReMoDD). In *2012 34th International Conference on Software Engineering (ICSE)*, pages 1471–1472, Zurich, June 2012. IEEE.

[39] Tanumoy Pati, Dennis C. Feiock, and James H. Hill. Proactive modeling: Auto-generating models from their semantics and constraints. *SPLASH 2012: DSM 2012 - Proceedings of the 2012 ACM Workshop on Domain-Specific Modeling*, pages 7–12, 2012.

[40] Jules White, D.C. Schmidt, Andrey Nechypurenko, and Egon Wuchner. Model intelligence: an approach to modeling guidance. *Upgrade*, 9(2):22–28, 2008.

[41] Arvind Nair. Integrating Recommender Systems into Domain Specific Modeling Tools. *ProQuest Dissertations and Theses*, (May):120, 2017.

[42] Martin Weyssow, Houari Sahraoui, and Eugene Syriani. Recommending metamodel concepts during modeling activities with pretrained language models. *Software and Systems Modeling*, February 2022.

[43] José Antonio Hernández López and Jesús Sánchez Cuadrado. MAR: a structure-based search engine for models. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 57–67, Virtual Event Canada, October 2020. ACM.

[44] Lars Heinemann. Facilitating reuse in model-based development with context-dependent model element recommendations. In *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*, pages 16–20, Zurich, Switzerland, June 2012. IEEE.

[45] Florian Deissenboeck, Benjamin Hummel, Elmar Jürgens, Bernhard Schätz, Stefan Wagner, Jean-François Girard, and Stefan Teuchert. Clone detection in automotive model-based development. In *Proceedings of the 13th international conference on Software engineering - ICSE '08*, page 603, Leipzig, Germany, 2008. ACM Press.

[46] Nam H. Pham, Hoan Anh Nguyen, Tung Thanh Nguyen, Jafar M. Al-Kofahi, and Tien N. Nguyen. Complete and accurate clone detection in graph-based models. In *2009 IEEE 31st International Conference on Software Engineering*, pages 276–286, Vancouver, BC, Canada, 2009. IEEE.

[47] Harald Störrle. Towards clone detection in uml domain models. *Software & Systems Modeling*, 12:307–329, 2011.

[48] Florian Deissenboeck, Benjamin Hummel, and Elmar Juergens. Code clone detection in practice. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, volume 2, page 499, Cape Town, South Africa, 2010. ACM Press.

[49] Reina Uba, Marlon Dumas, Luciano García-Bañuelos, and Marcello La Rosa. Clone Detection in Repositories of Business Process Models. In Stefanie Rinderle-Ma, Farouk Toumani, and Karsten Wolf, editors, *Business Process Management*, pages 248–264, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[50] Chathura C. Ekanayake, Marlon Dumas, Luciano García-Bañuelos, Marcello La Rosa, and Arthur H.M. Ter Hofstede. Approximate clone detection in repositories of business process models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7481 LNCS:302–318, 2012.

[51] Marlon Dumas, Luciano García-Bañuelos, Marcello La Rosa, and Reina Uba. Fast detection of exact clones in business process model repositories. *Information Systems*, 38(4):619–633, June 2013.

[52] Dhavleesh Rattan, Rajesh Bhatia, and Maninder Singh. Model clone detection based on tree comparison. In *2012 Annual IEEE India Conference (INDICON)*, pages 1041–1046, Kochi, India, December 2012. IEEE.

[53] Harald Störrle. *Effective and Efficient Model Clone Detection*, pages 440–457. Springer International Publishing, Cham, 2015.

[54] Zhengping Liang, Yiqun Cheng, and Jianyong Chen. A Novel Optimized Path-Based Algorithm for Model Clone Detection. *Journal of Software*, 9(7):1810–1817, July 2014.

[55] Andreas Ganser. *Operation-Based Model Recommenders*. Shaker Verlag GmbH, DE, 1 edition, 2018.

[56] J. Petro, M.E. Fotta, and D.B. Weisman. Model-based reuse repositories-concepts and experience. In *Proceedings Seventh International Workshop on Computer-Aided Software Engineering*, pages 60–69, Toronto, Ont., Canada, 1995. IEEE Comput. Soc. Press.

[57] Tom Mens, Carine Lucas, and Patrick Steyaert. Supporting disciplined reuse and evolution of uml models. In *UML*, 1998.

[58] Xavier Blanc, Franklin Ramalho, and Jacques Robin. Metamodel reuse with MOF. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3713 LNCS:661–675, 2005.

[59] Henning Agt-Rickauer. Supporting domain modeling with automated knowledge acquisition and modeling recommendations. 2020. Publisher: Technische Universität Berlin.

[60] Gunter Mussbacher, Benoit Combemale, Jörg Kienzle, Silvia Abrahão, Hyacinth Ali, Nelly Bencomo, Márton Búr, Loli Burgueño, Gregor Engels, Pierre Jeanjean, Jean Marc Jézéquel, Thomas Kühn, Sébastien Mosser, Houari Sahraoui, Eugene Syriani, Dániel Varró, and Martin Weyssow. Opportunities in intelligent modeling assistance. *Software and Systems Modeling*, (June), 2020.

[61] Gunter Mussbacher, Benoit Combemale, Silvia Abrahão, Nelly Bencomo, Loli Burgueño, Gregor Engels, Jörg Kienzle, Thomas Kühn, Sébastien Mosser, Houari Sahraoui, and Martin Weyssow. Towards an assessment grid for intelligent modeling assistance. *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings*, pages 296–305, 2020.

[62] Thomas Hartmann, Assaad Moawad, François Fouquet, and Yves Le Traon. The next evolution of mde: a seamless integration of machine learning into domain modeling. *Software & Systems Modeling*, 18:1285–1304, 2017.

[63] Maxime Savary-Leblanc. Improving MBSE Tools UX with AI-Empowered Software Assistants. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 648–652, Munich, Germany, September 2019. IEEE.

[64] Konstantinos Barmpis and Dimitrios S. Kolovos. Towards scalable querying of large-scale models. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[65] Dimitrios S Kolovos, Louis M Rose, Richard F Paige, Esther M Guerra, Jesús Sánchez Cuadrado, Juan M De Lara, István Ráth, Dániel Varró, Gerson Sunyé, and Massimo Tisi. MONDO: Scalable Modelling and Model Management on the Cloud. In *STAF2015 Project Showcase*, L'Aquila, Italy, 2015.

[66] Gwendal Daniel, Gerson Sunyé, Amine Benelallam, Massimo Tisi, Yoann Vernageau, Abel Gómez, and Jordi Cabot. NeoEMF: A multi-database model persistence framework for very large models. *Science of Computer Programming*, 149:9–14, December 2017.

[67] Gwendal Daniel. *Efficient persistence, query, and transformation of large models*. Theses, Ecole nationale supérieure Mines-Télécom Atlantique, November 2017.

[68] Roy Mendieta, Jose Luis de la Vara, Juan Llorens, and Jose María Álvarez Rodríguez. Towards Effective SysML Model Reuse:. In *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development*, pages 536–541, Porto, Portugal, 2017. SCITEPRESS - Science and Technology Publications.

[69] Bojana Bislimovska, Alessandro Bozzon, Marco Brambilla, and Piero Fraternali. Textual and Content-Based Search in Repositories of Web Application Models. *ACM Transactions on the Web*, 8(2):1–47, March 2014.

[70] Bojana Bislimovska, Alessandro Bozzon, Marco Brambilla, and Piero Fraternali. Graph-Based Search over Web Application Model Repositories. In Sören Auer, Oscar Díaz, and George A. Papadopoulos, editors, *Web Engineering*, volume 6757, pages 90–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Series Title: Lecture Notes in Computer Science.

[71] D. H. Akehurst and B. Bordbar. On querying UML data models with OCL. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2185:91–103, 2001.

[72] Alessandro Bozzon, Marco Brambilla, and Piero Fraternali. Searching repositories of web application models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6189 LNCS(July):1–15, 2010.

[73] Andrea Calì, Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Querying uml class diagrams. In *FoSSaCS*, 2012.

[74] Miriam Fernández, Iván Cantador, and Pablo Castells. Core: A tool for collaborative ontology reuse and evaluation. In *EON@WWW*, 2006.

[75] Francesco Basciani, Davide Di Ruscio, Juri Di Rocco, Ludovico Iovino, and Alfonso Pierantonio. Exploring model repositories by means of megamodel-aware search operators. *CEUR Workshop Proceedings*, 2245:793–798, 2018.

[76] Wolfgang Kling, Frédéric Jouault, Dennis Wagelaar, Marco Brambilla, and Jordi Cabot. MoScript: A DSL for Querying and Manipulating Model Repositories. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Anthony Sloane, and Uwe Aßmann, editors, *Software Language Engineering*, volume 6940, pages 180–200. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. Series Title: Lecture Notes in Computer Science.

[77] Hadjer Khider, Slimane Hammoudi, and Abdelkrim Meziane. Business process model recommendation as a transformation process in MDE: Conceptualization and first experiments. *MODELSWARD 2020 - Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*, pages 65–75, 2020.

[78] Omar Alam, Jonathan Corley, Constantin Masson, and Eugene Syriani. Challenges for reuse in collaborative modeling environments. In *MoDELS*, 2018.

[79] Jörg Kienzle, Gunter Mussbacher, Omar Alam, Matthias Schöttle, Nicolas Belloir, Philippe Collet, Benoit Combemale, Julien DeAntoni, Jacques Klein, and Bernhard Rumpe. VCU: The Three Dimensions of Reuse. In Georgia M. Kapitsaki and Eduardo Santana de Almeida, editors, *Software Reuse: Bridging with Social-Awareness*, volume 9679, pages 122–137. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computer Science.

[80] Yanis Hattab. Extending Concern-Oriented Reuse to Existing Modelling Languages. (April), 2020.

[81] Manar H. Alalfi, James R. Cordy, Thomas R. Dean, Matthew Stephan, and Andrew Stevenson. Models are code too: Near-miss clone detection for Simulink models. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 295–304, Trento, Italy, September 2012. IEEE.

[82] Bakr Al-Batran, Bernhard Schätz, and Benjamin Hummel. Semantic Clone Detection for Model-Based Development of Embedded Systems. In Jon Whittle, Tony Clark, and Thomas Kühne, editors, *Model Driven Engineering Languages and Systems*, volume 6981, pages 258–272. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Series Title: Lecture Notes in Computer Science.

[83] Marcello La Rosa, Marlon Dumas, Chathura C. Ekanayake, Luciano García-Bañuelos, Jan Recker, and Arthur H.M. ter Hofstede. Detecting approximate clones in business process model repositories. *Information Systems*, 49:102–125, April 2015.

[84] Elizabeth P. Antony, Manar H. Alalfi, and James R. Cordy. An approach to clone detection in behavioural models. In *2013 20th Working Conference on Reverse Engineering (WCRE)*, pages 472–476, Koblenz, Germany, October 2013. IEEE.

[85] Aastha Saini, Vivek Sharma, and M Tech Student. Detecting model Clones using C K Metrics Suite. *International Journal of Engineering Research and General Science*, 3(3):1605–1612, 2015.

[86] Andrej Dyck, Andreas Ganser, and Horst Lichter. Enabling model recommenders for command-enabled editors. In *CEUR Workshop Proceedings*, 2013.

[87] Andreas Ganser, Horst Lichter, Alexander Roth, and Bernhard Rumpe. Staged model evolution and proactive quality guidance for model libraries. *Software Quality Journal*, 24(3):675–708, September 2016.

[88] Tom Mens, Carine Lucas, and Patrick Steyaert. Supporting Disciplined Reuse and Evolution of UML Models. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Jean Bézivin, and Pierre-Alain

Muller, editors, *The Unified Modeling Language. "UML"'98: Beyond the Notation*, volume 1618, pages 378–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. Series Title: Lecture Notes in Computer Science.

[89] Markus Herrmannsdörfer and Benjamin Hummel. Library Concepts for Model Reuse. *Electronic Notes in Theoretical Computer Science*, 253(7):121–134, September 2010.

[90] Matthieu Allon, Gilles Vanwormhoudt, Bernard Carré, Olivier Caron, Matthieu Allon, Gilles Vanwormhoudt, Bernard Carré, Olivier Caron Isolating, and Reusing Template. Isolating and Reusing Template Instances in UML To cite this version : HAL Id : hal-01327456. 2019.

[91] Tihamer Levendovszky, Gabor Karsai, Miklos Maroti, Akos Ledeczi, and Hassan Charaf. Model Reuse with Metamodel-Based Transformations. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Cristina Gacek, editors, *Software Reuse: Methods, Techniques, and Tools*, volume 2319, pages 166–178. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. Series Title: Lecture Notes in Computer Science.

[92] Matthew Stephan. Towards a Cognizant Virtual Software Modeling Assistant using Model Clones. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 21–24, Montreal, QC, Canada, May 2019. IEEE.

[93] Gunter Mussbacher, Benoit Combemale, Silvia Abrahão, Nelly Bencomo, Loli Burgueño, Gregor Engels, Jörg Kienzle, Thomas Kühn, Sébastien Mosser, Houari Sahraoui, and Martin Weyssow. Towards an assessment grid for intelligent modeling assistance. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 1–10, Virtual Event Canada, October 2020. ACM.

[94] Juri Di Rocco, Claudio Di Sipio, Davide Di Ruscio, and Phuong T. Nguyen. A GNN-based Recommender System to Assist the Specification of Metamodels and Models. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 70–81, Fukuoka, Japan, October 2021. IEEE.

[95] Hugo Lourenco, Carla Ferreira, and Joao Costa Seco. OSTRICH - A Type-Safe Template Language for Low-Code Development. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 216–226, Fukuoka, Japan, October 2021. IEEE.