The Simplified Platform, an Overview

dr. ir. Mark A.T. Mulder [1], Rick Mulder, Fiodor Bodnar, Mirjam van Kessel, Jorge Gomez Vicente

Abstract:

Simplified is a new approach to modelling and meta-modelling. This platform is a result of experience with a previous research tool for modelling Design and Engineering Methodology for Organisations (DEMO). It's cloud based development makes it suitable for research and business applications. The configurable notations, flexible user interface, and real-time transformation and visualisations makes the platform adaptable and understandable for every stakeholder.

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' [Mu22]. The DEMO [Di06, DM20] method is a core method (based on a theoretically founded method*ology*) within the discipline of Enterprise Engineering (EE) [Di13]. The DEMO methodology focuses on the creation of so-called *essential* models of enterprises. The latter kind of models aim to capture the organisational essence of an enterprise by leaving out (as much as possible) details of the socio-technical implementation. The organisational essence is then expressed primarily in terms of the actor roles involved, as well as the business transactions [Re96] (and ultimately in terms of speech acts [Ha81]) between these actor roles. More specifically, an essential model comprises the integrated whole of four aspect models: the Construction Model (CM), the Action Model (AM), the Process Model (PM) and the Fact Model (FM).

As part of this work, requirements for tooling were defined in order to find the most suitable tool for DEMO modelling. Among other requirements, the tool would have to support the essential model of an organisation. Unfortunately, no suitable tooling was found that could support all the requirements.

The lack of good tooling for demo modelling prompted us to start the development of a new tool. This initiative resulted in the creation of *Plena*, an add-on to an existing Sparx modelling software. This add-on supported the import of the interview lines along with the modelling of all aspect models. During the further development of Plena, it became apparent

[1] TEEC2 BV, The Netherlands markmulder@teec2.nl

that the underlying Sparx modelling software was increasingly becoming a limiting factor which impeded the implementation of the desired features. Among other features, we wanted to support collaborative design for online team modelling, 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.

Considering the limitations of the Sparx software we have decided to develop a standalone cloud based modelling platform from scratch, which would allow us to implement all the required features. The resulting platform is called the *Simplified modelling platform*; or *Simplified* in short form.

It is worth mentioning that [GBDV20] concluded that our previous tool Plena scored quite high on its usability. However, Plena lacked when it came to usability criteria ‘U3’ *‘Ease of learning’* and ‘U6’ *‘Reduction of excess’*. This has also been addressed in Simplified and will be later discussed in more detail.

This paper describes the used architecture and the implemented interfaces. After that a series of features are introduced and respective benefits for research, business users of organisations, and modellers in general are introduced. We also provide a brief overview of all technologies used in the development and production environments. Finally, we conclude with a short summary and the current state of affairs of this platform.

2 Architecture

The platform consists of total of six layers, divided in two servers: application server (involving the layers interface, message, process, cache, and persistence), and finally database server (database layer). The server architecture is visualised in Fig. 1. The layers are visualised in Fig. 2

First, the interface layer consists of two interfaces for accessing the platform. The REST/JSON API is the simplest interface, allowing the retrieval of the generic public information that requires no authentication. No modelling information can be exchanged using this API. This information includes the installed public notations (e.g. ArchiMate, DEMO, BPMN) and the creation of free accounts. The other interface, that facilitates the access to the platform is an authenticated asynchronous web socket messaging interface that can not only receive and handle messages, but also can broadcast to all relevant connected users. This interface can be used by authorised developers to build their own user interaction. Next, the message layer handles messages either by using the process layer for partial message processing, or by forwarding the request to the distributed cache in the cache layer. The cache layer contributes to fast retrieval of information and distributes the load between multiple servers. This cache layer will, in turn, send data or initially retrieve the information from the persistence layer. The configurable persistence layer will activate the right persistence provider that triggers the correct driver to communicate information to the database layer. Lastly, in theory the database cluster realising a database layer can be any database, but at the moment it is limited to the Microsoft SQL server and to the open source MySQL server.

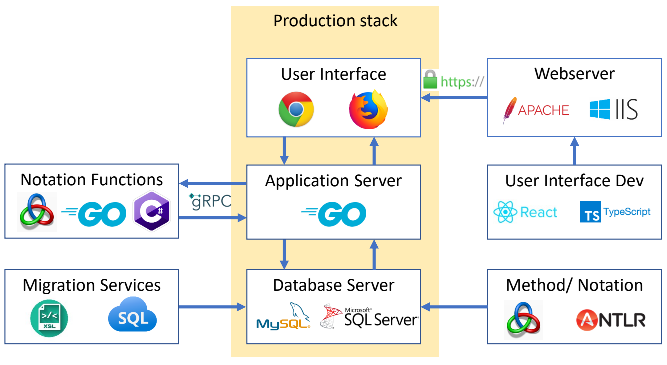


Fig. 1: Architecture of Simplified

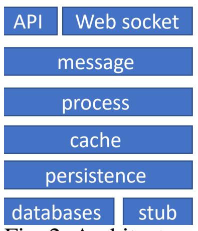


Fig. 2: Architecture Layers

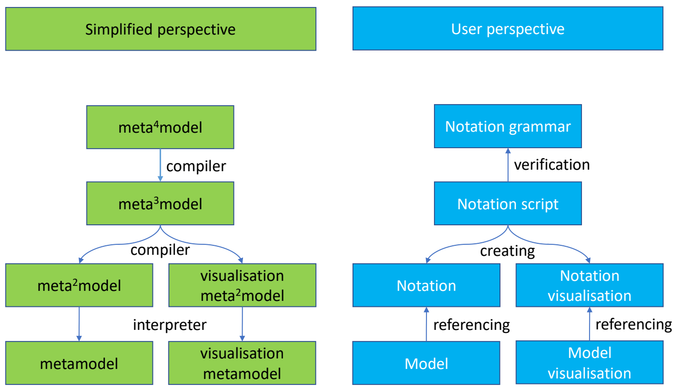


Fig. 3: Notations from different perspectives

Messaging to the asynchronous interface is done in JSON format with a dynamic payload structure, allowing for per message configuration, and developing structures during the lifetime of the platform. This messaging structure is used for the user interface in any form. Other clients can use this structure to interface with the back-end engine for notation specific operations. Every client, that is also a server, can request the platform to forward special operations upon user request. The user request results in a gRpc call to the providing server platform that returns the calculated information. This information is then processed and returned to the client UI for the visualisation.

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. 3. To be able to create a model and a visualisation of that model one has to agree on a notation for both aspects. This results in creating a meta-model. To be able to describe this meta-model we have created a script language that can describe notations. This script language is described in a grammar and the levels of abstraction are reflected in the platform and the transformation between the levels are either interpretation or compilation. Below you can find an extract from the grammar for the 'DEMO 3' notation (the '...' contains a lot more definitions).

```
ScriptVersion01 Notation for DEMO version 3.7
typedef TRSRT ENUM (None, Original, Informational, Documental)...
element "Transaction Kind" TK37 ( ... , "Transaction Sort" TRSRT) ...
connection Executor Executor37e from TK37 to EAR37 ...
diagram "Org... Constr... Diagram" OCD37 contains (... TK37, ...)
table "Transaction Product Table" TPT37 select (x."Identification", ...
visual TKVisual37ocd of TK37 on (OCD37) { ...
fillcolor(255,255,255) // White Filling ...
print(10, 20, 40, 25, "{element.identification}", 0) ... } ...
```

The platform will be available in the middle of June 2022 on https://simplified.engineering/.

3 Features and Benefits

We observe that some of the features mentioned below should not be regarded anymore as distinctive tool features in the current state-of-the-art because many tools on the web have this same set of features [Lu22, Ca22, JG22, Ed22, Go22b, Nu22, Cr22].

Therefore, we conjecture that these features constitute a baseline for the modelling tools. The following items give a summary of the baseline features, but more items could be added outside the scope of this paper.

- Platform: login with username and password, activity logging, activity audit.
- User interface components: repository browser, notation toolbox, showing object properties, search option.
- User interface behaviour: remembering layout preferences per user, moving all docking components.
- Modelling behaviour: automatic saving, available notation elements and connections, drag and drop elements from source to the canvas, naming elements and connections, showing next most logical steps, making notes, moving elements and connections on the canvas (single or groups), resize the elements, start and end connections on any place of the border of an element, making anchors in a connection line, duplicate elements and connections, adding attributes to an element, aligning elements.

Next to these baseline features, Simplified has its own distinctive features described in the next paragraphs.

All users working on the same models, automatically collaborate. Simplified adheres to the concept of optimistic locking, or rather the last save wins. The 'what you see is what you saved' concept ensures that all users have the same view of the model. Given the fact that there can be multiple views of the same model catering the needs of various stakeholders, a change in the underlying model automatically leads to changes in the respective views of the model, keeping them up-to-date. Viewers can also follow changes being applied to a live version of the model from their own viewpoints.

With Simplified one can support one's own modelling methodology and associated notation. The notation can be transformed to a notation script that allows for specifying the elements, connections, visualisations and restrictions for the model. Creating a notation can be done from scratch or one can build on an existing notation. Furthermore, predefined notation scripts for DEMO, BMPN, ArchiMate, VISI and IDEF0 are available on the platform. Other notations such as PetriNet, ER, UML, etc. can be added on request.

Modelling in a multi cultural environment requires a user interface that can be used in multiple languages. Multilinguality is built into the UI and at the moment Simplified supports five languages, but this can easily be extended. Having said that, the model itself should be translatable too. Simplified allows the model and all of its elements to be manually translated to all the required languages in order to present the model to users as comprehensively as possible.

During the development process of the Simplified modelling platform we have taken into account usability requirements [Na12, GBDV20, Mu22]. Firstly, 'ease of learning' is significantly improved when compared to its predecessor Plena as it is no longer an add-on to an existing modelling tool, but is a standalone cloud based modelling environment. As a result, there is no such underlying requirements as installation of software and plugin imports which contribute to the ease of learning. Secondly, for the 'reduction of excess' we have now full control over the UI and what is going to be shown to the user. Moreover, users get a freedom of choice on how their UI layout will look like as all windows within the UI are dockable. For instance, this gives the user a possibility to place a 'repository browser' or 'toolbox' docking window to a desired position on the screen.

In the Architecture section we touched briefly upon the use gRpc [Fo22] for support of the special operations. This interface allows for cross-programming-language development and we have used it to let remote servers subscribe to our server to link to actions in the user interface. gRpc is used to create a lightweight synchronous protocol between servers instead of the asynchronous web socket. gRpc extensions can be used not only in the new models but also in the existing ones. For example, model validation logic can be imported and run against the model to check for its compliance with the underlying notation.

For the launch of the tool in the middle of June 2022 we are considering four standard and one custom licence types to be available to the users of Simplified. Standard licence offering will include the Basic, Standard, Pro, Enterprise, and Research&Education versions [2]. It is worth noting that Basic licence type will provide a free, thus low barrier entry point to the modelling environment with basic functionality to the user. We categorise Basic and Standard for personal, and the rest for professional use.

4 Visualisations

Simplified is a server based solution which supports models and visualisations of those models [Mu22]. Therefore, new technologies that want to connect to those visualisations can do so by using the messaging system. In the end, the visualisation can be summarised by a view with visual elements having a x,y,z coordinate and a size. Those properties are supported and can be extended within the platform. Systems that use views for gamification, mobiles, desktops, AR, VR, or live translations for stakeholders can potentially be connected. We already have a gamification connected to the system. The current modeller is a web page supporting all 2.5D modelling and a white label offering. White labelling helps with fast integration of Simplified platform functionalities in your own environment, leads to reduced investment and decreased time to market on the modelling features.

4.1 Modeller

Simplified Web UI is built of (web) pages in a modular fashion. Every set of features has its own page. Each page is built with modular function blocks which share a state with the other modular function blocks. Each modular block has its functions and a visual representation. Next, the modeller page consists of the visualisations of a modular function block that have a docking possibility within the page. In addition to sharing a 'session' state between some pages, pages have their own states as well. As explained in the next sections, examples of modular blocks are Toolbox, Browser and Properties. The page layout is saved as a session variable, which allows each user to have its own unique layout. Due to the modular nature, it is also possible to have stakeholder layout presets. Other pages that do not have the docking layout, still use the modular setup, which makes it possible to customise each page to specific needs of the stakeholder.

4.1.1 Browser

The browser has 2 modes: a tree mode and a list mode. The tree mode displays the structure of the project, with models at the top. Under them are the folders, which in turn contain elements, diagrams and other folders. Using the switch view button we can switch to the list mode. The list mode displays all elements and diagrams in a list. This also enables 3 extra

[2] https://teec2.nl/products/modelling-platform/

drop down fields where we can filter on elements we want to see in our list. In addition to that, we can have multiple browsers, each in different modes, and link them to specific property dockers. These browsers, each with their own view, will all have the up-to-date data displayed in their own way as shown in Fig. 4.

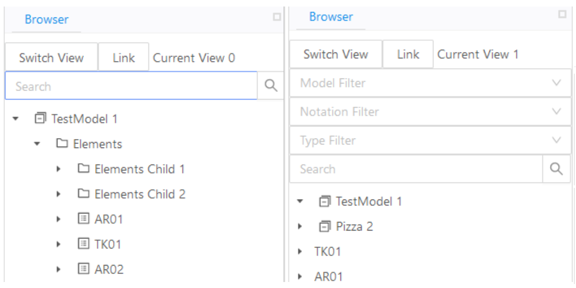


Fig. 4: Different ways to display elements in the browser

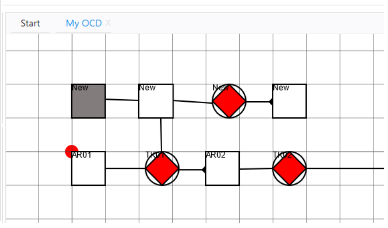


Fig. 5: The diagram view

4.1.2 Diagram

The diagram is arguably the most important part of the modeller. A diagram can be opened from the browser and represents a graphical view or visualisation of a part of the model following the defined viewpoint of the used notation as presented in Fig. 5. Element drawings are requested from the server on demand. When an element cannot be found or it does not exist, it will be displayed as a red cross with a box around it. This way you can still interact with the element with basic functionality, such as moving and scaling. All actions made by users on the diagram are displayed in real time to other people working on the same diagram. This allows for smooth collaboration and modelling together.

4.1.3 Toolbox

The toolbox contains the notations that are available to the user as visualised in Fig. 6. These notations have categories, of which two are standard, namely connections and elements. In a notation we can specify custom categories, which contain a subset of elements. By doing so we can ensure that the user can have quick access to their most important elements. The toolbox can be refreshed, which comes in handy in case of an updated notation or a new notation that is not yet shown in the toolbox. This is especially useful when a co-worker updates a notation and you want to immediately see the new updated graphics for example. Missing drawings of specific elements will be displayed in a similar fashion as in a diagram with a red cross with a box around it.

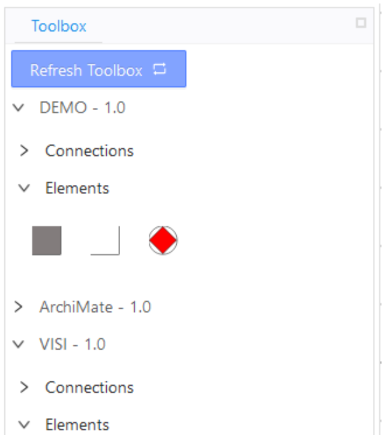
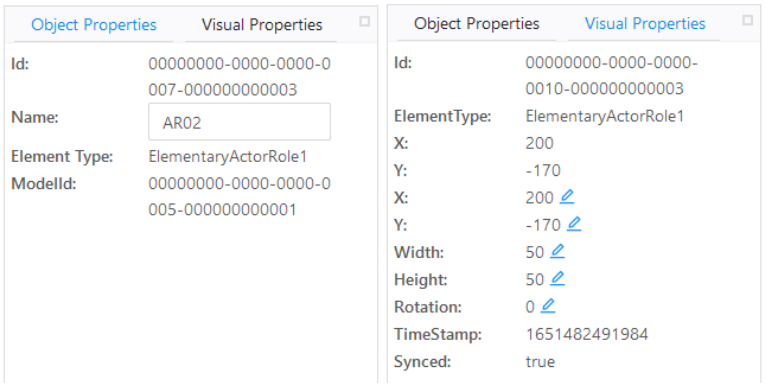


Fig. 6: Toolbox with several notations and their categories

4.1.4 Properties

The property module displays the properties of the selected element as visualised in Fig. 7. When a visual element is selected, in addition to the visual element properties, it also shows a separate tab with the element properties. This property module can be either a global or a linked property module. In case of a global property module it does not matter where an element is selected, its properties will be displayed in the property module. Whereas, with a linked property module, only the properties of the element that is selected in a specific linked browser will be shown. This way the user can have several browsers, each with their own intended usage, such as a broad overview, or a detailed focus.



Id:	00000000-0000-0000-0007-000000000003
Name:	AR02
Element Type:	ElementaryActorRole1
ModelId:	00000000-0000-0000-0005-000000000001

Id:	00000000-0000-0000-0010-000000000003
ElementType:	ElementaryActorRole1
X:	200
Y:	-170
X:	200
Y:	-170
Width:	50
Height:	50
Rotation:	0
TimeStamp:	1651482491984
Synced:	true

Fig. 7: The Element and Visual Element properties

4.2 Mobile & Tablet

Cross-platform is an important topic today[Sh16]. Due to the modular nature of Simplified, it is possible for mobile/tablet users to customise their interfaces in a way that works well with touchscreens. Since the website is developed mobile-first, there is no need for a separate app or download, thus further contributing to a lower barrier to start modelling from any device.

4.3 Gamification

Gamification is a way to explain information playfully while engaging the audience to participate. We have spent a considerable amount of time trying to communicate various aspect models to stakeholder groups and came to the conclusion that a different visualisation of the same model can facilitate the model understanding [MP21]. To involve the stakeholders interactively, the gamification interface connects directly to the platform. Therefore, it can operate in the same real-time fashion as the modeller does.

5 Development and production

For development we are using the IntelliJ IDE environment [Je22]. This supports the MySql [Or22] and MsSql [Mi22] development, as well as the Go [Go22a], React [Re22a], and Antlr [Pa22] languages. Even our own language, written in Antlr, can be supported during development. This IDE supports the whole team as the sources have been stored and shared using GitLab.

The web browser front-end is written in React and is using open source libraries to support the most common user interface components like docking windows and graphical shapes. React.js [Re22a] is an open-source JavaScript library, which is lightweight and it also provides a freedom to choose between different tools. React uses HTML (JSX) inside JavaScript which extends the functionalities of HTML structures into JavaScript.[Li22] JavaScript is fast, but updating the DOM makes it slow. Whereas React minimises DOM changes by monitoring the component's state with the Virtual DOM [Re22b] and finds the least expensive way to update the DOM.

The back-end is completely written in Go (for some people known as Golang). Go [Go22a] is an open-source programming language that is compiled to fast [Ga22] machine code. In addition, Go inherits the disciplined syntax of C with a feature to manage memory safely. The Go language has Goroutines. These routines are basically functions that can run simultaneously and independently, which makes it scalable. The Antlr grammar parts also compile to Go. The testing of the platform is done using unit tests in Go on the back-end, and Selenium tests procedures on the front-end. Manual testing is registered in Testmonitor[Ce22].

For production, all layers are run on open source components. The back-end is run as a docker configuration on Kubernetes. This allows for scaling of the application back-end. This application server connects to a MySql cluster for storage, thereby using the cluster principle to allow for scaling. Remote developers in companies or universities are independent of our development as they can run their servers against our platform to gain access to models and add functionality to the user interfaces.

6 Conclusion

The new platform, Simplified, takes away the limitations that we experienced during the development of Plena. It supports the collaborative design and multiple notations expressed in multiple languages. Though development has not been completed yet, the current version can hold all models for the following notations: DEMO, ArchiMate and BPMN.

Besides, up till now no restrictions have been seen that would impede the addition of more notations to the platform. The architecture layers are helpful in extending the support for more databases which we recently experienced while adding the MySql support. The extension of features can 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 have a road map of features that include, but are not limited to the following features:

- The referencing of models from a template, inheritance (advising or forcing) references.
- The support of visualisation concepts like swim lanes on diagrams, matrices, cubes, and automated layout features.
- User support on choosing the next step of the methodology.
- Advanced ruling that can efficiently verify models.
- Automated transformation between notations, and generation within notations and to other systems.

Finally, a lot of features that come with specific modelling methodologies will be added. We can conclude that the chosen direction of research and development is paying off as in a relatively short period of time we established a greenfield platform without restrictions for (meta)modelling support.

Bibliography

[Ca22] Camunda: , Web-based tooling for BPMN, DMN and Forms. https://bpmn.io/, April 2022.

[Ce22] Ceelen, Rene: , Test Management Software. Simplified. https://www.testmonitor.com/, April 2022.

[Cr22] Creately: , Business Process Mapping Tool. https://creately.com/lp/bpm-software-online/, April 2022.

[Di06] Dietz, J. L. G.: Enterprise Ontology – Theory and Methodology. Springer, Heidelberg, Germany, 2006.

[Di13] Dietz, J.L.G.; Hoogervorst, J.A.P.; Albani, Antonia; Aveiro, David; Babkin, Eduard; Barjis, Joseph; Caetano, Artur; Huysmans, Philip; Iijima, Junichi; Kervel, Steven JH Van: The Discipline of Enterprise Engineering. International Journal of Organisational Design and Engineering, 3(1):86–114, 2013.

[DM20] Dietz, J.L.G.; Mulder, H.B.F.: Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation. Springer Nature, November 2020.

[Ed22] Edrawsoft: , Unlock Diagram Possibilities. https://www.edrawsoft.com/, April 2022.

[Fo22] Foundation, Cloud Native Computing: , A high performance, open source universal RPC framework. https://grpc.io, April 2022.

[Ga22] Game, Benchmark: , Go costs versus Java costs. https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go.html, April 2022. Accessed: 2022-05-03.

[GBDV20] Gray, Thomas; Bork, Dominik; De Vries, Marné: A New DEMO Modelling Tool that Facilitates Model Transformations. In: Enterprise, Business-Process and Information Systems Modeling, pp. 359–374. Springer, 2020.

[Go22a] Golang: , Build fast, reliable, and efficient software at scale. https://go.dev/, April 2022.

[Go22b] Google: , WebEA - New Tabbed Interface. https://sparxsystems.com/products/procloudserver/5.0/index.html#9AEC9F32-C964-4795-844E-FB9145DCDE43, April 2022.

[Ha81] Habermas, Jürgen: Theorie des kommunikativen Handelns, volume 2. Suhrkamp Frankfurt, 1981.

[Je22] JetBrains: , Why IntelliJ IDEA. https://www.jetbrains.com/idea/, April 2022.

[JG22] JGraph: , Security-first diagramming for teams. https://app.diagrams.net/, April 2022.

[Li22] Lindley, Cody: , What Is JSX. https://www.reactenlightenment.com/react-jsx/5.1.html, April 2022.

[Lu22] Lucid: , Where seeing becomes doing. https://www.lucidchart.com/pages/, April 2022.

[Mi22] Microsoft: , SQL Server 2019. https://www.microsoft.com/nl-nl/sql-server/sql-server-2019, April 2022.

[MP21] Mulder, Mark.A.T.; Proper, Henderik.A.: On the Development of Enterprise-Grade Tool Support for the DEMO Method. In: CAiSE 2021. 2021.

[Mu22] Mulder, M.A.T.: Enabling the automatic verification and exchange DEMO models. PhD thesis, Radboud University Netherlands, March 2022.

[Na12] Nassar, Victor: Common criteria for usability review. Work, 41(Supplement 1):1053–1057, 2012.

[Nu22] Nulab: , Teams who diagram together, thrive together. `https://cacoo.com/`, April 2022.

[Or22] Oracle: , MySQL. `https://www.mysql.com/`, April 2022.

[Pa22] Parr, Terence: , What is ANTLR? `https://www.antlr.org/`, April 2022.

[Re96] Reijswoud, V. E. Van: The structure of business communication: Theory, model and application. PhD thesis, TU Delft, 1996.

[Re22a] React: , React - A JavaScript library for building user interfaces. `https://reactjs.org/`, April 2022.

[Re22b] React: , Virtual DOM and Internals – React. /urlhttps://reactjs.org/docs/faq-internals.html, April 2022.

[Sh16] Shin, Dong-Hee: Cross-Platform Users' Experiences Toward Designing Interusable Systems. International Journal of Human–Computer Interaction, 32(7):503–514, 2016.