

The Simplified Platform, Cases 2024

dr. ir. Mark A.T. Mulder¹, Rick Mulder

¹TEEC2, Hoevelaken, the Netherlands

Abstract

The Simplified platform is the web-based approach to modelling and meta-modelling. This platform started from the experience with a previous research tool for modelling Design and Engineering Methodology for Organisations (DEMO) and has increased the available notations to OntoUML and ArchiMate. The cloud-based platform's extension ability makes it suitable for research and business applications. The platform's configurable notations, flexible user interface, and real-time transformation, verification, and visualisations make it adaptable and understandable for every stakeholder. This update paper will list the current state of the simplified platform.

Keywords

Modelling, Meta-modelling, Collaboration, Enterprise Engineering

1. Introduction

The history of the Simplified modelling platform started with the research project towards the PhD 'Enabling the automatic verification and exchange DEMO models' [1]. The DEMO [2, 3] method is a core method (based on a theoretically founded methodology) within the discipline of Enterprise Engineering (EE) [4]. We have described the history in detail in our previous paper [5].

The lack of good tooling for demo modelling prompted us to start the development of a new tool. We created a cloud-based modelling platform which supports collaborative design, multiple notations, API and white label UI integration that would allow customers to apply their own corporate design language to the UI and multiple languages, all while applying state-of-the-art development methods.


The platform consists of total of six layers, divided in two servers: application server (involving the layers interface, message, process, cache, and persistence), and database server (database layer). The interface layer consists of two interfaces for accessing the platform. Further information on the architecture was published earlier [6]. The modelling part of the back-end is designed to store the model and the metamodel of a notation or methodology. The architecture uses dynamic metamodels that restrict the models on run-time. The notation architecture structure is visualised in fig. 1. Developers and Users use different terminologie, causing you to look at the same thing from two different perspectives as seen in fig. 1. Whereas the user defines a model, the developer only sees the metamodel being instantiated. The platform is now available under licence on <https://simplified.engineering/>.

BI Week 2024 Tools, , November, 2024, Vienna, AU

✉ markmulder@teec2.nl (dr. ir. M. A.T. Mulder)

🆔 0000-0002-1846-0238 (dr. ir. M. A.T. Mulder); 0000-0002-9500-0702 (R. Mulder)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

This paper describes six relevant cases we did in the past year. These cases will be reported in the STARR format. Finally, we conclude with a summary of this platform’s current state of affairs.

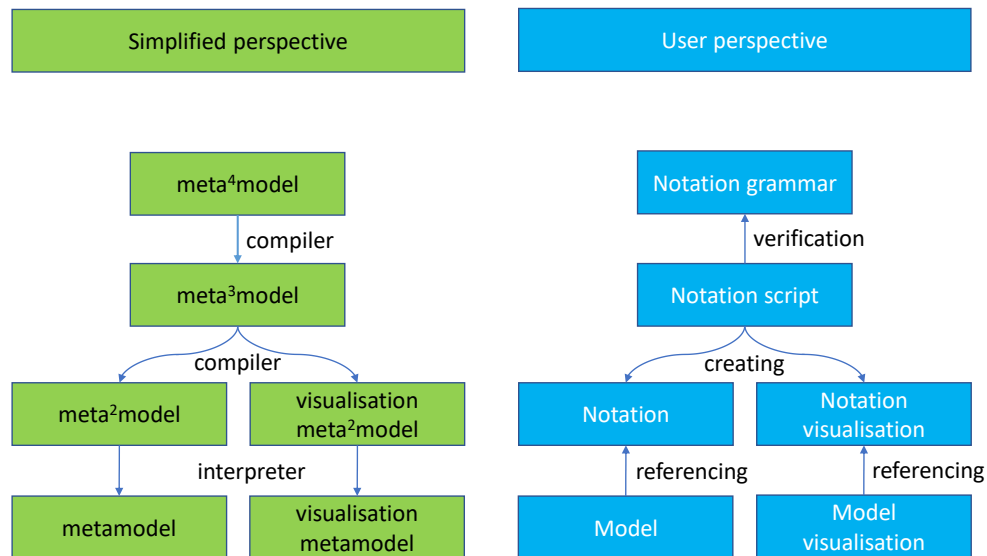


Figure 1: Notations from different perspectives.

2. Cases

In this section, we will discuss several cases we encountered in the last year. We will use the STARR reporting for these cases.

2.1. Governmental

The first case contents is still confidential but we can give some generic remarks.

Situation Our customer needed a research project to get an answer on the integration of Simplified Modelling Platform (SMP) within their own environment. Their desire to switch to SMP was caused by the inability of their current tools to solve their pain points.

Task We have developed a server extension that is able to convert a model, complying to a custom notation, into a JSON format, and push this JSON to a Git environment. The process involves two steps: First, the model is created in within a simplified repository and model environment. Subsequently, while the server extension is running, the user brings up the context menu for the diagram and chooses the method to convert to JSON and publish to a gitlab environment. In the background a message is sent that lets the server know that the user wants to run a function in an extension execute a task. The server forwards this request to the extension along with any possible parameters needed for this operation. The server extension then receives this request and executes

the function, requesting additional information from the server. The extension then creates the JSON and commits this to the Git server. The Git server can then start the CI/CD process upon receiving a new JSON being uploaded to the GitLab.

Approach We have used an iterative approach to create the right environment to customise the notation and create the server extension step by step to meet the demands.

We started by analysing their problem step by step, translating them into tool solutions. With the tool solutions we modelled their problems, until we encountered the next problem, restarting this process of analysing.

Result The custom notation and the server extension have been created and the integration between the SMP and Git environment have been established with a server extension under control of the customer.

Reflection While the process is still ongoing, we have learned about creating custom notation in a modelling environment, just like the Problem-Solution Chain (PSC) case. The integration with multiple Git environments have brought good opportunities to store models in a vendor independent location.

2.2. Problem-Solution Chains (PSCs)

For the previous case and this case, that is to be published later in detail, we created a new way of creating a notation.

Situation A colleague researcher approached us to model a research problem executed by a master student. No other existing tools could quickly and easily create a custom notation. The core idea behind PSCs is that basic but solid ‘problem-solution thinking’ is key in the early stages of problem-solving, including business-IT alignment efforts linking business needs and IT functionalities. The use of Problem-Solution Chains (PSCs) mainly aims to support steps 2-4 in the six-step approach of [7].

Task We needed to create a domain notation and a domain model that represented the research topic PSC.

Approach During interactive sessions we created the notation that complied with the research topic.

Result The new method allows for creating a model that represents a notation using a custom notation specification. By choosing the ‘elevate model’, a function that will convert a modelled notation, as shown infig. 2, into a useable notation on the platform, one can realise, within the same environment, a new notation that is active right away. This allows for easy adjusting and designing new notations in research projects.

Reflection During the interactive session we found that the creation of the notation as a script did not met the needs of the stakeholder. Therefore, we added the creation of the notation as a model. This will be published later this year.

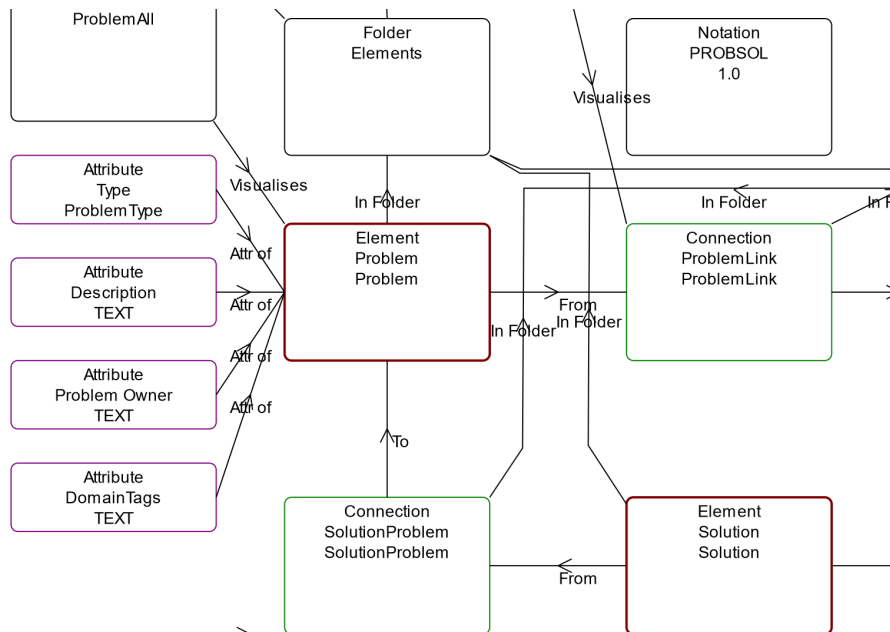


Figure 2: Part of PSC Meta Model in SMP

2.3. Staging generation with transformations between JSON - DEMO - IG

In this case we created a server extension that supports model transformation allowing for the creation of a datawarehouse staging and reporting environment within the InformationGrid platform.

Situation For a customer we created the storage of information in the IG environment. This information acts as a datahub and a datawarehouse from where the datamarts are populated.

Task The process starts with the import of an one level deep JSON structure that is converted to a DEMO Entity Type (ET) that is usually found on the fact diagram. This ET is converted from the source model in DEMO notation to the target model in InformationGrid (IG) notation components in simplified, mirroring the structure in IG itself. This is automatically generated and converted to several components in the low-code platform, using an API to create the components. This pipeline allows for changes to the source model, to be forwarded to the platform in an automated process.

Approach After doing this process manually several times, we decided to automate the transformation. Using the existing examples we deduced the format and created the transformation algorithm.

Result The generation of the staging area results in a reduction of programming from two to three hours to five minutes and a reduction of error to almost zero. This is shown in the hours we have spent, creating solutions for the customer before and after creating the generation.

Reflection The generation of this staging area was a good step towards the complete generation of DEMO entities to the IG environment project that is running now. The generation has had a

single update since its first inception to optimise the performance of the IG system by using the build-in indexes.

2.4. VISI generation to IG

VISI is an ISO-standard ¹ for a workflow notation and execution used in the construction branch ². Executing a workflow in the IG environment ³ is possible and can be generated.

Situation At the start of this project there were 3 suppliers of software that could process VISI messages of the VISI framework. Target of this process is to be able to run a domain driven design model to generate the operational part of the software on a low/no-code platform.

Task The project consists of the transformation of the VISI model to the generic format of the SMP. Thereafter the transformation of the VISI notation to the InformationGrid 2022 (IG22) notation will create an potential executable model of the VISI framework. The upload of the IG22 model to IG creates an callable API that can receive and produce messages. The server extension on SMP will complete the communication between IG and the SOAP interface. A special UI interfaces between the users and the information in IG.

Approach We are iterating over all parts of the modelling and transformations needed to make a complete executable form of the model. The project is very dynamic as the VISI standard also evolves during the project.

Result The current state of this case is that the model has been transformed from the exchange XML to the IG model and needs the right preconditions to be able to execute the messages within the IG platform.

Reflection This project took longer than expected because of the changing preconditions, changing standard and developments in the target platform. We hope this project is finished end 2024.

2.5. Business Process Model and Notation (BPMN) transformation to System Dynamics (SD)

The fifth case is the guidance of the creation of a BPMN-SD transformation server extension. This was the first time an external party had created a simplified server extension.

Situation We wish to transform a BPMN model to SD. This is a research project for a master student. The mapping between BPMN and SD was expected to exist, and would be able to be programmed. Challenges lie in finding the exact mapping, which will not be one to one.

Task The main objective of the project was to create a server extension that was able to transform a BPMN model in Simplified into a SD model in simplified. This was a masters project and the duration was a little over a year.

¹<https://github.com/nl-digigo/visi/tree/master/VISI%201.6/Documentation>

²<https://www.digigo.nu/standaarden/visi/>

³<https://nl-digigo.github.io/visi/visi1.6>

Approach As a start we provide a template for the server extension. This contains the necessary parts for connectivity, communication, and a library of callable methods in order to retrieve data from Simplified. Close contact was kept with the master student through the use of discord. In order to better support the transformation, we created a new transformation method from the ground up. The transformation was designed so that it can transform an arbitrary number of elements in a tree structure to another arbitrary number of elements in another tree structure. This allowed for the freedom to do a 1-many or many-many mapping as needed, in addition to 1-1 mappings.

Result The result is a server extension that transforms BPMN models to SD models. We have created a generic transformations for Chain to Chain transformation and enhanced the notation uploading and visualisation.

Reflection During this exercise we upgraded the documentation on notation scripts because it was unclear how to use them. We also created a new way of uploading notations allowing for deletion of old, and creation of new attributes with the relink option to keep models valid while updating the notation due to research insights. Next, we added metamodeller UI changes, such as last script uploaded, and error messages handling to provide feedback, and predefined variables and a easy way to add variables to test your shape. Furthermore, we allowed the use of variables in for example rectangles. A big step was the addition of the transformation of a tree of elements into a tree of different elements was created, called chain to chain. Finally, we learned that it was very beneficial (or required) to have basic programming knowledge in the way of setting up programs, splitting functions instead of writing big functions, and creating sections based on actions you want to accomplish.

2.6. Glossary Listing

Glossary was requested due to a wish to have the terminology in one way for the modeller and a different way for stakeholders.

Situation A customer needed a glossary list for definitions in the company. This glossary would be used to define the information of parts of the model, but also information that did not directly relate to any model in the SMP.

Task We needed a way to create, edit and view all definitions within a glossary. To make this happen we have created a notation 'glossary' with the element 'glossary item' that can hold definitions. A separate functionality can later transform these definitions in linkable HTML formatted texts. We split the functional UI into two parts, a creator for terminology and a display, where you can also edit them. The creator can create a single item in the appropriate location in the model. The display will list all items of the same type in a default dynamic table. We created this dynamic table to also edit specific types.

Approach We have started with a basic design that can do all fundamental functionality. In the validation of the functionality with a focus group we have noted extra functionality that will be implemented in the next iteration.

Result While the project was intended to create a glossary list, the generic build of SMP allowed us to reuse this functional UI to create model elements without visuals or diagrams.

Reflection We have added a group of stakeholders to the platform using an UI that represents the thinking of business lists. Definitions can be added in the glossary without the need to know how to model. The list can also be queried (through simple filters for now) and pave the way for more complex queries to the model environment.

3. Conclusion

The platform, Simplified, takes away more and more limitations that we experienced during modelling of organisation and information. It supports the collaborative design and multiple notations expressed in multiple languages. The extension of features proves to be done in a modular fashion, allowing for future feature development without disruption. Additional research is needed to extend the features of modelling and a broader investigation of the limitations and gaps of other modelling tools is yet to be conducted.

We encountered a lot of supported projects that reveal the need of changes to modelling and modelling tools. We are positive that we can support these research and commercial needs for modelling.

Finally, a lot of features that come with specific modelling methodologies will be added.

References

- [1] M. A. T. Mulder, Enabling the automatic verification and exchange demo models, 2022.
- [2] J. L. G. Dietz, Enterprise Ontology – Theory and Methodology, 2006.
- [3] J. Dietz, H. Mulder, Enterprise ontology: A human-centric approach to understanding the essence of organisation, 2020.
- [4] J. Dietz, J. Hoogervorst, A. Albani, D. Aveiro, E. Babkin, J. Barjis, A. Caetano, P. Huysmans, J. Iijima, S. J. V. Kervel, The discipline of enterprise engineering, 2013.
- [5] M. A. T. Mulder, R. Mulder, F. Bodnar, M. van Kessel, J. Gomez Vicente, et al., The simplified platform, an overview, 2022.
- [6] M. A. T. Mulder, H. A. Proper, F. Bodnar, R. Mulder, Simplified enterprise modelling platform architecture (2022).
- [7] J. Luftman, T. Brier, Achieving and sustaining business-it alignment, California Management Review 42 (1999). doi:10.2307/41166021.