[GJRR23] R. Gupta, N. Jansen, N. Regnat, B. Rumpe:
User-Centric Model-Aware Recommendations for Industrial Domain-Specific Modelling Languages.
In: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C),
pp. 330-341, IEEE, Oct. 2023.

2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) | 979-8-3503-2498-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/MODELS-C59198.2023.00064

elling methodology for providing integrated recommendations and guidance to modellers considering multiple viewpoints in heterogeneous DSMLs?

A framework integrated with the DSML that is able to capture these methods and actively provide recommendations to users is the primary aim of this research. To this end, we have developed a framework integrated with a widely used commercial modelling tool to combine aspects of methods within industrial DSMLs, and aim to provide ample support, guidance, methods, and recommendations for a more holistic modelling experience for users of various domains. Given our years of language engineering experience in industrial DSMLs for a wide variety of users in various domains using commercial graphical modelling tools such as Enterprise Architect [13], Rational Rhapsody [14], and MagicDraw [15], the main contributions of this paper are:

- We provide an architectural overview of the integration of various methods for a modelling language within a graphical modelling tool using the example of MagicDraw (section IV).
- We discuss various techniques to demonstrate active model-aware recommendations to users with the help of configurable business rules.
- We illustrate the applicability of our implementation using a real industrial use case of a function model designed in the healthcare industry to model functions in an X-ray collimator (section V).

Finally, we discuss the benefits of our approach (section VI), consider related work (section VII), and conclude the paper (section VIII).

II. Background

DSLs and DSMLs are subject to the usual challenges of maintenance and evolution, common in software engineering, as they are considered software themselves [16]. In general, a software language consists of [17], [3]: (1) an abstract syntax that contains the essential information of a model, e.g., in the form of context-free grammars or class diagrams [18]; (2) a concrete syntax used to describe the concrete representation of the models, e.g., graphical formats like tabular, box-and-line, or tree-based [4], [17], textual [5], or projectional [19]; (3) semantics, in the sense of meaning [20]; and (4) context conditions, to check the well-formedness of a language.

Industrial DSMLs require well integrated methods within the modelling environment that users can benefit from. Method engineering is the discipline of designing, constructing and adapting methods, techniques, and tools for the development of information systems [21]. Methods provide a structure needed to achieve a specific goal [22] and are therefore beneficial to any kind of user. Although method engineering for DSMLs has been discussed in the literature [23], [10], the research needs further validation in real industrial contexts. While it is often assumed that industrial users of DSMLs are modelling experts, the reality is that appropriate guidance and assistance are still needed for either using the DSML or gaining more domain expertise. Therefore, if users are provided with model information for their current modelling situation in the form of training materials, suggestions, proposals, or actionable items, it would make their modelling much more efficient. By providing a set of tasks, activities, or processes in the form of process models [24] that are aligned to their modelling, users can identify the accomplished processes and those parts of the models that are incomplete and needs further consideration.

Techniques to deploy methods and recommendations to users have been proposed in the literature to solve these challenges [25], [26]. However, these are either overly generic or often tied to their specific technological space. It is nearly impossible for language engineers to consider every single aspect of the language, including standards or guidelines, which are always specific to a domain. In contrast to a static or passive source of information, such as documents or webpages, where users must search for relevant information proactively, methods integrated within the DSML that analyse model information can be more beneficial in actively assisting modellers directly during modelling.

The benefits of integrating a method within the DSML and the modelling tool means the entire modelling experience is centred in a single modelling environment, which means users can model more effectively. Thus, there is a need to start the discussion towards defining such model-aware recommendations for industrial DSMLs that is ultimately independent of a specific implementation or a graphical modelling tool and benefits all kinds of users, novice or advanced.

III. Modelling Tool: MagicDraw

In large organisations such as Siemens AG, the methods, tools, and concepts for the development of graphical DSMLs are often tied to specific research departments. This introduces challenges in combining the modelling language, methodologies for the language, and an appropriate graphical modelling tool. Over the years, we have worked with commercial graphical modelling tools such as MagicDraw, Enterprise Architect, and Rational Rhapsody to develop and maintain graphical DSMLs across many domains. These DSMLs are used to model devices such as magnetic resonance imaging (MRI), X-ray equipments, electric motors, IT processes, and gas turbines. We have also worked with academic and industry partners to build a DSML for modelling heterogeneous viewpoints of a system in the SpesML project [27]. In this paper, we focus on MagicDraw as the choice of modelling tool in the development of graphical DSMLs.

MagicDraw is primarily based on the Unified Modelling Language (UML), provides extensions for the Systems Modelling Language (SysML), and customisations for DSML engineering. To this end, [7] describes a systematic engineering approach to develop graphical industrial DSMLs using the concept of modularised reusable DSL building blocks. These building blocks consist of reusable language components, integrated methods that describe how to use a DSML, and user experience design (UXD) considerations that help language engineers develop widely accepted industrial DSMLs. The reusable language components, completely or in part, define

the language [28]. The methods describe methodologies for the DSML that ultimately intends to help users in their modelling with training material, methodical steps, documentation, recommendations, and guidance to DSML users, which is the focus of this paper. The UXD part defines standards and usability heuristics for language engineers [29] for enhancing the user experience in designing models. These heterogeneous building blocks compose together to create the DSML. While the methods and UXD parts are conceptually independent of the language definition, we consider them as important as the language itself for improving a DSML's usability.

MagicDraw stores the language definition and its customisation in a DSML plugin artefact that is deployed in a modelling environment. This plugin consists of a MagicDraw project containing the language definition as a language profile, predefined templates for easily creating novel models, perspectives that limit or add functionalities of the modelling tool, and additional Java extensions to support customisations that enhance the default capabilities of the DSML. Some of the possible customisations are modifying the appearance of a model element, which model elements can be created on a specific model diagram, and so on. In our methodology, we utilise the customisation capabilities of MagicDraw to enhance DSMLs by dynamically providing recommendations and methods for users tailored to models at different stages during their modelling.

IV. Methodology and Realisation

This section gives a high-level overview of the architecture, configuration, and implementation of our plugin.

A. Architecture Overview

Figure 1 shows the classical three-tiered architecture [30] we use to show the applicability of providing user-centric model-aware recommendations for users. The presentation tier is the graphical user interface (UI) where users interact directly with the application and displays information relevant for the users. The UI is developed using Java Swing that gathers and displays information from the tiers underneath. The logic tier, written in Java, collects and processes information from the UI, makes the relevant requests to the data tier using business logic, i.e., a specific set of business rules, and returns specific recommendations to the presentation tier. These business rules are a set of conditions for defining specific actions in a business context. Finally, the data or the database tier is where model information including training material, documentation, and recommendations related to the models is stored. The application communicates between tiers using application programming interface (API) calls. The application data is stored in MongoDB, a NoSQL database [31], using JSON documents as it does not require a schema and is distributed, providing scalability, availability, and reliability of data.

B. MagicDraw Plugin

We use the MagicDraw Open Java API to build the customisation capabilities that provide model information at different modelling stages to their users. To our knowledge, while such graphical modelling tools, per default, provide an extensive set of validation rules [32], there is still a lack of an integrated DSML-centric recommendations view in MagicDraw as proposed in this paper. The reason lies in the nature of DSMLs being highly tailored for a specific domain and, thus, a particular modelling scenario cannot be predetermined at a general-purpose level. We create a MagicDraw plugin using Java that is eventually bundled as the DSML in the form of an archived file and provided to users. The MagicDraw Open Java API capabilities allows us to develop a UI for the plugin that adds to the already existing MagicDraw functionalities. MagicDraw also provides plugin development support in Eclipse, so we combine the UI development using WindowBuilder [33], an easy-to-use bi-directional Java UI designer, with the plugin development in the same integrated development environment (IDE). The plugin requests and retrieves relevant recommendation data from a database using API calls. Finally, we check the model data against the configurable business rules in the logic tier.

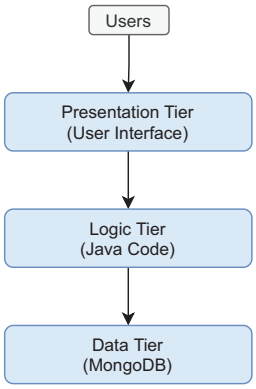

Fig. 1. Three-tier architecture for generating model-aware recommendations.

C. Database Configuration

As models grow complex in nature, so does the respective data and the relevant recommendations related to such models. We organise the data in the MongoDB database using document stores, in the form of JSON documents. This is important to our implementation, as we can serialise and deserialise these JSON documents into Java objects during application runtime and work effectively in the logic tier. Language engineers can therefore separate out the concerns of the business logic from the rest of the application. Henceforth, we refer to the pre-configured MongoDB as the database. In principle, the database can be configured to run locally on the same environment where the DSML is configured or remotely, i.e., the database is hosted on an external server. We configure MagicDraw's environment variables to provide the database server and port details for the plugin to connect.

Structure. The database consists of *collections* that store different kinds of information. For each collection, a *tags* field exists, containing a list of keywords assigned to models for which recommendations are provided. This ensures consistency between a DSML and the database, as only the tags

in the database need to be updated. For example, a tags such as “Collimator-V1” and “Collimator-V2” points to the same DSML element “Collimator” in different DSML versions. Each collection, listed in Table I, consists of *fields* that contain values specific to that field. For example, internally, we have DSML-specific information stored in various training documents in the form of Microsoft Word documents. We extract the necessary information from such documents, create the collection *textFromDocs* in the database, and populate the fields *docName* and *docDescription*. Another example is the storage of data related to training videos in the *hyperlinks* collection. These collections are managed by language engineers and the data can also be updated by domain experts without making any changes to the DSML itself. The data stored in these collections provide DSML-specific training materials, including training videos, general documentation, process models and their processes relevant to the models in consideration, frequently asked questions, and model-aware recommendations to assist users in effective modelling. The dataset consists of information collected over the years on various domains and specific topics within those domains, thus also reusing existing material and employing it in a model-aware fashion. Individual datasets can be created for each DSML project by understanding the domain-specific problems that modellers face frequently.

D. User-Centric Modelling Recommendations

To recommend modelling hints and suggestions to users, the logic tier must consider the current situation of a model and query the database against the configured business rules. Such *model awareness* is used to characterise the current context of the models or model elements and provide assumptions about the current situation of modelling constructs. To provide user-centric model-aware recommendations, the MagicDraw plugin UI is separated into three distinct parts. The UI observes, analyses, and provides recommendations related to DSML diagrams, matrices, tables, and model elements that belong to such diagrams. The first part displays the general overview of these diagrams in the form of static data that is stored in the database. The second part provides model-specific recommendations, either specific to the diagram or the elements that are configured on the diagram. The third part of the UI shows prescriptive process models relevant to the current models. Figure 2 shows an example of the UI with the *Overview*, *Recommendations*, and *Process Models* tabs shown to the user.

1) Overview: As part of the UI, the first tab that the user sees when they open a model diagram is the *Overview* tab. As the name suggests, this tab provides the necessary overview related to the language applied to the made diagram, i.e., the currently open diagram. Our implementation covers standard UML diagrams, such as class diagrams, sequence diagrams, and state machine diagrams, as well as customised diagrams specific to the DSML, such as a feature model diagram in a DSML that supports feature models. The logic tier processes the type of the diagram and sends a request to the concerned database collection to perform a text search on the *tags*

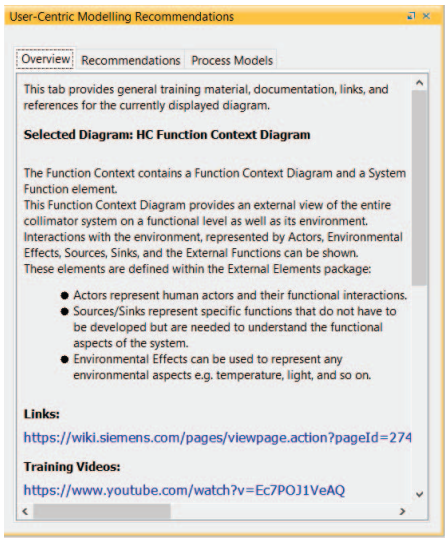

Fig. 2. An example of the UI with the *Overview*, *Recommendations*, and *Process Models* tabs. Here, a general overview, such as training information and links is shown as part of the *Overview* tab.

field of the collection. An example of such a search is to return documents from a database collection, where the type of a diagram is “Feature Model Diagram” and the *tags* in the collection contain “Feature Model” and “Feature Model Diagram” keywords. The JSON documents returned from the database based on the search are processed by the logic tier, translated into a hypertext markup language (HTML) format, and sent to the UI for display. The following sections of the *Overview* tab are populated:

General Overview. The first section displays static data pertaining to the currently open diagram. It returns a description of the diagram stored in the *textFromDocs* collection in the form of training material, handbooks, or tutorials. As the content is configured to be in HTML format, the use of styling is allowed to make the content presentable, such as with the use of icons, colours, item lists, size, and so on.

Links. The second section displays hyperlinks that link to internal company webpages or external websites that detail more information about the currently open diagram. It returns a list of hyperlinks that are stored in the *hyperlinks* database collection and is configured with Java MouseListeners [34] that, on click, redirect to the respective webpages.

Training Videos. The third section displays links to training videos for the currently open diagram. The logic tier pulls information from the *hyperlinks* database collection that it identifies as training videos. Complex DSMLs require training videos both for novice and advanced users. Often, these training videos are created towards the end or even after the deployment of the DSMLs to users. Frequent updates to such training videos ensures that users can see the most up-to-date content in such videos.

Frequently Asked Questions. The final section displays a list of frequently asked questions and their respective solutions for the currently open diagram. Language engineers and

TABLE I
MONGODB COLLECTIONS AND THEIR DESCRIPTION FOR STORING TRAINING MATERIALS, BUSINESS RULES, RECOMMENDATIONS, AND SO ON.

Collection	Fields	Description
elementsInDiagram	diagramName: String diagramElements: Array	Stores a list of DSML diagram elements that can be created as part of a DSML diagram.
faqs	question: String solution: String	Stores frequently asked questions and their possible solutions for a DSML model or model element.
hyperlinks	title: String link: String	Stores internal and external webpage links to training material, and links to videos for a DSML model or model element.
processModels	processName: String data: Base64 String	Stores prescriptive process models and processes as images encoded in base64 format.
recommendations	elementName: String recommendation: String	Stores recommendations for a DSML model or model element.
rules	rule: String recommendation: String	Stores pre-configured business rules and their recommendations for DSML models or model elements.
textFromDocs	docName: String docDescription: String	Stores documentation related to a DSML model or model element.

modelling experts are often asked about specific modelling techniques, such as how to build a specific model or the constructs of a DSML. Over the years, we have collected questions that are frequently asked to language engineers, domain experts, or to modellers in various domains. These questions, and their possible solutions, are stored in the *faqs* database collection. Solutions to these problems can also be modified directly in the database, therefore eliminating the need to update documentation within the DSML itself.

2) Recommendations: The second tab of the UI that the user sees when they open a model diagram is the *Recommendations* tab. This tab provides recommendations for the currently open diagram based on certain pre-configured business rules. These recommendations are dynamic in nature, meaning the recommendations shown to the user vary based on the current state of the models in the model diagram. These recommendations are stored in the database in the *recommendations* and *rules* collections and are presented to the users on the UI against the pre-configured rules. The recommendations are provided for the currently open diagram, the visible model elements on the diagram, or on a combination of model elements that are part of the diagram. The logic tier processes the type of the diagram and performs a text search on the *tags* field of the respective collections. The JSON documents returned from the database are then processed by the logic tier, translated into an HTML format, and finally, the recommendations are displayed to the users.

Rules. The recommendations that are provided to the users are based on certain pre-configured business rules. We build a simple rules engine that is part of the DSML plugin [35]. We create such a rules engine in the MagicDraw Java plugin and store them along with their recommendations in the database collection. Each rule is supported with a rationale that provides reasoning as to why the rule is required and what its benefits are. The rules are checked against the current state of the diagram, so the plugin is aware of the currently designed modelling constructs. The following rules are configured by a language engineer for a DSML model diagram:

- **Rule 1 (R1)**: If model elements that are displayed as part of the legend items in the diagram are not present on the diagram, provide a specific recommendation for the missing model element.
  Rationale: Standard UML diagrams or DSML-specific diagrams in MagicDraw can be configured with legend items that are model elements used to define different styles and visually group symbols in a diagram, therefore identifying model elements using visualisations. Such legend items are a good indicator of the visible and missing elements in a diagram, therefore helping in identifying missing parts of a model diagram.
- **Rule 2 (R2)**: If model elements configurable from the diagram elements toolbar are not present on the diagram, provide a recommendation on the missing elements.
  Rationale: MagicDraw's configurable model element toolbar can help users easily design model elements on a diagram. Users often forget to utilise the DSML functionalities that are provided as part of the toolbar but only model specific elements. This rule encourages users to double-check if their models utilise the DSML functionalities provided by the model element toolbar.
- **Rule 3 (R3)**: Provide specific recommendations for model elements visible on the currently open diagram.
  Rationale: Over the years we have collected feedback from users on what specific values a model element should consist of, or properties that need to be configured for each model element. We build a dataset of recommendations for such model elements and provide these recommendations to users. These recommendations change automatically as the users progress in their modelling.
- **Rule 4 (R4)**: Provide recommendations based on logical and relational operators on a single or a combination of model elements for the currently open diagram.
  Rationale: We provide recommendations for a single model element on a diagram based on relational operators. Conditions such as $<$, $>$, $<=$, $>=$, $==$, and $!=$ are checked for such model elements. An example of a simple relational condition could be if a model element of type energy is electrical, then provide a specific rec-

ommendation. We also configure logical operators for a combination of model elements on a diagram. Conditions such as AND, OR, and NOT are checked against a combination of model elements. An example could be if a data sink exists in the diagram but not a data source, providing a specific recommendation to the users.

Model-Aware Recommendations. The recommendations that are displayed on this tab of the UI are based on the rules that are checked against the models and the respective database collections described earlier. These active recommendations are based on the analysis of the current models and then populated in the UI in two sections. In the first section, recommendations that are specific to the DSML diagram are displayed. The logic tier analyses the currently open diagram, checks the rules configured for this diagram, and retrieves a list of recommendations from the database. By analysing the current state of the diagram, the plugin detects any changes made to the currently configured modelling constructs and adjusts the recommendations displayed to the users accordingly. This means the plugin is aware of the model constructs and provides dynamic recommendations based on the current modelling stage. A key benefit of providing recommendations to only the visible parts of the diagram is that the recommendations targets the user-specific view that a user creates to express only a certain aspect of a whole system. To view recommendations for other parts of the system, a user should navigate to the respective models. Certain constructs that may never be used will also be recommended, as users have often requested for such possibilities during their modelling. Following is a list of recommendations (non-exhaustive) that are specific to the currently open DSML diagram:

- Displaying missing parts in the model diagram.
- Suggesting the use of a certain model element (or a combination of model elements) in the model diagram.
- Specifying which constructs of the model diagram should be modified.
- Suggesting layout changes for certain diagram elements.

The second section of the *Recommendations* tab populates recommendations that are specific to the model elements visible on the DSML diagram. The plugin analyses the model elements that have been configured as part of the model diagram and provides dynamic recommendations based on the currently selected model element, its properties, and values. The following is a non-exhaustive list of recommendations that can be provided to users for the model elements configured in a DSML diagram:

- Configure specific properties of a model element.
- Checking the type of a model element and suggesting additional information related to the type.
- Setting specific values for a certain model element based on previous feedback from users or domain-experts.
- Redirecting users to the *Overview* tab of the UI to access training material specific to the model element.

3) Process Models: The third and final tab of the UI that the user sees when they open a DSML diagram is the *Process Models* tab. This tab displays prescriptive process models in the form of activity diagrams that are relevant to the currently open diagram. These process models describe the processes, tasks, and activities that need to be performed for achieving the desired modelling for this diagram and its constructs. In addition, these process models can also indicate the current state for modelling the constructs of the diagram. The *processModels* database collection stores various processes and process models, as images, in a base64 encoded format [36] that are relevant to a particular diagram with the relevant tags. The logic tier performs a search on the collection based on the *tags* field and retrieves a list of processes and process models that are stored by language engineers. Data in this collection is stored in the order in which they were inserted into the database and is therefore retrieved in the same order. The logic tier translates this information into a series of graphics using Java 2D™ API to decode the base64 data, paints the graphics into specific colours by checking against any pre-configured rules, and prints the sequence of created 2D graphics as process models on the tab. An advantage of translating the process model information in the logic tier is to analyse the diagram and adjust the graphics if there are changes in the diagram. Such changes can be accounted for displaying specific styles, such as colouring and icons. These styling options help users identify which processes, tasks, or activities of the process models are complete or incomplete, and elevates the overall user experience of modellers [29].

V. Industrial Case Study

Motivation. We demonstrate the applicability of our research in a real industrial context. Our colleagues at Siemens Healthineers have adopted MBSE techniques to foster their modelling situation by using a variety of concepts, methods, and tools that span from requirements engineering to product structure modelling involving bill of materials. To this extent, they have collaborated with Siemens AG and a number of academic and industrial partners on a public funded project, SpesML, that ensures a seamless exchange of models with the respective module suppliers and allows for the integration of entire systems. As part of this project, we have worked closely with all partners to deliver a DSML in MagicDraw that integrates the requirements, functional, logical, and technical views defined in the Software Platform Embedded Systems (SPES) [37] methodology as well as modelling specific constructs in the healthcare industry that support product line engineering. Ultimately, Siemens Healthineers want to achieve effective modular reuse of model elements with active modelling support and guidance. As an industrial use case, let us take a look at a Siemens Healthineers model example used in the context of describing decomposed functions in a system.

Precondition and Scope. We consider the following medical case. A patient is admitted to a hospital with a suspected forearm fracture. Once the patient is admitted, the on-duty treating clinician decides that they need to perform a 2D X-ray of the patient's forearm for further diagnosis. The clinician refers the patient to the hospital's radiology department for this

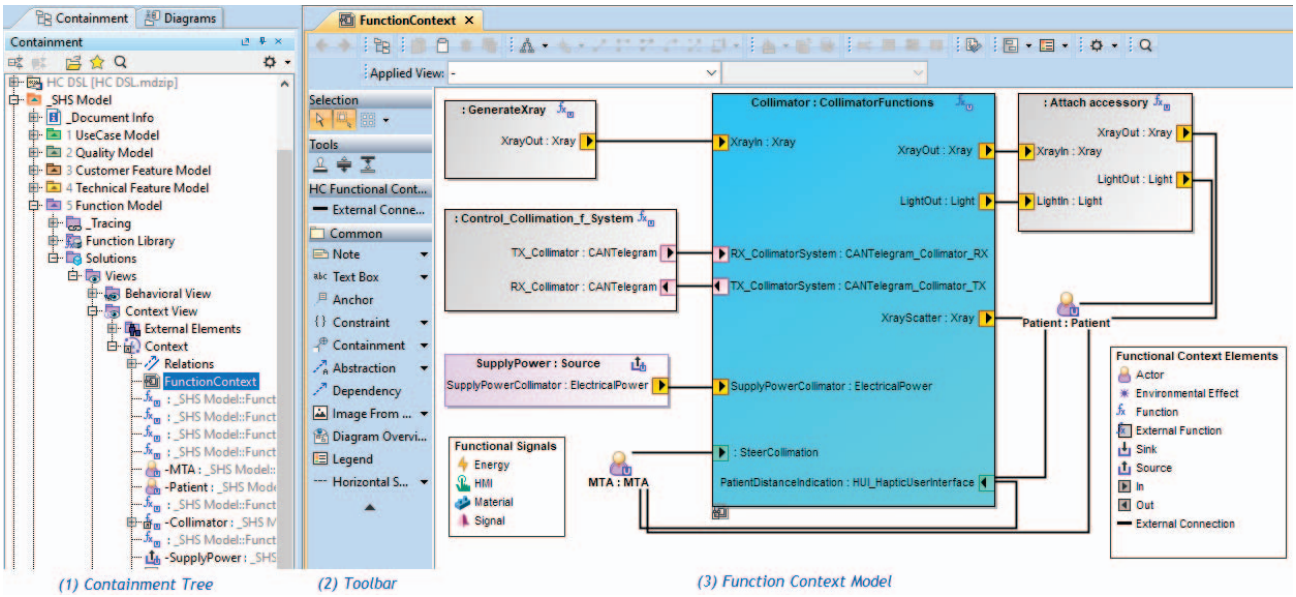

Fig. 3. Example of an X-ray collimator function context diagram designed using the Siemens Healthineers DSML containing (1) the containment tree, (2) the diagram toolbar with configurable model elements, and (3) the main function context model.

reason. Once the patient arrives in the hospital's radiology department, the patient is received by the medical technical assistant (MTA). The MTA has already prepared the X-ray system and has already entered the necessary patient data into the X-ray system. The MTA then positions the patient correctly on the X-ray system and leaves the room to start the process of taking an X-ray. The final result is that a 2D digital radiograph of the patient's forearm was generated and stored digitally.

Model. As part of the X-ray system, there exists many physical (technical view) elements and interfaces between these physical elements. A very specific part of the X-ray system is the collimator, which collimates precisely the radiation beam emerging from the X-ray system onto the patient's forearm. From a modelling perspective, this part can be decomposed into a set of functions that must be performed for the collimator to work properly. Figure 3 shows a function context model (functional view) for this collimator that has been modelled using the DSML in MagicDraw. On the left side of the figure, in the containment tree, DSML users can utilise the individual building blocks of the DSML to construct their models. On the right side of the figure is the function context model, which provides an external functional view of the entire collimator system on a functional level, as well as its interactions with actors, environmental effects, and other specific functions that are needed to fully understand the functional aspects of the system. As seen in the function context diagram, the user has been able to decompose various parts of the collimator. The central part of the function context model diagram describes the main function, the collimator itself, that receives (Rx) certain inputs and transmits (Tx) certain outputs from other smaller sub-systems such as a collimator controller, a power supply source, or accessories that are attached to the patient's forearm. There are also two actors, the MTA and the patient configured in the diagram performing interactions with the collimator. The input and output ports are configured with their respective functional signals that are listed in the legend items on the bottom left, with the colours of the ports signifying the type of functional signal it carries. These legend items represent various input and outputs of the functional elements such as "SteerCollimation" which is a human-machine interface (green), Xray a type of energy (yellow), and so on.

Methods and Recommendations. To build the collimator function context model diagram using the DSML provided, a user performs a series of steps. A user with an experience in modelling X-ray systems will be able to easily model the parts of the collimator without much hassle. However, in the case of a novice user, or for modelling advanced functionality, or for being guided appropriately in their modelling, they often encounter various challenges, including endless amount of time and resources spent going through passive information sources. To solve this, we understand the current modelling situation on the diagram, and display methods and recommendations on the right side of the DSML.

Figure 2, described earlier, shows a general overview of the currently open modelling diagram, the function context model diagram. By understanding that this diagram will be used by Siemens Healthineers to model a medical device such as an X-ray, or more specifically a collimator, we can use an appropriate dataset for providing an overview that caters to the individual modellers and their modelling goals. The first part of the overview tab identifies the type of the diagram, and provides static documentation related to a function context that is stored in the database. Subsequently, any additional hyperlinks are displayed as clickable links that redirect users to a more relevant page that further describes information about the function context and its interactions with system interfaces. Hyperlinks to training or general videos are also provided that

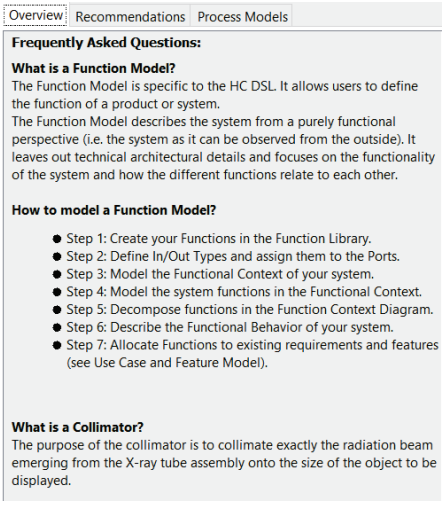

Fig. 4. An example of providing frequently asked questions in the *Overview* tab for the collimator function context diagram.

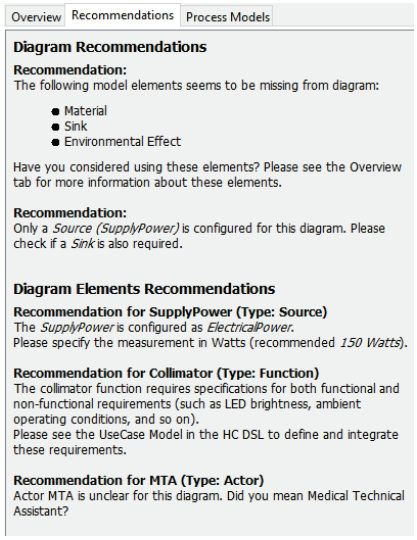

Fig. 5. The *Recommendations* tab showing recommendations for the diagram and its elements for the collimator example.

maybe useful in modelling a collimator, its functions, or its decomposed functions.

Figure 4 is also part of the *Overview* tab, and provides users with a list of questions and possible solutions that may be beneficial. Over the years, we have collected commonly asked questions to language engineers and domain-experts on either the use of DSMLs, on general domain-specific questions such as "*What is a Collimator?*", or in particular, how to model a specific scenario with the DSML. Here, the information is retrieved from a database that is populated with questions related to the function model, a collimator, or a set of methodical steps that users can follow. The displayed information is aware of the currently displayed models, for example, a specific question and answer related to the collimator.

Figure 5 shows the second tab, the *Recommendations* tab, of the UI. In this figure, dynamic recommendations are provided for the function context diagram, based on the current state of

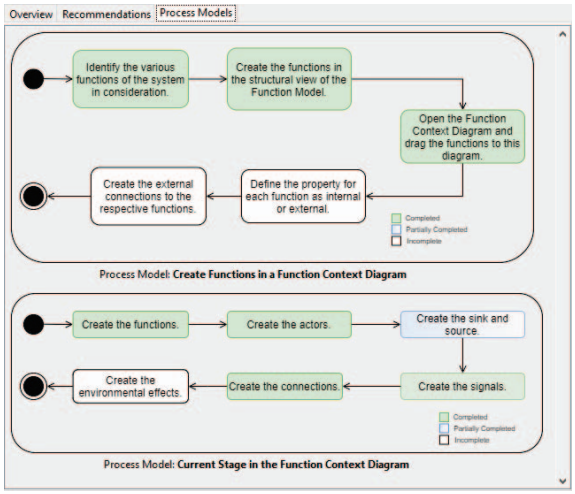

Fig. 6. The *Process Models* tab of the UI shows process models for the collimator function context diagram.

the models and their properties. This tab is divided into two parts, the first for providing recommendations specific to the collimator function context diagram itself, and the second to provide recommendations that are specific to model elements of the collimator function context. The figure shows the first recommendation as listing model elements, the functional signal material, the sink, and the environmental effect, that are currently missing from the function context diagram. This recommendation follows rule **R1**, where the current model items in the function context diagram are checked against the legend items, Functional Context Elements and the Functional Signals. This recommendation is provided because for the collimator example, it may be of importance to model a sink which collects the final 2D digital radiograph. Rule **R2** is also checked against, but since the external connection element, configurable from the diagram toolbar already exists in the actual model, there is no further recommendation on missing elements. The second recommendation provided in this figure analyses the function context diagram, and applies the logical rule **R4** on it. The rule condition stored in the database is "`Source NOT Sink`", meaning the rule checks if a source exists, but not a sink. In our experience, we have very frequently found that models generally are configured with both a source and a sink. The second part of the recommendations are checked against rule **R3** and are specific to the visible model elements on the function context diagram. The third recommendation indicates that the power supply source is electrical, but is without a measurement value. Based on historical data, a recommended value can be recommended for the electrical power supply. Once the measurement value is set in the model, the recommendation would be automatically removed. The fourth and the fifth recommendations follow rule **R4** and are based on relational operators. As an example, for the fourth recommendation, the rule condition stored in the database is "`Function == Collimator`".

Figure 6 shows the third and final tab, *Process Models* tab, of the UI. In this figure, we display two process models

generated from a list of processes retrieved from the database. The first process model details a sequence of steps needed to create functions in a function context diagram. While some individual processes can overlap with the information in other tabs, the processes listed here are checked against the current function context model and rules for this model, and subsequently assigned appropriate styling options. For example, the green processes indicate that the process is complete, the fading blue processes indicate partially completed processes, and the white processes indicate they are currently incomplete. The final process model is generated with legend items, that describes and identifies the processes as "Completed", "Partially Completed", and "Incomplete". The second process model describes the current state of the function context diagram. In particular, it identifies if all the model elements in the function context diagram have been created or not. This can be checked against the rules for the diagram, such as if two actors are created **(rule R4)**, the process "Create the actors." is marked as complete. In this example, a sink and a source is configurable for the function context, but only a source exists, so the respective process is partially complete, meaning modellers do not necessarily follow the order of the processes. As an environment effect is still missing from the collimator example, it is marked as incomplete.

Evaluation. A focus group consisting of experienced practitioners and researchers, with 8-15 years of experience in software and systems modelling, to understand current challenges in modelling, discussing possible solutions, and describing the functionalities was set up. They reasoned that an integrated DSML infrastructure should not only provide methodological guidance, but also be model-aware, context-specific, and evolve dynamically as the DSMLs grow in complexity. Data from past Siemens Healthineers projects was reused, that provided the basis of this case study. Further, the participants often fine-tuned the business rules and recommendations for practitioners. Previously such practitioners navigated through pages of static training data, handbooks, and overly generic technical documents, as no alternative solutions existed providing a more complete guidance infrastructure directly on the modelling tool along with the DSML. The focus group discussed the research and development of such an infrastructure from a conceptual and an implementation viewpoint. In the end, the participants concluded that developing such a guidance infrastructure should be active and synchronised with the ongoing modelling work, as current solutions provide a rather plain technical view without any continuous support.

VI. Discussion

The methodology presented in this paper enables language engineers to directly integrate aspects of providing effective model-specific methods, techniques, and recommendations to users. Understanding the current state of their models, the methodology described can suggest specific processes, activities, or tasks that are needed to be performed to further assist modellers. Even when the meaning of individual model elements is explicit, establishing or communicating a particular modelling method is often challenging. By providing recommendations based on a set of predefined configurable rules, language engineers can introduce a more holistic way of integrating a method within the DSML itself.

Generally, context conditions, defined as part of the DSML, can be used to check the well-formedness of a model but essentially do not tell the modeller anything about the semantics (in terms of the meaning) and behaviour of the models. While such validation rules in the DSMLs generate warnings or errors, they provide a rather strict constraint to the syntax. On the other hand, methods and recommendations focus on supporting users in their modelling activity, giving them the confidence to model effectively in their domains without endlessly searching through pages of tedious documentation.

To demonstrate the applicability of the general methodology described in the paper, we developed a three-tiered architecture consisting of a presentation tier (UI), a logic tier (business logic and rules), and a data tier (NoSQL database). We used MagicDraw, a commercial modelling tool with vast customisation capabilities, such as for providing additional UI capabilities to improve a model's aesthetics. We utilise these customisation capabilities of MagicDraw to assist users with a UI, built using MagicDraw's Open Java API, by providing information about the general overview of the currently open diagram. This comprises hyperlinks that redirect them to sources of information, including webpages and videos, commonly asked questions for the models in consideration, and recommendations for the DSML diagram and its elements. Additionally, we provide a set of process models that detail activities already performed or currently missing from the diagram. For each domain, our dataset consists of configurable rules that provide active domain-specific information based on the current context of the models and the specific modelling situation the users are currently in. Due to their external disposal in a database, we constantly update the rules and their recommendations to include more complex modelling scenarios, meaning, these rules can be reused across other domains as well. While updating a DSML can be a time-consuming process, editing data externally is relatively straightforward and intuitive even for non-experts, but should be left to the domain experts. This way, conflicts between a modeller's and a domain expert's description of a model can be avoided.

As part of our ongoing work, we aim to also provide actions for the recommendations that the users can directly perform, e.g., to set a specific measurement value to a model, such as assigning a recommended wattage to a power supply unit, or complex rules that checks links between models in different diagrams and allows navigation between recommendations. Further, we are also developing techniques to make the process models navigable for users to look at the individual processes in detail. Additionally, we are also looking at ways to provide modelling predictions with machine learning algorithms.

The validity of this study is the extent to which the methodology described in this paper, both in concept and in implementation, is free from systematic errors or bias. While the methodology is described generally, our study has been

researched using a commercial modelling tool, MagicDraw, and thus, underlies its characteristic technical details and introduces a vendor-locked scenario. MagicDraw, Rational Rhapsody, or Enterprise Architect do not represent the only tools to build DSMLs and ongoing work to identify and evaluate other approaches is underway. However, as the UI and logic tier is developed using Open Java API, and the data tier is configured outside of MagicDraw, the actual implementation can be developed independent of a modelling tool, thus mitigating this threat. Yet, for implementing this methodology in a specific modelling tool, it should expose an API that can retrieve information about the models, such as various properties of a diagram. Another threat arises from the nature of guidelines potentially being too restrictive, such that modellers might be tempted to merely ”fill out” a model without thinking of novel solutions. However, as our proposed methodology only provides lightweight suggestions without imposing restrictions, we deem that threat low. In fact, as the recommendations not only guide users based on historical data but also provide a general overview, they foster novel thinking.

We validate our study using a real-world industrial context example of decomposing various functions of a collimator in an X-ray system modelled by experts at Siemens Healthineers. Implementing this methodology resulted in increased interaction between language engineers and domain experts to capture various aspects of the domain in detail. We observed that for the X-ray collimator function context example, around 12 rules and their recommendations were configured. As other model diagrams, e.g., feature models of the DSML, contain a similar number of rules and since the recommendations are calculated only for the displayed diagrams, we have not observed scalability issues. This means that users are not burdened with a plethora of recommendations and configuring a simple rule engine by language engineers suffices. Defining process models for a model diagram is rather straightforward for language engineers as they can identify during DSML development the different parts that can be modelled. Efforts in integrating methods within the DSML and the modelling tool have resulted in positive feedback from practitioners that use our DSMLs to improve the overall modelling experience. One key takeaway in providing an integrated methodology within the DSML is that both novice and advanced users can better understand the DSML constructs and can model with more confidence. To this end, we consider this research a good reference point in future discussions towards improving the development of industrial DSMLs that is more aware of the modelling situation of users.

VII. Related Work

Domain-specific knowledge is required for a deeper understanding in providing users with comprehensive modelling recommendations related to their DSMLs and its constructs. Environments that provide methods supporting computer-aided method engineering [38], [39], [40] has been studied for method engineers, but they do not largely consider domain-specific aspects or the modelling tool itself [10]. While MBSE methods such as MagicGrid [41], allows for a comprehensive modelling of technical systems in SysML, it is overly generic and challenging to implement as a rather independent method framework, thereby forcing profile-based tools to extract the methodological part away from a DSML definition. Recently, commercial content providers have built recommender systems that gather and understands the users' preferences and make recommendations [25], [42]. This is also extended to model-driven engineering by building user profiles and ranking algorithms [43]. In contrast, we do not store preferences of users, rather make modelling recommendations based on their current state of their models along with feedback from domain-experts and modellers in the past. While certain studies look specifically at algorithms such as a nearest-neighbour approach [44] or mutually reinforcing methods [45] to classify recommendations, we employ a simple rule engine in Java [35], that checks against the current modelling constructs and the relevant database entries, and is integrated directly within the modelling tool. In their study [46], the authors suggest a service oriented architecture for linking MBSE tools to the services they offer such as the creation of a composite service comprised of smaller services. Studies pertaining to context-aware methods have also been discussed in the literature [47], [48]. While they discuss techniques to represent domain-specific modelling through business rules, the context of use, they are primarily used for modelling domain-specific business, and do not provide any methods or recommendations to users based on this information. Process-aware, process models, and their verification in DSMLs have been researched in the literature [49], [50]. In our implementation, we suggest process models based on the current models that provide a more holistic view of the current state of the modelling scenario. While large domain-specific textual databases require techniques to improve document retrieval [51], our database configurations are specific to each DSML project, therefore a simple document retrieval is sufficient.

VIII. Conclusions

As systems in various domains become heterogeneous and more complex, so does the challenge of developing efficient methods, guidance, and support for users of industrial DSMLs. To this end, various graphical modelling tools provide capabilities that support displaying prescriptive information, such as documentation, activity diagrams, etc. However, there still exists the challenge of integrating better support and guidance for users that also considers the current modelling situation. To address this challenge, we provide a methodology for augmenting modelling languages with proactive recommendations based on a user's modelling situation directly on the modelling tool. Although our results constitute a promising step towards aligning methods within the DSML, there is a lot of further work to be done on this topic, such as integrating machine learning algorithms that predict models. Embedding proactive recommendations in DSMLs alleviates modellers' effectiveness and facilitates an immediate understanding and solving of complex modelling scenarios.

REFERENCES

[1] R. France and B. Rumpe, "Model-driven Development of Complex Software: A Research Roadmap," *Future of Software Engineering (FOSE '07)*, pp. 37–54, May 2007.

[2] H. A. Proper and M. Bjekovic, "Fundamental challenges in systems modelling," *EMISA Forum*, vol. 39, no. 1, pp. 13–28, 2019.

[3] T. Clark, M. v. d. Brand, B. Combemale, and B. Rumpe, "Conceptual Model of the Globalization for Domain-Specific Languages," in *Globalizing Domain-Specific Languages*, ser. LNCS 9400. Springer, 2015, pp. 7–20.

[4] T. Degueule, B. Combemale, A. Blouin, O. Barais, and J.-M. Jézéquel, "Melange: A Meta-language for Modular and Reusable Development of DSLs," in *8th International Conference on Software Language Engineering (SLE)*, Pittsburgh, United States, 2015.

[5] L. Bettini, *Implementing domain-specific languages with Xtext and Xtend*. Packt Publishing Ltd, 2016.

[6] J.-P. Tolvanen, "MetaEdit+ integrated modeling and metamodeling environment for domain-specific languages," in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 2006, pp. 690–691.

[7] R. Gupta, S. Kranz, N. Regnat, B. Rumpe, and A. Wortmann, "Towards a Systematic Engineering of Industrial Domain-Specific Languages," in *2021 IEEE/ACM 8th International Workshop on Software Engineering Research and Industrial Practice (SE&IP)*. IEEE, May 2021, pp. 49–56.

[8] N. Regnat, "Why SysML does often fail - and possible solutions," in *Modellierung 2018, 21.-23. Februar 2018, Braunschweig, Germany*, 2018, pp. 17–20.

[9] M. V. Cengarle, H. Grönniger, and B. Rumpe, "Variability within Modeling Language Definitions," in *Conference on Model Driven Engineering Languages and Systems (MODELS'09)*, ser. LNCS 5795. Springer, 2009, pp. 670–684.

[10] B. Honke, "Situational Method Engineering for the Enactment of Method-Centric Domain-Specific Languages," Doctoral Thesis, Universität Augsburg, 2013.

[11] H. Agt-Rickauer, R.-D. Kutsche, and H. Sack, "Automated recommendation of related model elements for domain models," in *Model-Driven Engineering and Software Development: 6th International Conference, MODELSWARD 2018, Funchal, Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 6*. Springer, 2019, pp. 134–158.

[12] U. Frank, "Outline of a method for designing domain-specific modelling languages," ICB-research report, Tech. Rep., 2010.

[13] (2022) Enterprise Architect. [Online]. Available: https://sparxsystems.com/

[14] (2022) IBM Rhapsody. [Online]. Available: https://www.ibm.com/products/systems-design-rhapsody/

[15] (2022) MagicDraw Enterprise. [Online]. Available: https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/

[16] J.-M. Favre, D. Gasevic, R. Lämmel, and E. Pek, "Empirical language analysis in software linguistics," in *Software Language Engineering: Third International Conference, SLE 2010, Eindhoven, The Netherlands, October 12-13, 2010, Revised Selected Papers 3*. Springer, 2011, pp. 316–326.

[17] B. Combemale, R. France, J.-M. Jézéquel, B. Rumpe, J. Steel, and D. Vojtisek, *Engineering Modeling Languages: Turning Domain Knowledge into Tools*. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series, November 2016.

[18] K. Hölldobler, O. Kautz, and B. Rumpe, *MontiCore Language Workbench and Library Handbook: Edition 2021*, ser. Aachener Informatik-Berichte, Software Engineering, Band 48. Shaker Verlag, May 2021.

[19] F. Campagne, *The MPS language workbench: volume I*. Fabien Campagne, 2014, vol. 1.

[20] D. Harel and B. Rumpe, "Meaningful Modeling: What's the Semantics of "Semantics"?" *IEEE Computer*, vol. 37, no. 10, pp. 64–72, October 2004.

[21] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools," *Information and software technology*, vol. 38, no. 4, pp. 275–280, 1996.

[22] R. A. de Oliveira, M. Cortes-Cornax, A. Front, and A. Demeure, "A low-code approach to support method engineering," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2022, pp. 793–797.

[23] Q. Ma, M. Kaczmarek-Heß, and S. de Kinderen, "Validation and verification in domain-specific modeling method engineering: an integrated life-cycle view," *Software and Systems Modeling*, pp. 1–21, 2022.

[24] H. Scholta, "Similarity of Activities in Process Models: towards a Metric for Domain-Specific Business Process Modeling Languages," in *European Conference on Information Systems*, 2016.

[25] M. Nakatsuji, Y. Fujiwara, A. Tanaka, T. Uchiyama, and T. Ishida, "Recommendations over domain specific user graphs," in *ECAI 2010*. IOS Press, 2010, pp. 607–612.

[26] P. Roques, "MBSE with the ARCADIA Method and the Capella Tool," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[27] R. Gupta, N. Jansen, N. Regnat, and B. Rumpe, "Implementation of the SpesML Workbench in MagicDraw," in *Modellierung 2022 Satellite Events*. Gesellschaft für Informatik, June 2022, pp. 61–76.

[28] B. Rumpe, *Modeling with UML: Language, Concepts, Methods*. Springer International, July 2016.

[29] R. Gupta, N. Jansen, N. Regnat, and B. Rumpe, "Design Guidelines for Improving User Experience in Industrial Domain-Specific Modelling Languages," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. Association for Computing Machinery, October 2022.

[30] A. Aarsten, D. Brugali, and G. Menga, "Patterns for three-tier client/server applications," *Proceedings of Pattern Languages of Programs (PLoP'96)*, vol. 4, no. 6, 1996.

[31] P. Membrey, E. Plugge, T. Hawkins, and D. Hawkins, *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. Springer, 2010.

[32] S. Erdweg, T. v. d. Storm, M. Völter, M. Boersma, R. Bosman, W. R. Cook, A. Gerritsen, A. Hulshout, S. Kelly, A. Loh *et al.*, "The state of the art in language workbenches," in *International Conference on Software Language Engineering*. Springer, 2013, pp. 197–217.

[33] D. Rubel, J. Wren, and E. Clayberg, *The Eclipse Graphical Editing Framework (GEF)*. Addison-Wesley Professional, 2011.

[34] (2023) Java™ Platform, Standard Edition 8 API Specification. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/index.html?javax/swing/package-summary.html

[35] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[36] S. Josefsson, "The base16, base32, and base64 data encodings," Tech. Rep., 2006.

[37] W. Böhm, M. Broy, C. Klein, K. Pohl, B. Rumpe, and S. Schröck, Eds., *Model-Based Engineering of Collaborative Embedded Systems*. Springer, January 2021.

[38] S. Brinkkemper, K. Lyytinen, and R. Welke, *Method engineering: Principles of method construction and tool support*. Springer Science & Business Media, 1996.

[39] C. Green, D. Luckham, R. Balzer, T. Cheatham, and C. Rich, "Kestrel Institute: REPORT ON A KNOWLEDGE-BASED SOFTWARE ASSISTANT," in *Readings in Artificial Intelligence and Software Engineering*. Elsevier, 1986, pp. 377–428.

[40] A. Niknafs and R. Ramsin, "Computer-aided method engineering: an analysis of existing environments," in *Advanced Information Systems Engineering: 20th International Conference, CAiSE 2008 Montpellier, France, June 16-20, 2008 Proceedings 20*. Springer Berlin Heidelberg, 2008, pp. 525–540.

[41] A. Aleksandraviciene and A. Morkevicius, "MagicGrid® Book of Knowledge-A Practical Guide to Systems Modeling using MagicGrid from No Magic," *Inc, Allen Texas, USA*, 2018.

[42] S. Nadschläger, H. Kosorus, A. Boegl, and J. Kueng, "Content-based recommendations within a QA system using the hierarchical structure of a domain-specific taxonomy," in *2012 23rd International Workshop on Database and Expert Systems Applications*. IEEE, 2012, pp. 88–92.

[43] L. Almonte, E. Guerra, I. Cantador, and J. De Lara, "Recommender systems in model-driven engineering: A systematic mapping review," *Software and Systems Modeling*, pp. 1–32, 2021.

[44] S. Sippel, "Domain-specific recommendation based on deep understanding of text," *Informatik 2016*, 2016.

[45] J. Wang, J. Xiang, and K. Uchino, "Domain-specific recommendation by matching real authors to social media users," in *Advances in Web-Based Learning–ICWL 2016: 15th International Conference, Rome, Italy, October 26–29, 2016, Proceedings 15*. Springer, 2016, pp. 246–252.

[46] A. Aldazabal, T. Baily, F. Nanclares, A. Sadovykh, C. Hein, and T. Ritter, "Automated model driven development processes," in *Proceedings of*

the ECMDA workshop on Model Driven Tool and Process Integration. Citeseer, 2008, pp. 361–375.

[47] J. R. Hoyos, J. García-Molina, and J. A. Botía, "A domain-specific language for context modeling in context-aware systems," *Journal of Systems and Software*, vol. 86, no. 11, pp. 2890–2905, 2013.

[48] M. Lethrech, I. Elmagrouni, M. Nassar, A. Kriouile, and A. Kenzi, "Domain Specific Modeling approach for context-aware service oriented systems," in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE, 2014, pp. 575–581.

[49] J. C. Kirchhof, A. Kleiss, B. Rumpe, D. Schmalzing, P. Schneider, and A. Wortmann, "Model-driven Self-adaptive Deployment of Internet of Things Applications with Automated Modification Proposals," *ACM Transactions on Internet of Things*, November 2022.

[50] B. Combemale, X. Crégut, P.-L. Garoche, X. Thirioux, and F. Vernadat, "A property-driven approach to formal verification of process models," in *Enterprise Information Systems: 9th International Conference, ICEIS 2007, Funchal, Madeira, June 12-16, 2007, Revised Selected Papers 9*. Springer, 2008, pp. 286–300.

[51] R. Stanković, C. Krstev, I. Obradović, and O. Kitanović, "Improving document retrieval in large domain specific textual databases using lexical resources," in *Transactions on Computational Collective Intelligence XXVI*. Springer, 2017, pp. 162–185.