Development of Light-Weight Web-based Metamodeling Tool

Guntis Mosāns[a] and Jānis Kampars[a]

[a] *Riga Technical University, Kalku 1, Riga, LV-1658, Latvia*

Abstract

To deal with the increased complexity of businesses and their information systems, organizations can use models to untangle these complexities, accelerate the development of information systems, and provide their documentation. Metamodeling allows the formalization of the problem domain and provides a means for creating quality models. Metamodeling tools should possess flexibility and adaptability as well as the possibility to integrate modeling components in existing systems. This paper aims to design a modular web-based metamodeling platform and demonstrate it with a practically implemented prototype. The metamodeling platform prototype is validated by implementing the metamodel underlying the Capability Driven Development (CDD) methodology and creating a web-based capability modeling tool.

Keywords

Metamodeling, web-based tools, CDD, Capability

1. Introduction

Models are helpful to deal with the complexities of designing large information systems. A number of modeling paradigms and methods are well-established. Methods such as UML, BPMN, 4EM, and others are frequently used in the analysis and development of complex systems. These methods are elaborated on the basis of their metamodels. Metamodeling facility (MOF) is frequently used to specify the meta-models [1]. Having a method based on a well-defined metamodel enables the development of consistent and interoperable models as well as to provide tools for implementing systems out of the models. There is a variety of tools supporting development with UML, BPMN, and similar methods [2]. However, specialized tools are needed if development needs call for the employment of domain-specific languages (DSL) [3]. Metamodeling platforms are tools for specifying DSL and the generation of modeling tools supporting these DSLS. MetaEdit++, Adoxx are some of the best-known metamodeling platforms [4]. These platforms have evolved over the previous 25 years, and significant changes in development technologies have occurred over this period. Therefore, the existing platforms are not well suited for the age of web technologies and service orientation. For example, for hybrid method engineering, metamodeling platforms should be realized on a component-based, distributable, scalable architecture and it must be possible to define meta-metamodels [5].

The objective of this paper is to prototype a light-weight web-based metamodeling tool with improved collaboration and interoperability features. The metamodeling tool allows the creation of DSL and generation of corresponding modeling tools. These modeling tools generated are intended as plug-ins for other web applications requiring modeling capabilities. Both the metamodeling tool and the modeling tools generated use standard web technologies. Functionality of the metamodeling tool is demonstrated by the generation of the Capability Driven Development (CDD) methodology modeling method.

The rest of the paper is organized as follows. Section 2 reviews the features of well-known metamodeling tools. Section 3 introduces the design of the proposed meta-modeling tool. An application example of the tool is provided in Section 4. Section 5 concludes.

PoEM'20 Forum: 13th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling, Forum, November 25–27, 2020, Riga, Latvia

EMAIL: guntis.mosans@rtu.lv (G. Mosāns); janis.kampars@rtu.lv (J. Kampars)
ORCID: 0000-0001-9373-4000 (G. Mosāns); 0000-0003-0045-5593 (J. Kampars)

2. Overview of Metamodeling Tools

Most tools are designed for a specific problem or platform. Many tools are designed to model domain-specific languages. There is a number of metamodeling tools and every tool has its own advantages and disadvantages [6]. This article will look at the most popular metamodeling tools, MetaEdit++ [7], Adoxx [8], EMF [9], and GME [10]. They are chosen for a detailed analysis to determine typical characteristics of existing metamodeling tools. The main features considered are easy to use, isare open source, easy to integrate with other systems, web-based, and supports XML and JSON formats. The summary of the analysis is given in Table 1.

EMF and GME require Java language programming knowledge to defining some features of metamodels and model transformations. All tools reviewed are stand-alone tools requiring local installation. Collaboration is achieved by using client-server technology and modeling repositories. These tools support model import/export facilities to support interexchange as well as to publish the metamodels created. However, they are not intended as plug-in tools in other modeling and development environments.

Table 1
Analysis of features of metamodeling tools

Feature	MetaEdit++	Adoxx	EMF	GME
Documentation	+++	+++	+++	++
Platform	Linux, Windows, MacOS	Linux, Windows, MacOS	Linux, MacOS, Solaris, Windows	Windows
License	Commercial	Free	Open Source	Open Source
Integration	Standalone	Standalone	Eclipse	Standalone
Model transformation	Yes	Using additional script	Using plugins	Using additional tool
API	Yes	Yes	Yes	Yes
Abstract syntax	Yes	Yes	Yes	Yes
Concrete Syntax	Vector graphic	Graphic language	Using GMF	Windows GDI
Defining relationships	Yes	Yes	Yes	Yes
Defining model constraints	Yes	Only relationships	Yes	Yes
Multi-user support	Yes	Yes	No	No
Metamodel import/export	XML	XML/ADL	XML	XML

The tools and studies reviewed confirm the need to create an easy-to-use tool that is open source, easily integrates with other systems, is web-based, and supports XML and JSON formats. Currently, all tools only support XML format.

3. Platform Design

The key requirements towards the platform are defined and the platform architecture and detailed design are elaborated according to these requirements.

3.1. Requirements

The main requirements are determined by analyzing the existing metamodeling tools. As determined in the previous chapter, the system must be web-based, easy to use and ensure the creation of metamodels and models. The modeling system must be easily integrated with other systems, as required by Model Driven Development (MDD) [11], and it must be possible to work with modeling capabilities [12]. The main system requirements are:

- Web application supporting access to commonly used web browsers;
- Light bandwidth load;

- Quick environment setup and fast response;
- System configuration must be flexible and scalable;
- The platform must be secured with an SSL certificate;
- The frontend of the metamodeling tool should allow us to define custom metamodels and provide metamodel import/export;
- The frontend of the metamodeling tool also must provide the creation of graphic designs, and the export and import functionality of the metamodel element design;
- The frontend also needs to provide metamodel templates;
- The backend of the metamodeling tool should provide REST API services for frontend and modeling tool generation;
- The backend also should ensure data storage in the database and provide metamodel base;
- The frontend of the modeling tool should provide the development of models from metamodels;
- The frontend also should provide models import and export and ensure automatic layout;
- The backend of the modeling tool should provide REST API services for the modeling tool frontend;
- The backend also needs to provide data storage and ensure the simultaneous cooperation of several authors in the development of the model;
- Models are developed using a graphical user interface and models are represented using XML and JSON;

The analysis of these requirements yields the use case diagram (Figure 1). The purpose of the use case diagram is to summarize what users will be able to do with the developed system and outline the initial requirements for the system. The creator of the metamodel and modeler are two main actors. The platform provides the functionality to create a metamodel, to generate a modeling tool on the basis of the metamodel, and to develop new models.

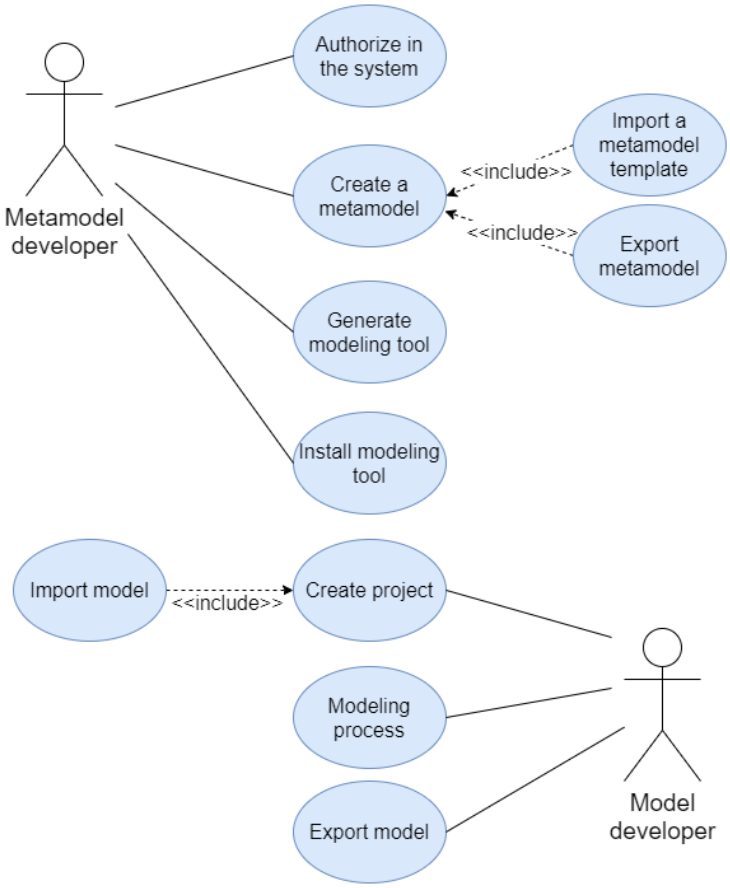

Figure 1: Use case diagram of the metamodeling platform

3.2. Architecture

The architecture of the metamodeling tool is constructed according to the requirements (Figure 2). It consists of two main parts, namely, the metamodeling tool and the generated modeling tool. The metamodeling tool part is responsible for the creating of the metamodel and generation of the modeling tool. It consists of the front-end and back-end components.

The metamodeling tool's frontend consists of a React framework. React was used because it provides performance through a virtual DOM (Document Object Model). DOM is a cross-platform and programming API that works with HTML, XML, and XHTML technologies. The DOM is fully operational using virtual memory [13]. The React framework provides import and export of metamodels in XML, JSON formats, metamodel editor using the mxGraph library, and data exchange with the backend using API. mxGraph was used because it provides a lot of graphical drawing options compared to other open-source solutions [14]. The metamodeling backend is built using the Nodejs framework. The main task of the framework is to provide API services for frontend and modeling tool generation. The model generation component is used to create model instances on the basis of the metamodel. Different modeling tools can be generated using the metamodeling tool.

The modeling frontend consists of React framework to ensure the modeling performance speed. The frontend provides a graphical modeling editor, model import and export, and API calls to the backend. The backend is responsible for this API call handling and data storage. It is built in the NodeJs framework. NodeJs was used because it is an asynchronous and event-driven framework which provides performance [15].

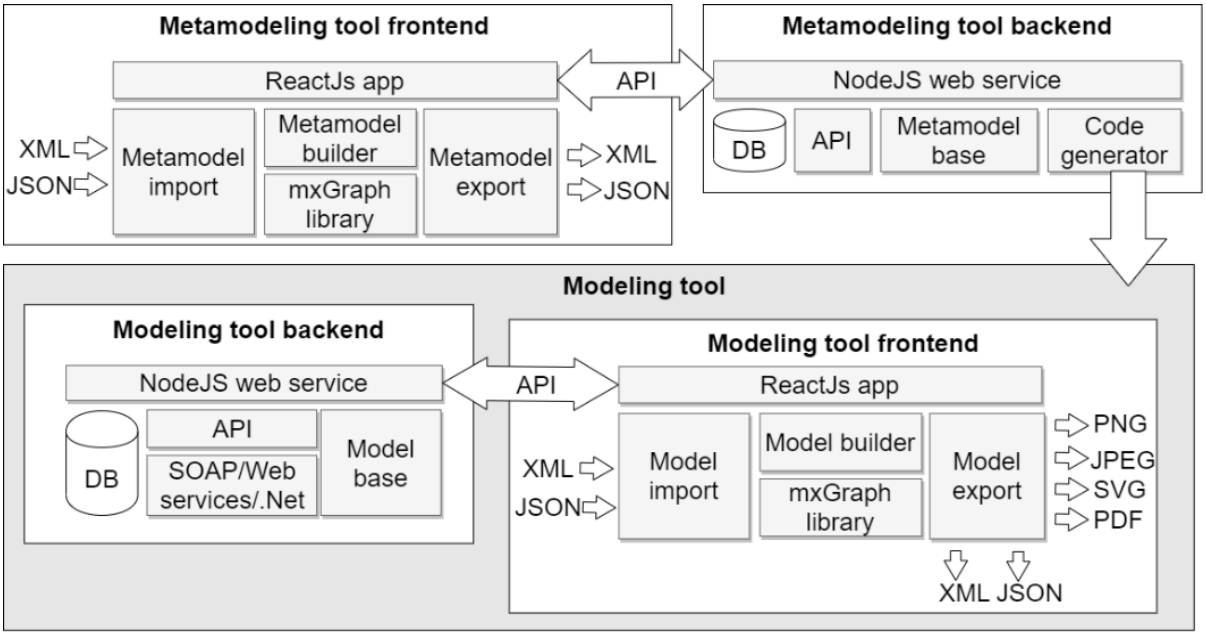

Figure 2: Metamodeling platform architecture

3.3. Design

The architecture is further elaborated and represented using a sequence diagram. Figure 3 shows the main activities of the metamodeling and modeling tools as a UML sequence diagram. Six metamodeling main activities are shown on the left. The first step is user authorization in the system. The next step is creating a project and in the third step, the user can import or create a metamodel. Metamodel can be imported in JSON format. The next step is the creation of metamodel elements. The metamodeler can create objects, relationships, roles, and ports. In the last step, the metamodeler can optionally export the metamodel (JSON), if not the metamodeler generates a modeling tool.

Modeling consists of five steps. The first stage is the creation of the project, then the modeler can import the model as needed. The model can be imported in JSON and XML formats. Then the modeler

if wants can share the model with others. In the fourth stage, the modeler performs the modeling process and in the last step, the user can export the created metamodel. Export is possible in JSON, XML, PNG, JPEG, SVG and PDF formats.

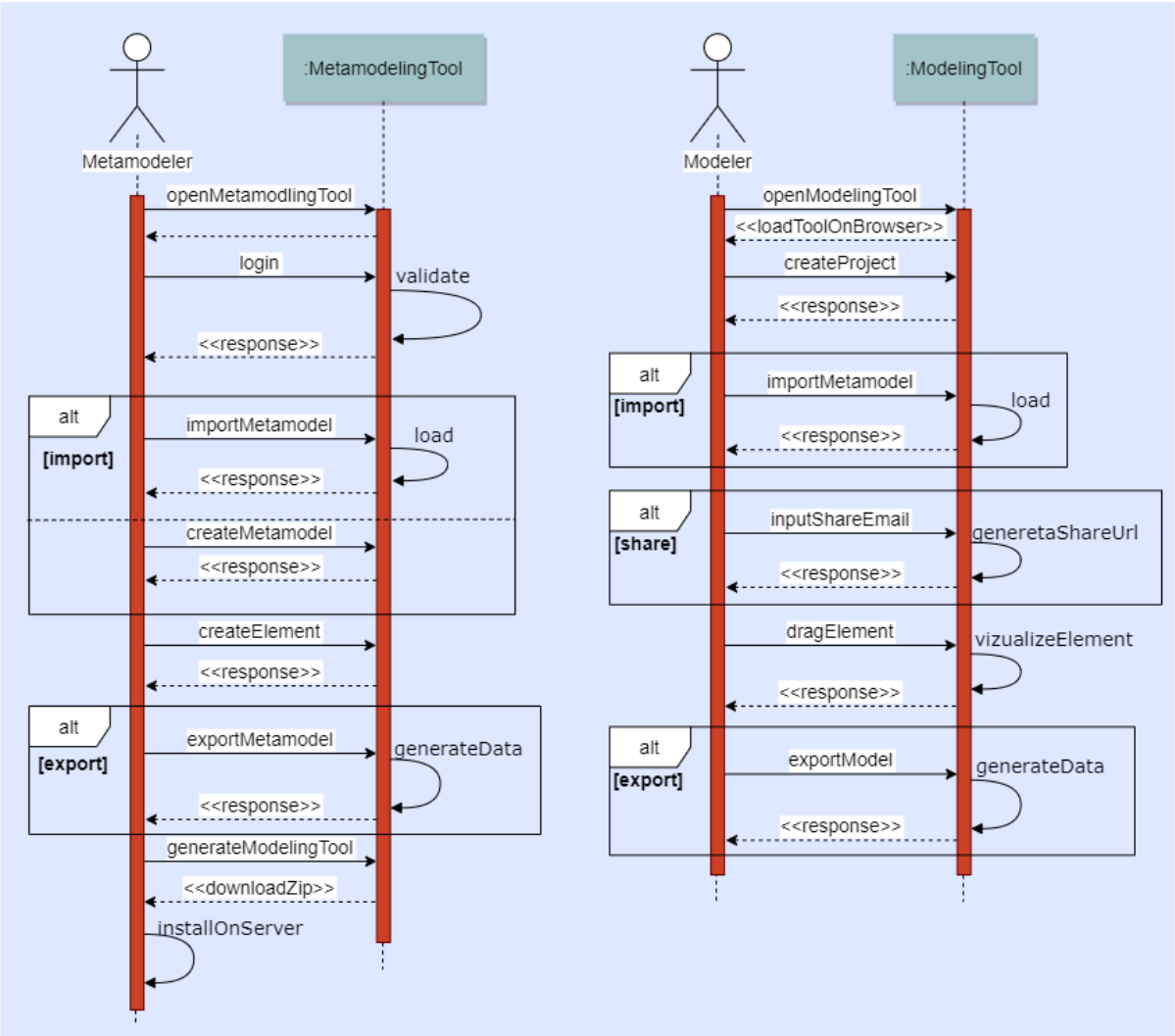

Figure 3: The UML sequence of the main activities of metamodeling and modeling

4. Demonstration

In order to illustrate the implemented metamodeling tool, a demonstration case is presented. The case considers the development of the web-based modeling tool for the metamodel of the Capability Driven Development (CDD) methodology [16].

CDD is a capability-based method for developing context-aware and adaptive systems. CDD defines capability, design, delivery, and knowledge accumulation processes. These three activities are integrated in a way that allows a company to perceive contextual information related to capability objectives and to adapt capability delivery to changes in context and performance [17].

To create a CDD modeling web application, one must firstly implement a metamodel using the metamodeling tool. Figure 5 shows the front-end of the metamodeling tool and its metamodel creation workspace in particular. A new metamodel is created as a part of the metamodeling project. The metamodel consists of objects, relationships, roles, and ports, which are also created in the workspace. The GOPPRR approach [18] to defining metamodels is used.

The demonstration platform supports object creation. It is possible to define object title, type, size, description, style, and properties. The style is defined using the mxGraph notation. Moreover, it is possible to upload an image for objects. When an image is uploaded, it is converted to the base64 format

and formatted with the mxGraph library notation. Properties are added to the object using another form dialog. At this moment it is possible to add text fields and drawing properties.

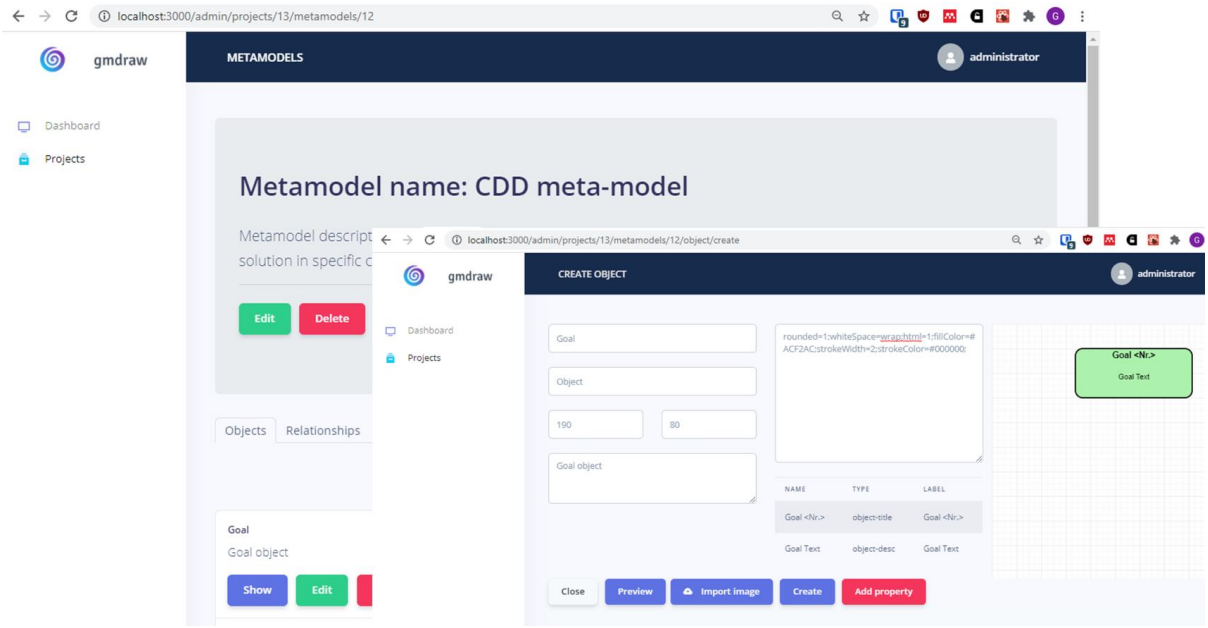

Figure 4: The metamodel specification environment

To implement the CDD meta-model, the following objects are created [19]:

- Goal – which is a desired state of affairs that needs to be attained. Goals can be refined into sub-goals forming a goal model;
- Capability – which is the ability and capacity that enables an enterprise to achieve a business Goal in a certain context;
- KPI – Key Performance Indicators are measurable properties that can be seen as targets for the achievement of goals;
- Process – is a series of actions that are performed to achieve a particular result. The Process supports Goals and has inputs and produces output in terms of information and/or materials.
- A Context Element – represents any information that can be used to characterize the situation of an entity;
- Adjustment is used when KPI value is not satisfying in terms of the design KPI or when a context element value exceeds its context element range;
- Patterns are reusable solutions for reaching business goals under specific situational contexts. The context defined for the Capability (Context Set) should match the context in which the pattern is applicable;
- Measurable Property - Measurable Property is any information about the organization's environment that can be measured;
- Service – is an implementation of capability such as business functionality that it is accessible through a well-defined interface;
- Context Element Range – which specifies the boundaries of permitted values for a specific Context Element and for a specific Context Set;
- Context Set – that describes the set of Context Elements that are relevant for the design and delivery of a specific capability.

The metamodeling frontend is built in React using Argon library design. The frontend forwards all REST API calls to the backend. For API calls, a Axios library is used. With the frontend it is also possible to create projects and, in the project, create multiple metamodels. The metamodeling framework provides an authentication page where users can login in system. The drawing functionality is implemented using the mxGraph library.

Once the metamodel has been created, a configuration file is generated. During the generation process, a JSON file is created, which needs to be imported into the backend of the modeling tool. Figure 5 shows a sample JSON file. The JSON file is generated using the metamodeling tool backend which loads the data from the database and returns the file to the metamodeling frontend. The backend also provides the possibility to create, update, read, and delete projects, metamodels and objects using REST API and provides authentication using JWT library. Metamodeling backend is built using the Express framework.

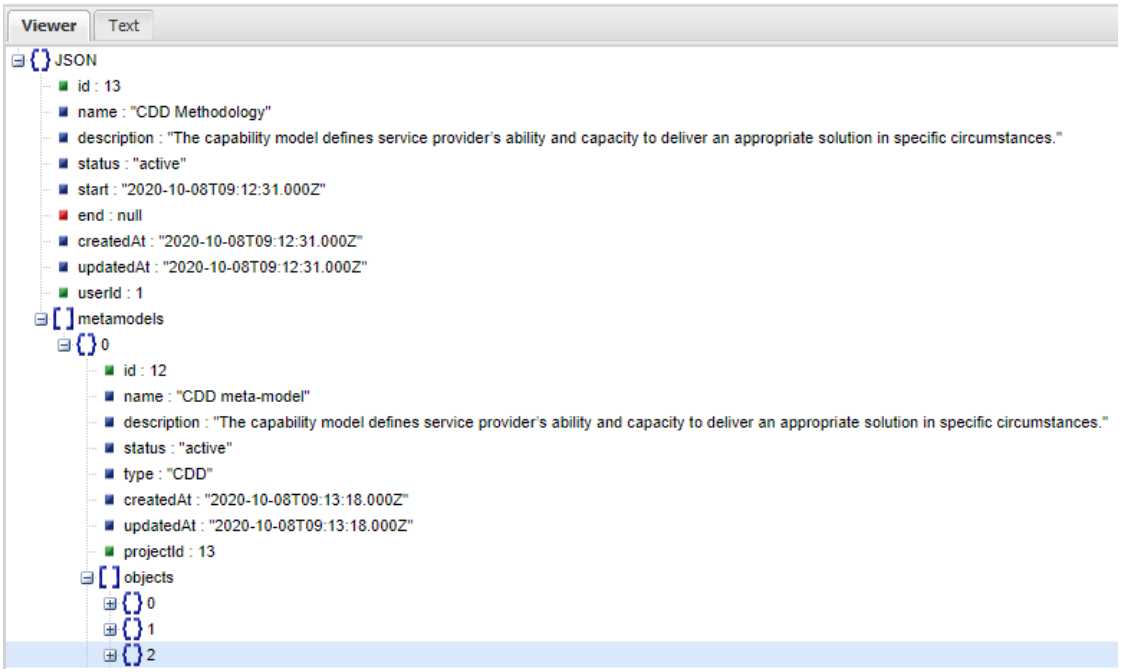

Figure 5: The generated JSON configuration file

The modeling tool frontend uses REST API create call to the backend to get metamodel objects. The modeling tool backend with API call load JSON file and returns the objects. As a result, a new web modeling tool becomes available and can be accessed by modelers on the Internet (Figure 6). Modeling frontend is built using the mxGraph ready drawing tool template. It is built on top of React framework. Tools consists of modeling tool, drawing container, a sidebar for drawing elements, a menu bar, an additional toolbar, and the right sidebar, where you can edit the modeling tool settings and element settings. The modeling backend is also built using the Express framework and provides REST API. It includes also JWT and Sequlize libraries which provide authentication and queries for database.

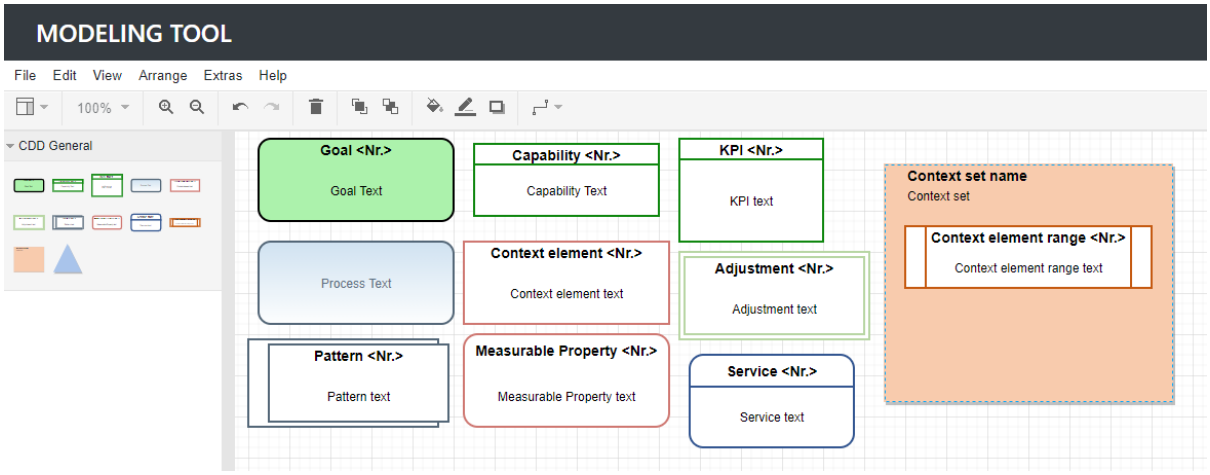

Figure 6: The generated modeling environment

The generated modeling tool is used to create capability models. A sample capability model considers the secure supplier onboarding case investigated in [17]. The case considers an ICT product called IoTool, which is a lightweight IoT gateway. The IoTool solution uses out-of-the-box devices which are not 100% secure. Using a capability-based approach it is possible to preserve data privacy pre-vent using sensing devices for DOS or similar malicious activities, and to provide the desired features requested by customers.

Figure 6 shows the create model of the secure sensing capability. The capability has four subgoals. The goal of secure sensing capability – goal 2. To preserve data privacy – goal 3. To have an appropriate risk level – goal 4 and to provide the desired features – goal 5. To see how we can achieve the desired goals, KPI are defined. To ensure product supply chain, a context is defined in terms of how the vendors and devices are characterized by trustworthiness (Context element 1) and vulnerability (Context element 2). Using measurable properties (Measurable property 4.1.1, 4.1.3, 4.2.1) are possible to evaluate the context. Both internal and external data sources are used. The Product vulnerabilities and Member trustworthiness assume values in the range 1..10 with categorical ratings. These context element range elements are included in the context set – secure sensing and are affected by capability 1.1 which is a secure supplier onboarding capability. This capability provides product configuration (Service 1) and supplier selection (Service 2).

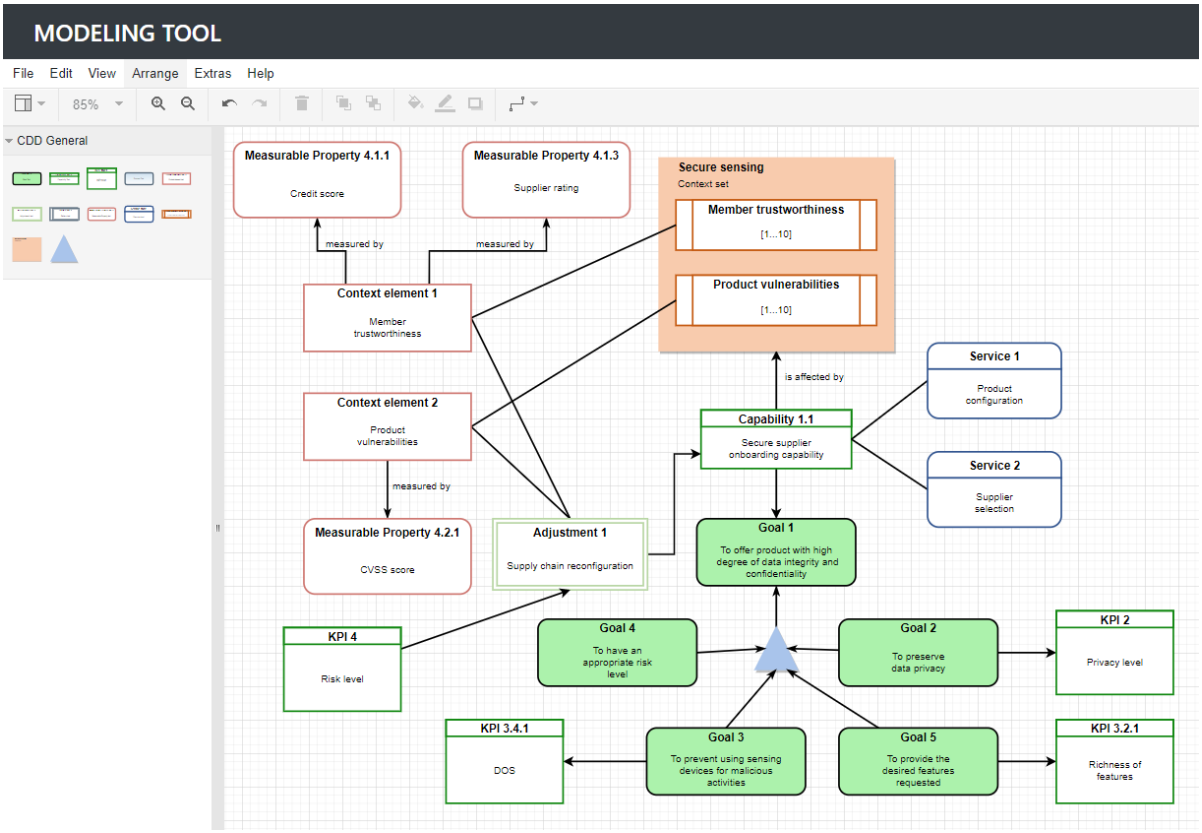

Figure 6: CDD modelling example implementation

5. Conclusion

Currently, the first adapted modeling tool has been developed in the ARTSS project using the CDD metamodel, thus confirming that the solution is easy to integrate and adapt.

All general requirements were met. To enable SSL for the system certificate, the system must assign a domain name. As the tool was adapted to the needs of the ARTSS project, the project was implemented under the project domain with a reverse proxy server, which in turn provides an SSL certificate. The system works in most browsers and can also be installed in Linux environments. The platform works well on a LAN, and the platform's response time in data entry forms does not exceed

five seconds. The interface language is English, and the system also provides UTF-8 encoding. The architecture of the developed platform is flexible, and it is possible, if necessary, to scale.

The front end of the metamodeling tool provides most of the requirements. It does not only provide a library of metamodel templates, but it does also not provide a library of graphical designs for objects, and it does not provide model updates in the case of metamodel updates. These requirements were not given such a high priority in the first version of the platform, so they were not implemented. The front end has a simple and easy-to-use interface and can create metamodels. Multiple metamodels can be created in one project, and metamodels can be both exported and imported for a specific project. Importing and exporting is provided using the JSON structure. The tool provides the ability to import a graphical design for each metamodel element.

The back system of the metamodeling tool provides the main basic functionality and fulfillment of requirements. The main functionality that the backend had to provide was the handling of API calls and the storage of data in a database. The system provides user authorization and registration using API and JWT library. The model generation requirement for the modeling tool was only partially implemented. In essence, in the first version of the prototype, it only provides the generation of a configuration file, which in turn is used by the components of the modeling tool. The metamodeling backend does not provide a metamodel base. Like the frontend, the base would be formed when the tool is used.

The front end of the modeling tool provides basic functionality. With the help of the system, it is possible to perform modeling from the created metamodel elements. The system also provides import of models in mxGraph XML format and exports in PNG, GIF, JPEG, PDF, SVG, and XML format. Because the system is built with the React framework, the system provides application speed and performance. You can change the location and position of model elements. The modeling frontend does not provide an automatic layout.

The last component of the platform is the back system of the modeling tool. The system provides API service processing and data storage in the database. The first version of the prototype did not provide integration with other systems and simultaneous model development. To implement the simultaneous development of models, it is necessary to study additional literature and review possible technologies and solutions, which were not performed within the framework of this work.

Compared to other solutions, the currently developed system is better in that it is web-based, better integrated with other solutions, scalable, provides good performance, and no need to install specific clients. In order for the platform to implement the requirements that are already present in the existing solutions, further development of the platform would require, which do not discuss as part of this work. The modeling tool needs to be improved with multiuser support and automatic layout capability. It would be necessary to introduce a graphic drawing of objects on the platform, add attribute components for each element, for example, so that you can define multiple text fields. It would also be necessary to introduce model restrictions, relationships between two elements and add the model's transformation, code generation, and integration functionality.

Acknowledgments

This research is funded by the Ministry of Education and Science, Republic of Latvia, project ARTSS, project No. VPP-COVID-2020/1-0009.

References

[1] Object Management Group (OMG), "Meta-modeling and the OMG meta object facility (MOF)," pp. 1–7, 2017.

[2] D. Karagiannis and H. Kuhn, "Metamodeling Platforms," *in Proc. of the E-Commerce and Web Technologies: Third International Conference, EC-Web 2002*, no. 2455. p. 182 ff, 2002.

[3] R. C. Gronback, *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. 2009.

[4] H.-G. Fill and D. Karagiannis, "On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform," *Enterprise Modelling and Information Systems Architectures*, vol. 8, no. 1. pp. 4–25, 2013.

[5] D. Karagiannis and N. Visic, “Next generation of modelling platforms,” *Lect. Notes Bus. Inf. Process.*, vol. 90 LNBIP, pp. 19–28, 2011.
[6] P. De Smedt, “Comparing three graphical DSL editors : AToM 3 , MetaEdit + and Poseidon for DSLs,” vol. 3, no. 2.
[7] K. Smolander, K. Lyytinen, V.-P. Tahvanainen, and P. Marttiin, “MetaEdit— A flexible graphical environment for methodology modelling,” 1991, pp. 168–193.
[8] M. Esperguel and S. Sepulveda, “Feature modeling tool: A proposal using ADOxx technology,” in *2016 XLII Latin American Computing Conference (CLEI)*, 2016, pp. 1–9.
[9] N. Boldt and D. Steinberg, “Introduction to the Eclipse Modeling Framework,” pp. 1–89, 2006.
[10] A. Ledeczi *et al.*, “The Generic Modeling Environment,” in *Workshop on Intelligent Signal Processing, Budapest, Hungary*, 2001, vol. 17, pp. 1–14.
[11] I. Zikra, J. Stirna, and J. Zdravkovic, “Analyzing the integration between requirements and models in model driven development,” *Lect. Notes Bus. Inf. Process.*, vol. 81 LNBIP, pp. 342–356, 2011.
[12] J. Grabis, J. Kampars, K. Pinka, and J. Pekša, “A Data Streams Processing Platform for Matching Information Demand and Data Supply,” in *Advanced Information Systems Engineering*, 2019, pp. 111–119.
[13] “Angular vs. React vs. Vue: A performance comparison - LogRocket Blog.” [Online]. Available: https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/. [Accessed: 09-Oct-2020].
[14] “mxGraph User Manual - JavaScript Client.” [Online]. Available: https://jgraph.github.io/mxgraph/docs/manual.html. [Accessed: 09-Oct-2020].
[15] H. Shah and T. R. Soomro, “Node. Js Challenges in Implementation,” *Glob. Journals Inc*, vol. 17, no. 2, 2017.
[16] S. Berziša *et al.*, “Capability Driven Development: An Approach to Designing Digital Enterprises,” *Bus. Inf. Syst. Eng.*, vol. 57, no. 1, pp. 15–25, 2015.
[17] J. Grabis, J. Stirna, and J. Zdravkovic, “A Capability Based Method for Development of Resilient Digital Services,” *Sel. Pap. ICEIS 2020*, 2020.
[18] H. Wang, G. Wang, J. Lu, and C. Ma, “Ontology Supporting Model-Based Systems Engineering Based on a GOPPRR Approach,” in *Advances in Intelligent Systems and Computing*, 2019, vol. 930, no. April, pp. 426–436.
[19] K. Sandkuhl and J. Stirna, *Capability management in digital enterprises*. 2018.