FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Open-source DEMO Construction and Process Model Designer
Student:	Petr Ančinec
Supervisor:	Ing. Marek Skotnica
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2019/20

Instructions

Design and Engineering Methodology for Organizations (DEMO) is a methodology for analysing and modelling organisations and its processes. Several modelling tools that support this methodology exist. However, these tools are closed-source and do not support all DEMO aspect models according to their latest DEMO specification. The primary goal of this project is to deliver an open-source web-based modelling environment with a great user experience that can be used as a standard across the enterprise-engineering community, and in DEMO certification courses.

Steps to take:

- Review the latest specification for DEMO Construction and Process model notation.
- Design a DEMO model designer with excellent user experience.
- Create an MIT-licensed implementation of the proposed designer in Typescript.

All DEMO aspect models are connected and affect each other. Therefore it is essential that the designer maintains consistency between them.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague November 8, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Open-source DEMO Construction and Process Model Designer

Petr Ančinec

Department of Software Engineering
Supervisor: Ing. Marek Skotnica

May 12, 2019

Acknowledgements

I would like to thank my supervisor Ing. Marek Skotnica for his guidance, mentoring and tips. I would also like to thank everyone who participated in the modeling speed case study.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 12, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Ančinec, Petr. *Open-source DEMO Construction and Process Model Designer.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Práce se zabývá konstrukčním a procesním diagramem metodiky DEMO. Nedávné změny ve specifikaci DEMO zapříčinili, že neexistuje modelovací nástroj, který by tuto specifikaci podporoval a zároveň kontroloval její pravidla a omezení. Hlavním cílem této práce je vytvořit moderní, uživatelsky přívětivý webový nástroj pro vytváření modelů v DEMO 4 specifikaci a tento nástroj poskytnout s volně přístupným zdrojovým kódem. K dosažení tohoto cíle je potřeba pochopit DEMO specifikaci, protože všechny modely jsou mezi sebou propojené a mají různá omezení, které je potřeba dodržet. Vytvořený nástroj usnadňuje uživatelům práci s modelováním díky jeho přívětivosti a také kontroluje, aby byly dodrženy omezení dané specifikací DEMO. Případová studie vytvořená v rámci této práce ukázala, že nový nástroj je v průměru 2.3krát rychlejší než nástroj, který byl nejlépe hodnocen v rámci analýzy existujících nástrojů.

Klíčová slova DEMO, OCD, PSD, konstrukční model, procesní model, modelování, Angular, TypeScript

Abstract

This thesis focuses on Organization Construction Diagram (OCD) and Process Structure Diagram (PSD) of the Design and Engineering Methodology for Organizations (DEMO). Recent developments in DEMO have increased the need for a new modeling tool that supports the new DEMO 4 specification and verifies its rules. The primary goal of the thesis is to create a modern, user-friendly, open-source, web-based modeling tool that will support the latest DEMO specification. It is necessary to understand the DEMO specification if we want to achieve this goal because all four aspect models are connected and have specific constraints that need to be abided. Therefore, DEMO specification has to be reviewed and analyzed to deliver an efficient and user-friendly modeling tool. The new modeling tool helps users create DEMO models faster compared to currently existing solutions, and it also prevents users from breaking the specification rules. A case study created during this thesis has shown that the new modeling tool is on average 2.3 times faster when compared to the modeling tool that placed first in the review of existing modeling tools.

Keywords DEMO, OCD, PSD, construction model, process model, modeling, Angular, TypeScript

Contents

List of Figures

List of Tables

Introduction

Many organizations are growing in size and require a modern solution to document the structure of the organization and what happens within it. DEMO offers a very efficient and scientific method of documenting organizations and mapping social interactions between subjects.

The idea behind this thesis is to create a new modeling tool that will improve the modeling experience for students and the DEMO community. Currently, there are only a few modeling tools, and most of them do not support the newest DEMO 4 specification. All of these tools have some disadvantages compared to the others. The newly created tool should not have them, therefore making the modeling faster and more friendly to the user.

Recent developments in DEMO have increased the need for a modeling tool that supports the new DEMO specification. There are many requirements for the tool to make it user-friendly and useful. One of the main challenges is preventing the user from creating invalid models and therefore making the modeling faster and simpler.

The theoretical part of this thesis is focused on reviewing the newest DEMO 4 specification and comparing it to DEMO 3 so that an application following all DEMO 4 constraints and rules can be developed.

The practical part of this thesis is focused on creating a web-based, open-source DEMO modeling application that will help the DEMO community and the students partaking in any of the DEMO courses. The application should be released under the MIT license so that it can be further developed, modified and maintained by the DEMO community.

Chapter 1

Theoretical foundations

At the beginning of this chapter, DEMO is briefly explained. Afterward, the Organization Construction Diagram (OCD) and the Process Structure Diagram (PSD) are reviewed in both DEMO 3 and DEMO 4 specifications.

1.1 Models and diagrams in DEMO

According to *Advances in enterprise engineering VIII* [1], the purpose of DEMO is pointing out the essential parts of an organization and simplifying the organization structure in an ontological model, which consists of 4 aspect models:

The Construction Model (CM) specifies the construction of the organization system by the identified transaction kinds and the associated actor roles, as well as the information links between the actor roles and information banks.

The Process Model (PM) contains the specific transaction pattern of the transition kind for every type in the CM.

The Action Model (AM) specifies the imperatively formulated business rules that serve as guidelines for the actors in dealing with their agenda.

The State Model (SM) specifies the state space and the transition space of the production world with object class, fact types, result types, and ontological coexistence rules. [1]

In DEMO Specification Language (DEMO-SL) [2], Jan Dietz describes the representation of CM and PM:

> The CM of an organization is represented in an Organisation Construction Diagram (OCD), a Transaction Product Table (TPT), and a Bank Contents Table (BCT).

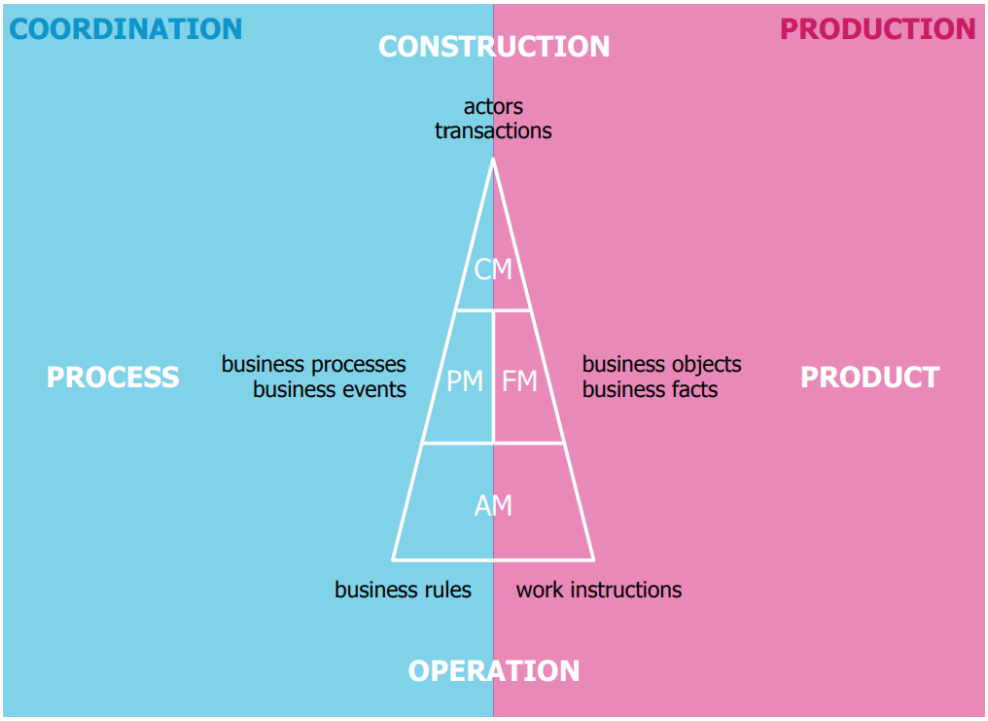

Figure 1.1: The aspect models of DEMO [3]

> A PM is represented in a Process Structure Diagram (PSD), optionally complemented by a Transaction Pattern Diagram (TPD) for one or more of the transaction kinds.

In the sections below, the OCD and the PSD will be reviewed in depth according to the DEMO specification version. The Global Logistics (GloLog) enterprise example from *The OMEGA Theory* [4] will be used to show the differences between DEMO versions and also when comparing other modeling tools. The GloLog enterprise consists of four business process kinds: the client order process, the supply order process, the sea transport process and the land transport process.

1.2 OCD and PSD in DEMO 3

The OCD in DEMO 3 contains a Scope Of Interest (SOI), actor roles, transaction kinds and different links that connect actor roles and transaction kinds.

The actor role can be elementary or composite. Actor roles who are outside of the scope of interest will always be composite because there is not enough information known about them or they are not crucial for the organization.

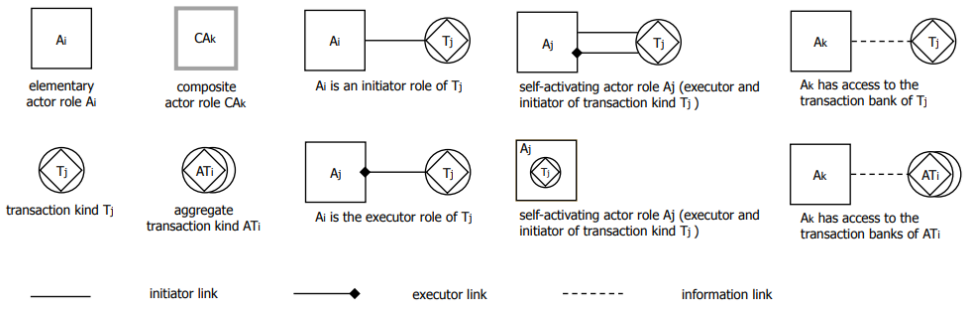

Figure 1.2: DEMO 3 OCD actor roles, transaction kinds and links [2]

In the diagram, actor roles are represented by a white rectangle for actor roles inside of the SOI and a gray rectangle for actor roles outside of the SOI. All actor roles are numbered and have a prefix (A- for elementary, CA- for composite) inside of the rectangle. [5]

The transaction kind can be original (red), informational (green) or documental (blue). To keep the OCD simple, in most cases, only the original transaction kinds are used. [3] A transaction kind that is aggregate consists of one or more transaction kinds and similarly to composite actor role we know they exist but do not want to know more about them. [6] Similar to actor roles, the transaction kinds are numbered and carry a prefix (T-).

There are three types of links that connect actor roles with transaction kinds to create transactions. Every actor role can be an initiator of many transactions and an executor of one or none. For this, the initiator and executor links are used. If the initiator and executor of the transaction are the same, we call the actor role self-activating. The last type of link is an information link. An information link between an actor role and a transaction kind means the actor role can access information of the transaction kind. [2]

The graphical interpretation of the OCD elements in DEMO 3 can be found in Figure 1.2.

The PSD is partially derived from the OCD. The PSD contains transaction kinds from the OCD with their width expanded to fit all the required information. There is also a non-proportional linear time axis from left to right, and the transaction kind is divided into three phases: the proposition phase (left from the diamond), the execution phase (inside the diamond), and the result phase (right from the diamond). There are two types of links in the PSD: the initiation link (starts from a C-fact symbol and ends in the request act) and the waiting link (starts from a C-fact symbol and ends in a C-act symbol). [2]

The graphical interpretation of the PSD elements in DEMO 3 can be found in Figure 1.4.

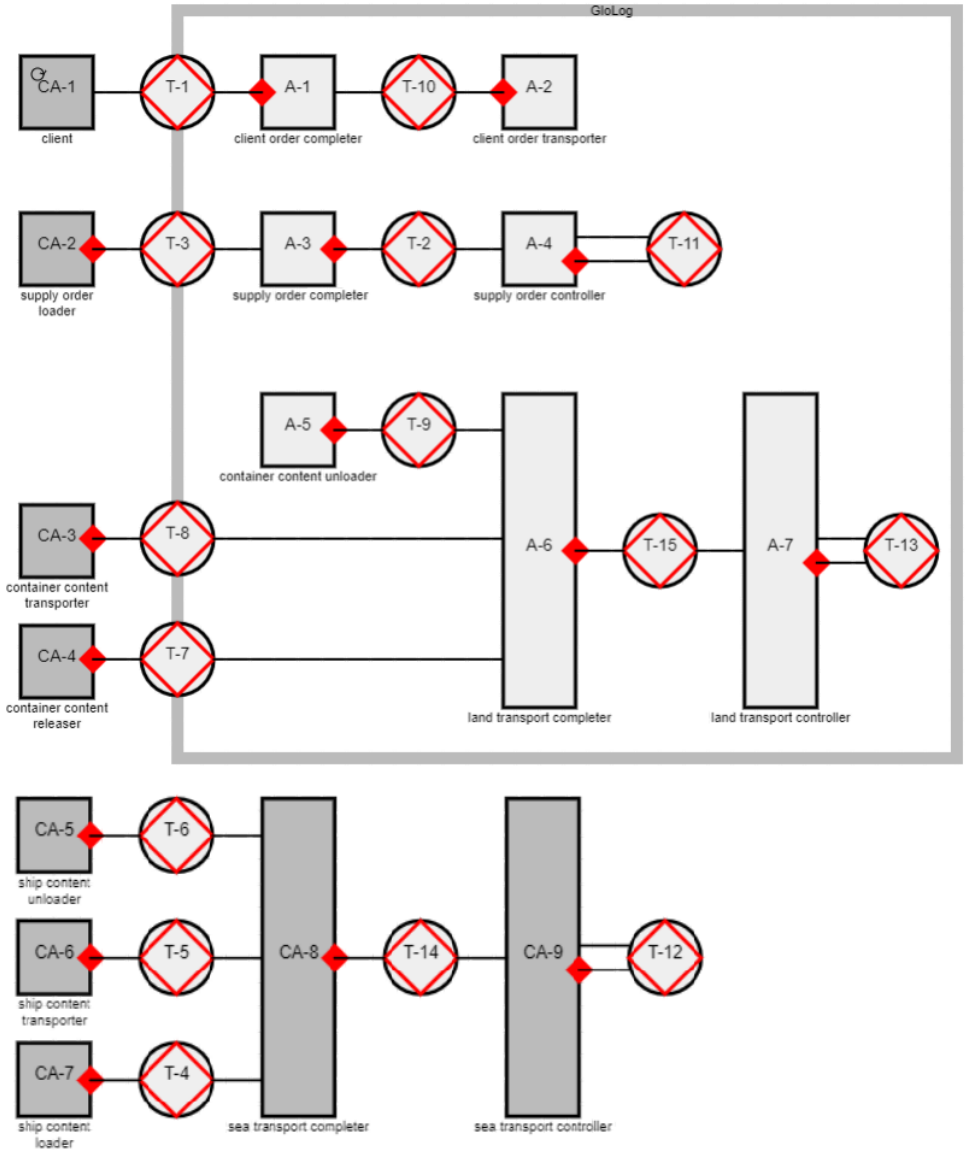

Figure 1.3: The OCD of GloLog enterprise according to DEMO 3

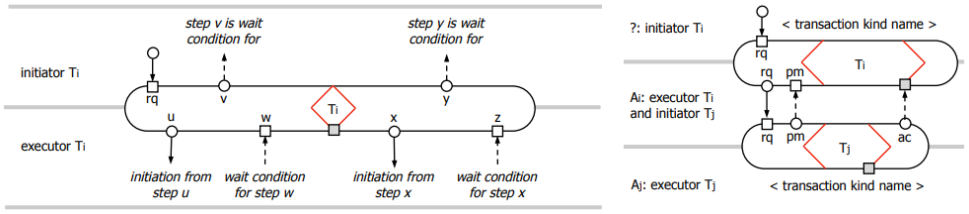

Figure 1.4: DEMO 3 PSD transaction kind [2]

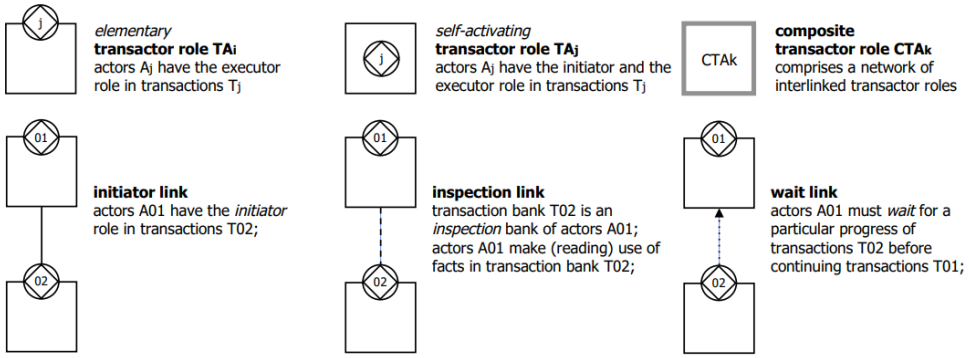

Figure 1.5: DEMO 4 OCD actor roles, transaction kinds and links [4]

1.3 OCD and PSD in DEMO 4

The OCD in DEMO 4 introduces a new element to simplify the diagram: the transactor role. Since every actor role is the executor of one transaction kind and every transaction kind has one executor actor role, they can be merged into a transactor role that represents both of them. Another simplification is the removal of prefixes. There is only a transactor role instead of an actor role and a transaction kind. Therefore, it can be numbered without a prefix. [4]

The SOI is no longer a grey rectangle that creates a border between actors. Instead, the SOI is only displayed using the color of the actor part of the transactor. The colors carry over from DEMO 3. The removal of the SOI border allows structuring the transactors into a tree and having a uniform look. [4]

Because of the transactor role, the executor link is no longer present as it is not needed. The inspection link replaces the information link. It goes from a transaction kind of a transactor role into an actor role of another transactor role, and it allows the actor role to read facts from the transaction kind. The last link of the OCD has been added in DEMO 4, and it is called a wait link.

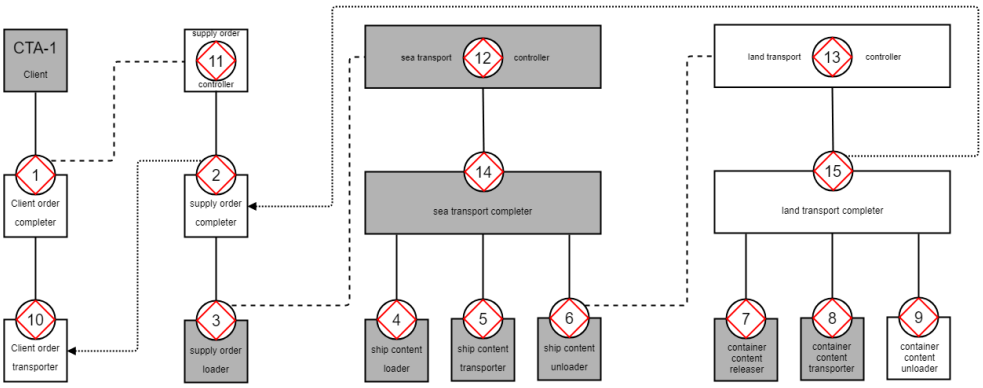

Figure 1.6: The OCD of GloLog enterprise according to DEMO 4

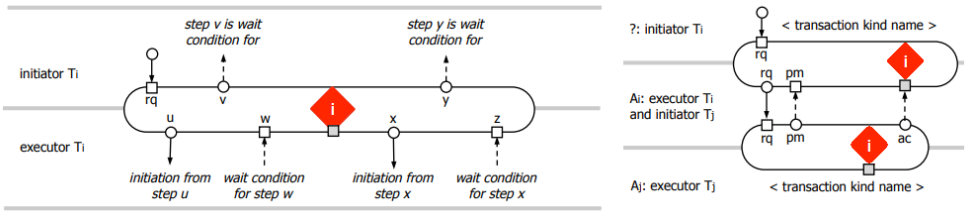

Figure 1.7: DEMO 4 PSD transaction kind [4]

Graphically, a wait link looks the same as an inspection link, except it ends with an arrow. If there is a wait link going from transaction kind into an actor, it means that the actor must wait for some progress of the target transaction before continuing. [4]

The graphical interpretation of the OCD elements in DEMO 4 can be found in Figure 1.5.

The PSD is still partially derived from the OCD. In DEMO 4, the interaction structure carries over from the OCD, which means the structure is the same as in DEMO 3 and the inspection links and the wait links do not carry over to the PSD. The transaction kinds do not have prefixes anymore, and the look has changed slightly. While the newly added inspection links and wait links provide useful insight, it might be too concise for studying the business processes in detail. [4]

The graphical interpretation of the PSD elements in DEMO 4 can be found in Figure 1.7.

Chapter 2

Review of existing modeling tools

Multiple modeling tools support DEMO and have different advantages and disadvantages. The purpose of analyzing other modeling tools is to use what was proven to be useful and to spot their weaknesses and try to improve upon them.

For uniform and straightforward analysis, the criteria will be the same for all tools. The criteria are:

Modeling capability - which aspect models are supported and what is their DEMO version.

Validation - preventing invalid models or detecting and displaying the cause.

Simulation - putting actors into actor roles and showing the actions step-by-step.

Modeling speed - a scale from 0 to 10 will be used to make the rating as fair as possible. The tools will be given points for:

- snapping edges to vertices (1),
- element creation (3),
- convenient element placement (1),
- quick change of element properties (3),
- fast repositioning of elements (2).

User experience - the same 0 to 10 scale will be used. The tools will be given points for:

- well-made UI (2),
- a simple way of saving and exporting projects (2),

- intuitive controls (3),
- responsivity (2),
- real-time collaboration support (1).

Availability - whether the tools are paid, free with restrictions or completely free.

Active development - if the tools are being worked on for new DEMO features.

In the sections below, every tool is described in depth following the criteria above. Table 2.1 gives a quick overview and comparison of all the tools.

2.1 Draw.io

Draw.io is a free online diagram software that supports community made palettes. [7] Currently, only the OCD and PSD palettes exist. [8] Since the palettes are just predefined styles of vertices and edges, Draw.io does not provide any model validation or simulation, and the modeling itself is slow because the edges have to be dragged into the canvas manually and it takes a lot of time and effort to make the connections look nice. All features including real-time collaboration and Google Drive synchronization are free of charge and along with a good looking UI provide an excellent user experience. Draw.io can export the model into HTML, SVG, PDF and image formats and save them into all sorts of online services and also into device storage.

In figure 2.1 is an OCD version 4 of the GloLog enterprise. Draw.io scored a total of 6 points in the modeling speed test and a total of 9 points in the user experience test.

Modeling speed score:

- edges snapping - 1
- element creation - 1
- element placement - 1
- properties change - 1
- element repositioning - 2

User experience score:

- UI - 2
- saving projects - 2
- intuitive controls - 2
- responsivity - 2
- collaboration - 1

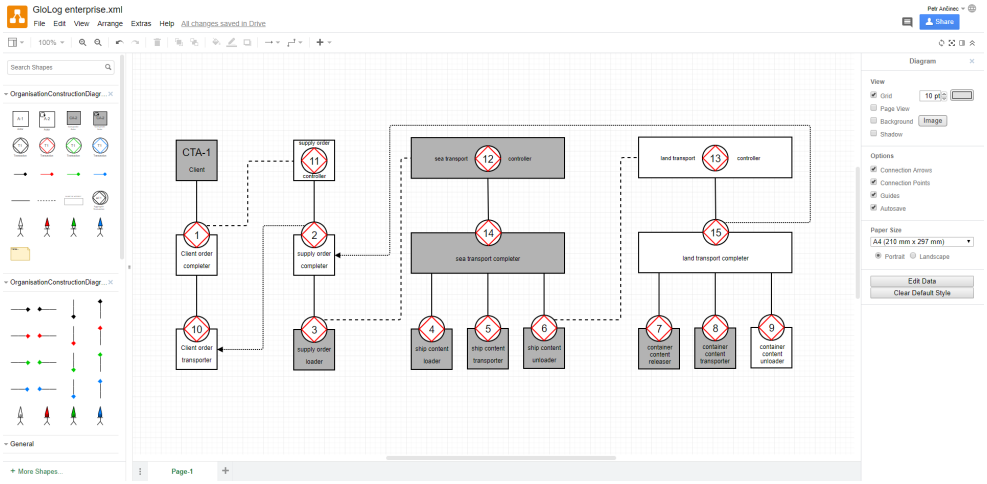

Figure 2.1: Draw.io modeler

2.2 DEMOWorld

DEMOWorld is an online modeling tool developed by ForMetis. It offers DEMO modeler with OCD, PSD and OFD support. The modeler has sufficient error validation and model simulation. [9] The modeling is relatively fast because the modeler attempts to create links between actors and transactions automatically and also snaps manually created links to predefined positions when a user is connecting them. DEMOWorld offers a trial version with a limit of three transactions per model as well as a free professor version with a limit of ten transactions. There is currently no way of increasing the limit above ten transactions. The UI does not look very modern, and the modeler itself has many pop-ups, but the navigation and modeling are pretty intuitive. DEMOWorld saves the models into its repository and allows export to a PDF file.

In figure 2.2 is an OCD version 3 of the GloLog enterprise. DEMOWorld scored a total of 7 points in the modeling speed test and a total of 5 points in the user experience test.

Modeling speed score:

- edges snapping - 1
- element creation - 2
- element placement - 1
- properties change - 2
- element repositioning - 1

User experience score:

- UI - 1
- saving projects - 1
- intuitive controls - 1
- responsivity - 2
- collaboration - 0

2.3 Modelworld

Modelworld is a free online modeling tool and repository for collaborative modeling. It supports OCD, PSD, and OFD creation and validation as well as model simulation. [10] While modeling in Modelworld, the tool automatically attempts to create links between actors and transaction kinds and it also allows snapping links to predefined positions. However, every time a user executes an operation, it takes the modeler roughly one second to process, which makes the modeling experience worse compared to DEMOWorld. On the other hand, the tool is entirely free, offers real-time collaboration and saving the models into its repository. It also allows importing and exporting the model using an XML as well as generating PDF and image formats.

In figure 2.3 is an OCD version 3 of the GloLog enterprise. Modelworld scored a total of 6 points in the modeling speed test and a total of 4 points in the user experience test.

Modeling speed score:

- edges snapping - 1
- element creation - 2
- element placement - 0
- properties change - 1
- element repositioning - 2

User experience score:

- UI - 0
- saving projects - 2
- intuitive controls - 1
- responsivity - 0
- collaboration - 1

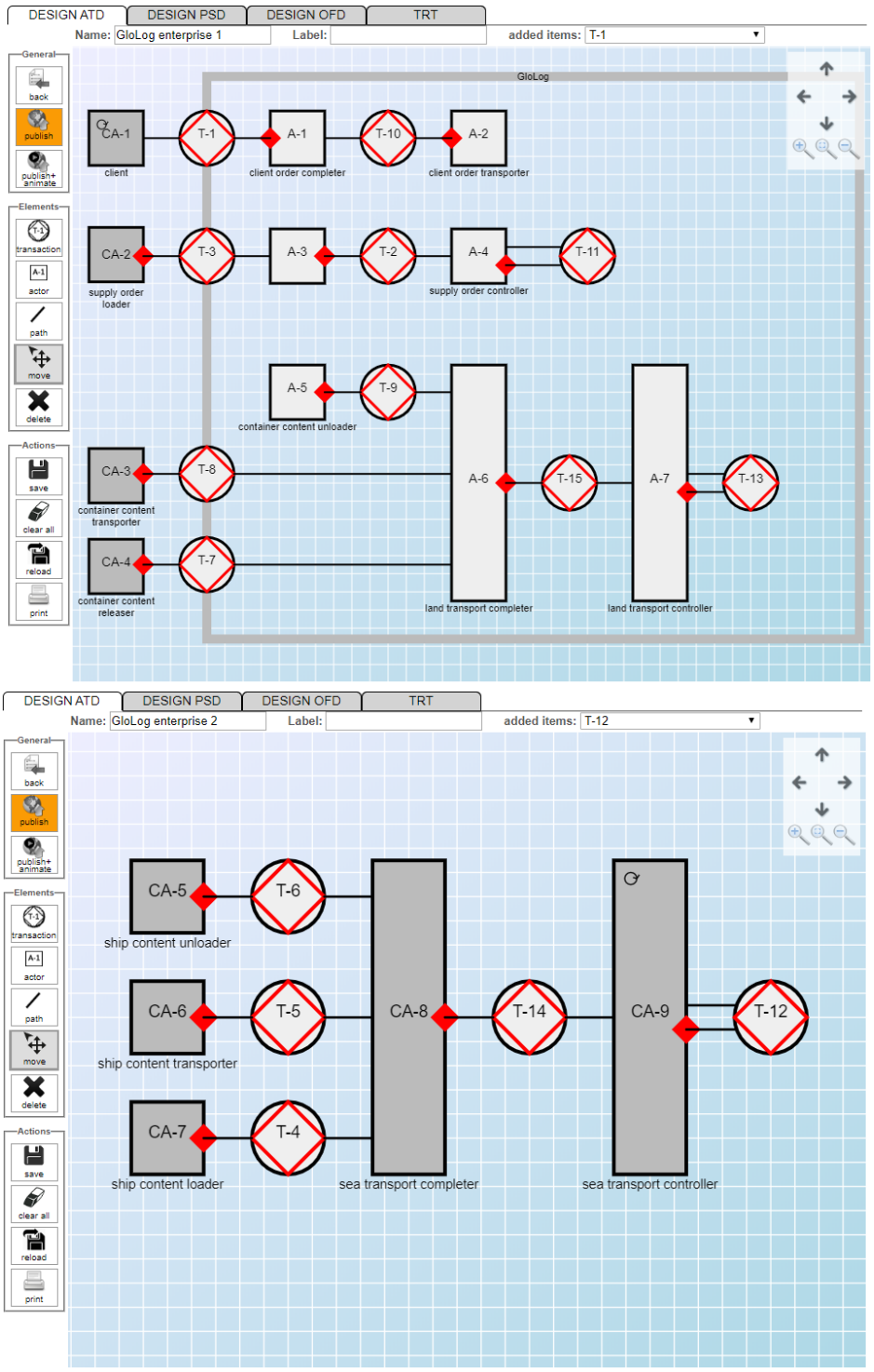

Figure 2.2: DEMOWorld modeler separated into two projects due to a transaction limit

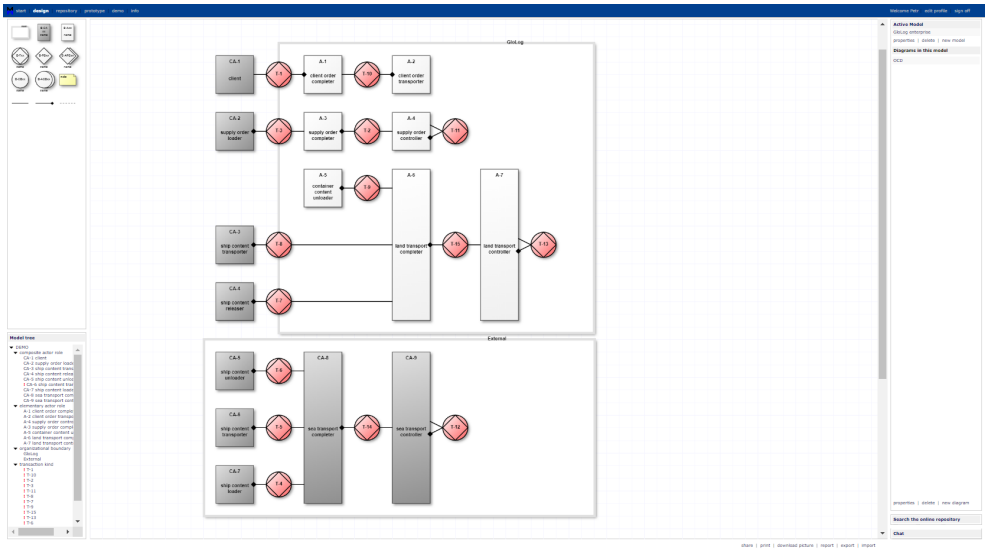

Figure 2.3: Modelworld modeler

2.4 Enterprise Architect - mDemosyne

Enterprise Architect (EA) is a client-based modeling tool developed by Sparx Systems commonly used by enterprises. [11] It supports all four models (OCD, PSD, OFD, AM) through a plugin called mDemosyne developed by Mphee. [12] The plugin also contains an external tool for full model validation that will list all errors in the model on demand. The model simulation is currently in development. The tool does not automatically create links between transactions and actors, but it does automatically snap the links to a grid. EA is a big and complex application which makes it more difficult to model in at first, and it has also reflected on the UI. The plugin itself is free of charge, but EA is a paid software. It allows export to XML, PDF and many other formats standardly supported by EA. A user can also import models from an XML file.

In figure 2.4 is an OCD version 3 of the GloLog enterprise. mDemosyne scored a total of 6 points in the modeling speed test and a total of 6 points in the user experience test.

Modeling speed score:

- edges snapping - 1
- element creation - 1
- element placement - 0
- properties change - 2

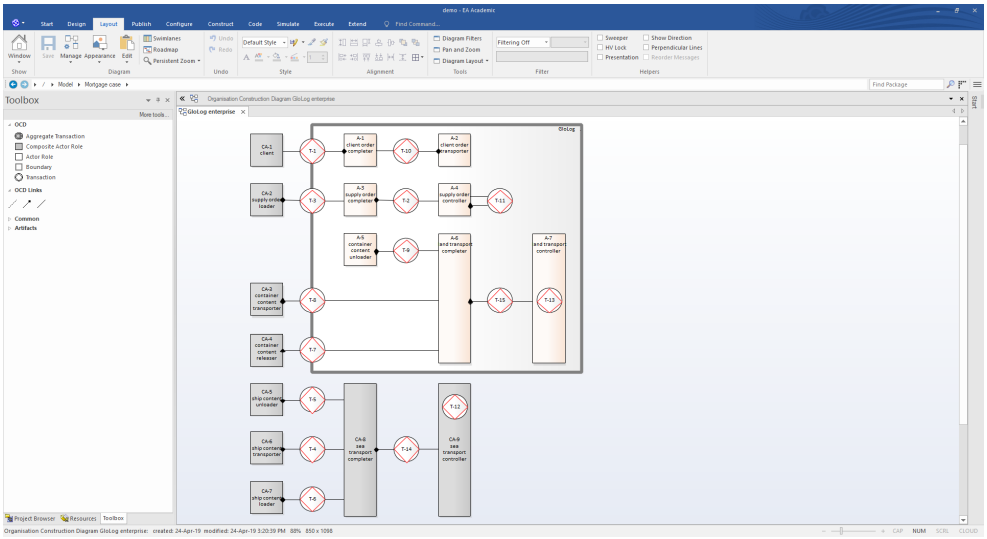

Figure 2.4: mDemosyne modeler

- element repositioning - 2

User experience score:

- UI - 1
- saving projects - 2
- intuitive controls - 1
- responsivity - 2
- collaboration - 0

2.5 UMLet

UMLet is a free, open-source client-based modeling tool written in Java. [13] It supports DEMO OCD, PSD, and OFD modeling through palettes. The application is a plain drawing tool and does not support any form of model validation or simulation. Modeling in UMLet is slow as there is no grid nor any feature that would make modeling easier and the user has to drag and adjust everything manually. The application is free along with the palettes. It offers export to PDF, SVG, and other image formats.

In figure 2.5 is an OCD version 3 of the GloLog enterprise. UMLet scored a total of 3 points in the modeling speed test and a total of 4 points in the user experience test.

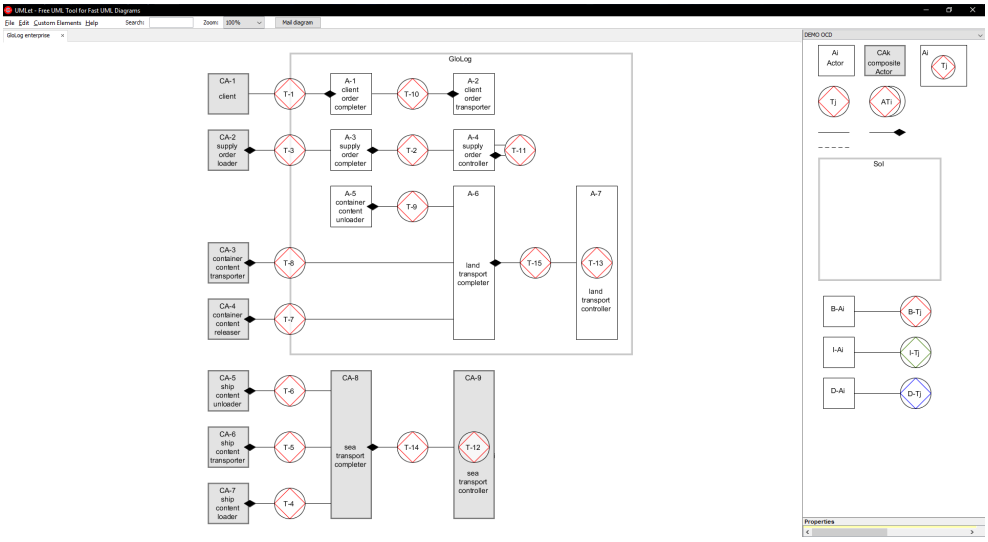

Figure 2.5: UMLet modeler

Modeling speed score:

- edges snapping - 1
- element creation - 0
- element placement - 0
- properties change - 0
- element repositioning - 2

User experience score:

- UI - 0
- saving projects - 2
- intuitive controls - 0
- responsivity - 2
- collaboration - 0

2.6 Summary

DEMOWorld scored the highest in the modeling speed test because the user can add new objects into the diagram and change their properties fast with few clicks. When designing the new modeling tool, the properties of transactors and actors should be configured using checkboxes and combo boxes rather than

Table 2.1: Modeling tools comparison

	OCD	PSD	OFD	AM	Validation	Simulation	Modeling speed	Availability	User experience	Active developement
Draw.io	v4[1]	v4	X	X	No	No	6	Free	9	No
DEMOWorld	v3	v3	v3	X	Yes	Yes	7	Restricted[2]	5	No
Modelworld	v3	v3	v3	X	Yes	Yes	6	Free	4	No
mDemosyne	v3	v3	v3	v3	Yes	Yes	6	Restricted[3]	6	Yes
UMLet	v3	v3	v3	3	No	No	3	Free	4	No

having multiple different transactor and actor objects that represent simple property change as seen in Draw.io.

On the other hand, Draw.io has by far the best user experience score. Draw.io achieved this mainly due to the intuitive controls of moving and resizing elements as well as dragging links and using overlays over an object to represent actions on the object. The new modeling tool should use an overlay to add new transactors and actors into the diagram and it should also use an overlay as a starting point for link dragging between transactors because it makes the controls intuitive to the user.

[1]The palettes are created for DEMO 3, but the elements can be grouped and used for DEMO 4 as well.

[2]The tool is restricted to 3 transactions for guests, 5 for students and 10 for professors.

[3]The plugin itself is free, but Enterprise Architect is paid (with a 30-day free trial).

Chapter 3

Architecture & design of a new modeling tool

This chapter provides a quick review of features that are or are not going to be implemented. Afterward, the chapter focuses on the data model, architecture and design of the application.

3.1 Features

3.1.1 In the thesis scope

- Implementing a web-based modeling tool under the MIT license that can run in a browser without any additional plugins.
- The modeling tool should support creating OCD according to DEMO-4 specification.
- The modeling tool should allow the user to capture the information required for PSD.
- The modeling tool should try to prevent the user from making errors in the model by not allowing the creation of invalid models.
- The modeling tool should have a property bar where a user can configure the element properties.

3.1.2 Out of the thesis scope

- The modeling tool will not have server integration.
- The modeling tool will not support creating OFD and AM.
- The modeling tool will not allow simulating or animating the created model.

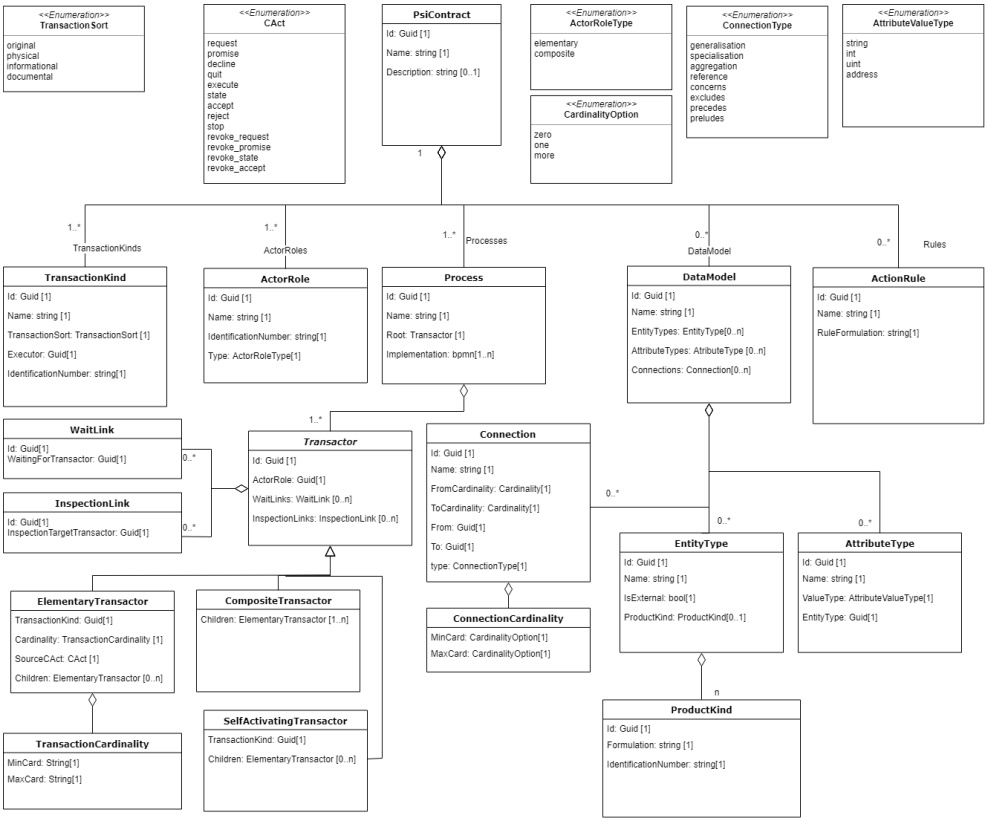

Figure 3.1: UML diagram of the *Contract meta model* [14]

3.2 Contract meta model

The Contract meta model is a model that keeps all the information about the OCD as well as information about other diagrams present in the application. It will allow saving, loading and undo/redo operations in the modeling tool. The graphical information is kept separate to make the loading easier and the model more clear. The usage of the model is explained in the Architecture section. The class diagram in Figure 3.1 shows the Contract meta model.

3.3 Architecture

The application has four main components - main screen, canvas, property bar, and model view. The modeling tool has to use the main screen component, the canvas component, and the property bar component. These components communicate with each other through two services – the state management service and the OCD communication service.

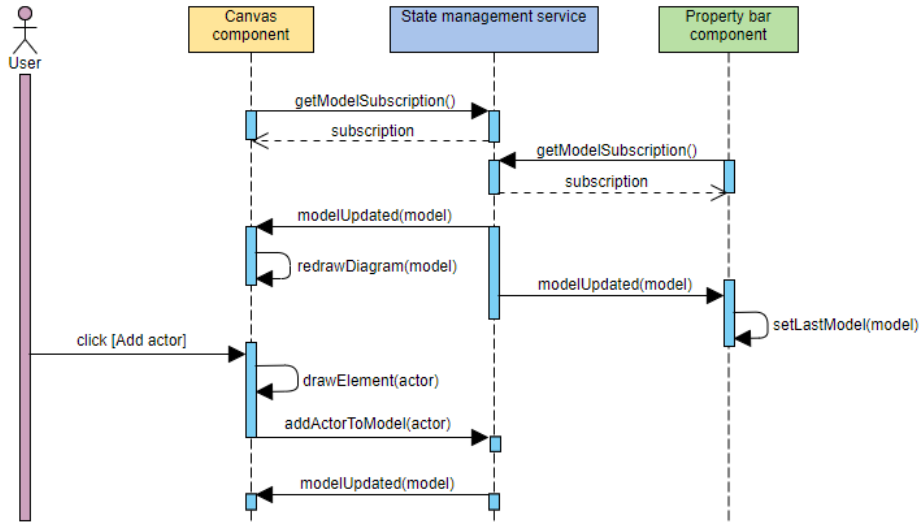

Figure 3.2: State management service

The state management service is responsible for the Contract meta model. The state management service will be treated as a black box because it is not part of this thesis. Every change that happens on the canvas is sent to the service using actions, and once the service processes these actions, it sends back the entire model asynchronously as an update event. Because the state management service supports synchronization between multiple devices and tabs similar to Google Docs, the canvas has to redraw the model when it receives this update so that it is always up to date with the model. The property bar is also subscribed to the update event, and it keeps track of the latest model so that it can pull actor roles and transaction kinds from it and suggest them to the user. The sequence UML diagram in Figure 3.2 describes the communication between the state management service and the components.

The second service is called OCD communication service, and it is used to send data between the property bar component, the canvas component and the main screen component. When a selection happens, the canvas component sends information required to update the property bar to the OCD communication service and the property bar updates based on an event created by this service. When a user changes something in the property bar component, the property bar component sends the change to the OCD communication service and the canvas receives an event and updates the selection according to the action. The main screen component sends an action to the canvas component when a user clicks on any button on the toolbar. The sequence UML diagram

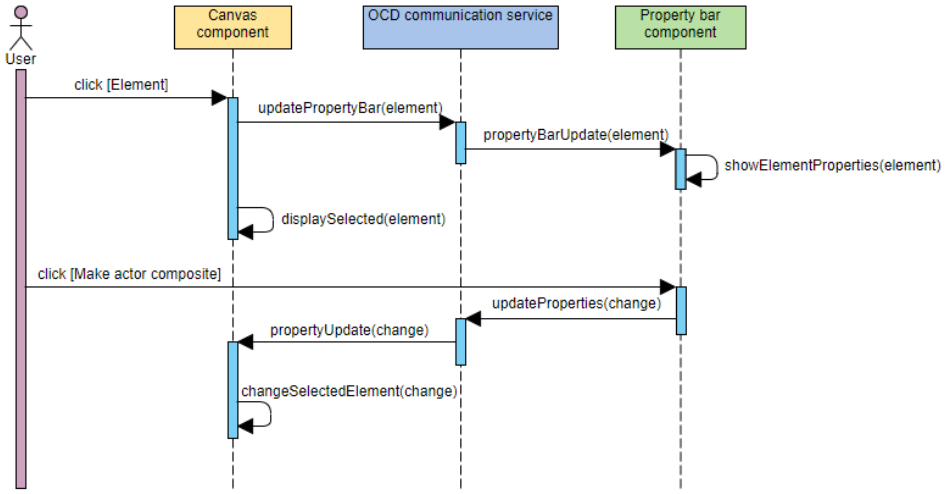

Figure 3.3: OCD communication service

in Figure 3.3 describes the communication between the OCD communication service and the components.

3.4 Design

From a design perspective, the functions of the components are:

Main screen component contains the model name, the menu buttons and the toolbar. The menu buttons allow the user to trigger

- exporting the diagram to PDF and SVG,
- saving and loading of the model,
- zooming in and out,
- undoing and redoing actions,
- removing objects and adding new processes.

Model view component visualizes the Contract meta model, and it is not a part of this thesis.

Canvas component is used for drawing and modeling the diagram.

Property bar component allows the user to change the size of the diagram and adjust properties of the currently selected element.

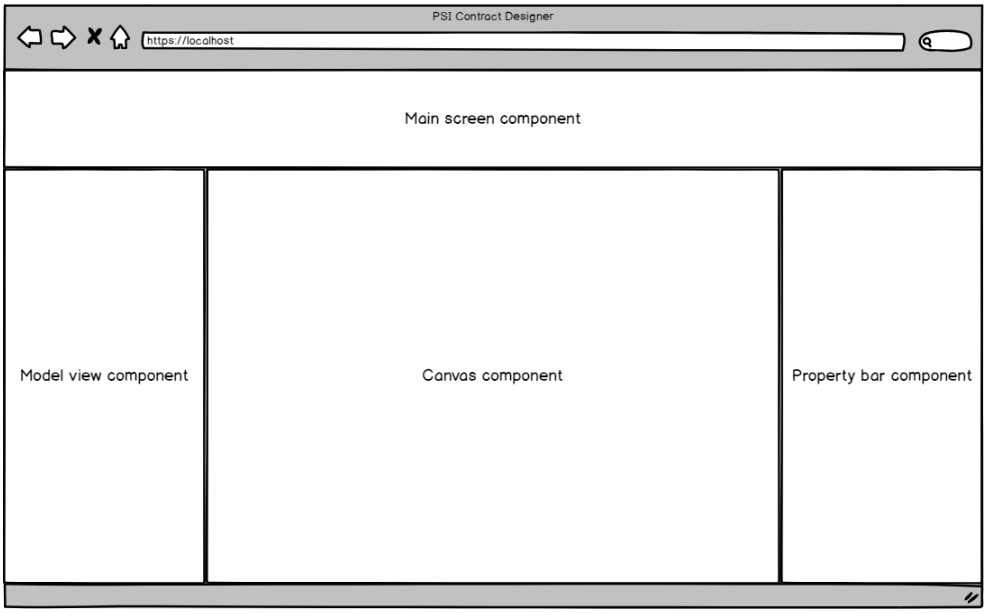

Figure 3.4: Layout of the components in the application [14]

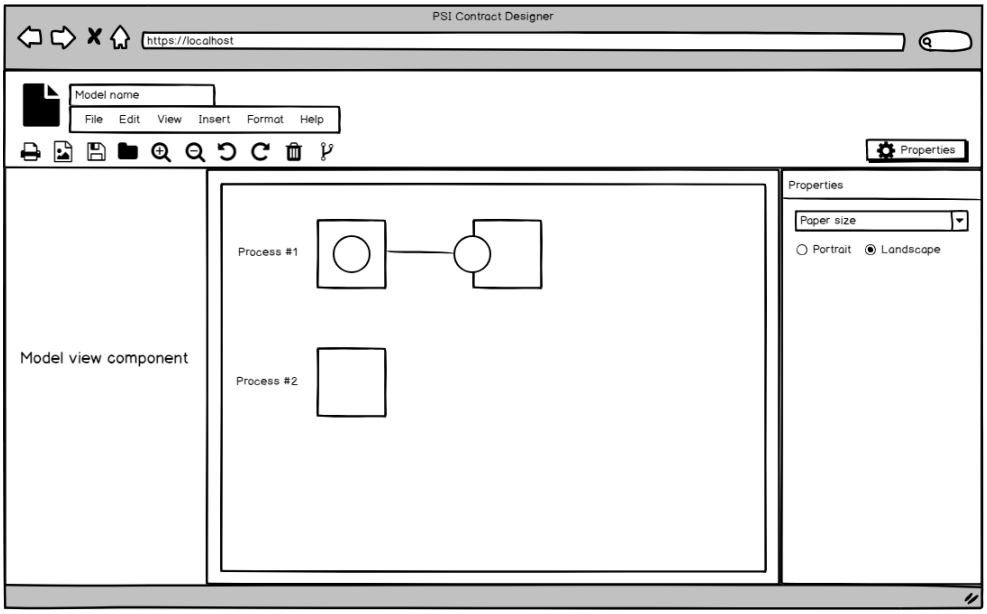

Figure 3.5: Wireframe of the application [14]

The layout of the components within the application can be seen in Figure 3.4. The components are placed where most of the users would expect them to be. The main screen component resides at the top. It is common for applications to have the menu buttons and the action buttons on the top and many users expect to find those controls there. That is why the controls are placed within the main screen component. The canvas component is in the middle of the screen because the diagram should be in the center of everything as it is an essential part. The property bar is on the right because applications like Draw.io and Google Docs, who are used by many users, place it on the right side of the screen and the user will most likely start looking from there. A wireframe of the application with some of the controls can be seen in Figure 3.5.

The property bar should be collapsible to allow extending the size of the canvas if needed. The property bar controls allow the user to modify the properties of the selected object as well as the paper size and orientation. The controls show based on the selected element:

Process label allows changing the paper size and orientation because there is nothing to configure. Same goes for empty selection and selection of multiple elements because a user can select multiple elements of a different kind and the options would not be intuitive.

Actor role allows changing all current actor roles and lets the user make the actor role composite. If an actor role is changed to one that already exists in the diagram, they become linked, and all edits made to any one of the linked actor roles modify all linked actor roles.

Transactor role allows changing everything the actor role does since a transactor role consists of an actor role and a transaction kind. It also allows changing all current transaction kinds based on the same principle as the actor role linking, except it links together different transaction kinds. It also lets the user change the transaction sort, which specifies whether the transaction kind is original, informational or documental.

Initiator link allows changing the cardinality of the target transactor. If the source of the link is a transactor, it also lets the user change the coordination act of the source transactor.

Inspection link allows changing the source and target position of the link. The link can start at the top or the bottom of the transaction kind, and it can end at the top or the bottom of the actor role. The inspection link can also be changed into a wait link, which gives the user additional options.

Wait link has all the controls as inspection link does. Additionally, it also allows changing coordination act of the source and target transactor.

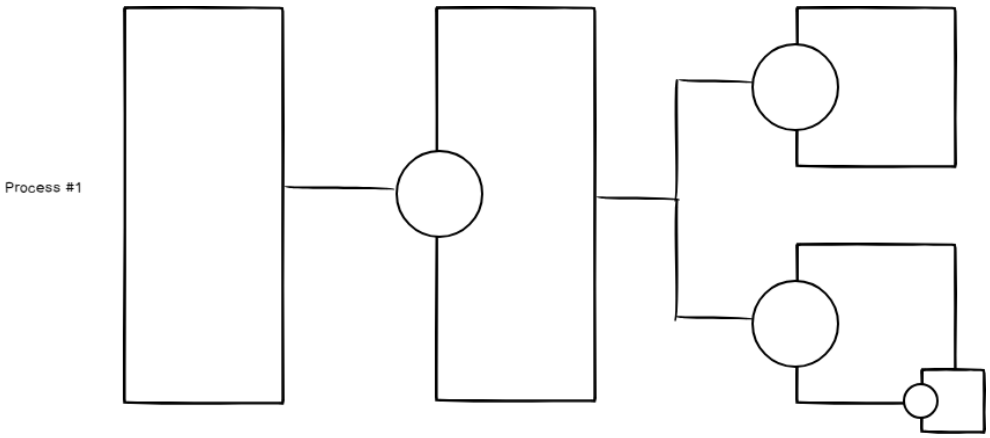

Figure 3.6: Process name, expanded actor, expanded transactor, regular transactor, regular transactor with overlay (left to right) [14]

The canvas contains a diagram that draws elements. The elements can be processes, actors, transactors, and links. The user can move the diagram around, zoom in/out, select elements and create new ones or modify the current ones. The elements cannot be moved because they have a fixed position and are in a hierarchical tree structure starting from the left to the right. As shown in Figure 3.6, the actors and the transactors are resized based on the height of all the child transactors they have so that the parent transactor is the same height as the entire subtree. To start modeling, the user can add a process into the diagram from the main screen component using the process button. The process represents one OCD tree and always starts with either actor or self-activating transactor. Once the process is added to the diagram, the user can hover over it to toggle the elements overlay. The user can build the OCD tree by clicking on the overlay icons. An example of the transactor overlay can be seen in Figure 3.6. Every element has a specific overlay based on the element type:

The process has two icons in the overlay that allow the user to add an actor and a self-activating transactor. If the process already has a child, the overlay no longer shows because every process has only one child transactor or actor.

The actor has an icon that allows adding an elementary transactor role and linking it with the actor using an initiator link.

The transactors have everything the actor has and additionally also two arrows that allow the user to start dragging a wait link to another transactor from the top or the bottom.

The links do not have any overlay because if they exist, they already link two elements, and there is no reason to create anything from them.

Since the elements have a fixed position in the tree structure, the user has to be able to reposition the elements within the tree. To achieve this, the user can drag an element over another element to swap them, drag an element near another element to append to that element or drag an element in between two elements to place the element in between. Only elements of the same type can be swapped, and only transactors can be appended to either actor or another transactor to keep the model valid.

Chapter 4

Proof of concept implementation

This chapter describes the choices of technologies used for the implementation, some interesting problems that happened during the development, how the modeling tool integrates with the application and how it is tested. At the end of the chapter, the modeling capabilities are shown, and a small case study is made to measure and compare how fast the new modeling tool is.

4.1 Used technologies

4.1.1 mxGraph

mxGraph is a JavaScript library that allows creating an interactive graph and charting applications that run natively in any major browser. [15] The main advantage of mxGraph is that it has been continuously developed and used in many large enterprises commercially from 2005 until 2016, which makes the library rich in features and well-tested with very few bugs. [16] The mxGraph library is also used for the Draw.io diagram software, which means it will be good enough for the implementation part of this thesis.

4.1.2 TypeScript

"*TypeScript is a language for application-scale JavaScript. TypeScript adds optional types to JavaScript that support tools for large-scale JavaScript applications for any browser, for any host, on any OS. TypeScript compiles to readable, standards-based JavaScript.*" [17]

Since TypeScript compiles to JavaScript, it can be used with the mxGraph library. TypeScript will allow creating more readable and higher quality object-oriented code.

```
<svg width="130" height="100" xmlns="http://www.w3.org/2000/svg">
  <rect fill="{rectangleFill}" x="32" y="0" width="97.5" height="100%"/>
  <circle fill="#ffffff" stroke="#000000" cx="32" cy="50%" r="30"/>
  <rect transform="rotate(45, 31.5, 50)" x="11.5" y="30" width="40"
        height="40" stroke="{diamondStroke}" fill="{diamondFill}"/>
  <text y="57" x="32" xml:space="preserve" text-anchor="middle"
        font-size="24" fill="#ffffff">{transactionNumber}</text>
</svg>
```

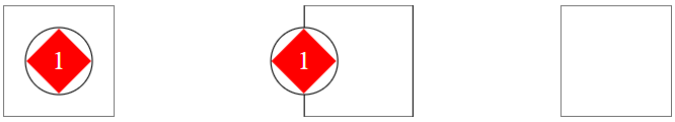

Figure 4.1: SVG template for elementary transactor with the graphical representation of transactor and actor elements

4.1.3 Angular

"*Angular is a platform and framework for building client applications in HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.*" [18]

Angular will be used because it is a modern framework that excels in single-page applications [19] and the component system will help dividing the application into separate modules. Because the implementation part of this thesis is a piece of a bigger project that uses Angular, it will be useful when it comes to integrating the implementation into the project.

4.1.4 Material Design

"*Material is an adaptable system of guidelines, components, and tools that support the best practices of user interface design. Backed by open-source code, Material streamlines collaboration between designers and developers, and helps teams quickly build beautiful products.*" [20]

Material Design will be used to create nice looking and responsive UI. All controls in the property bar and the toolbar will be using Material Design for a unified look that should make the application user-friendly and intuitive for new users.

4.2 Development process

The implementation part of this thesis is done according to the design and architecture described in Chapter 3. However, there were some problems during the development process that are worth mentioning.

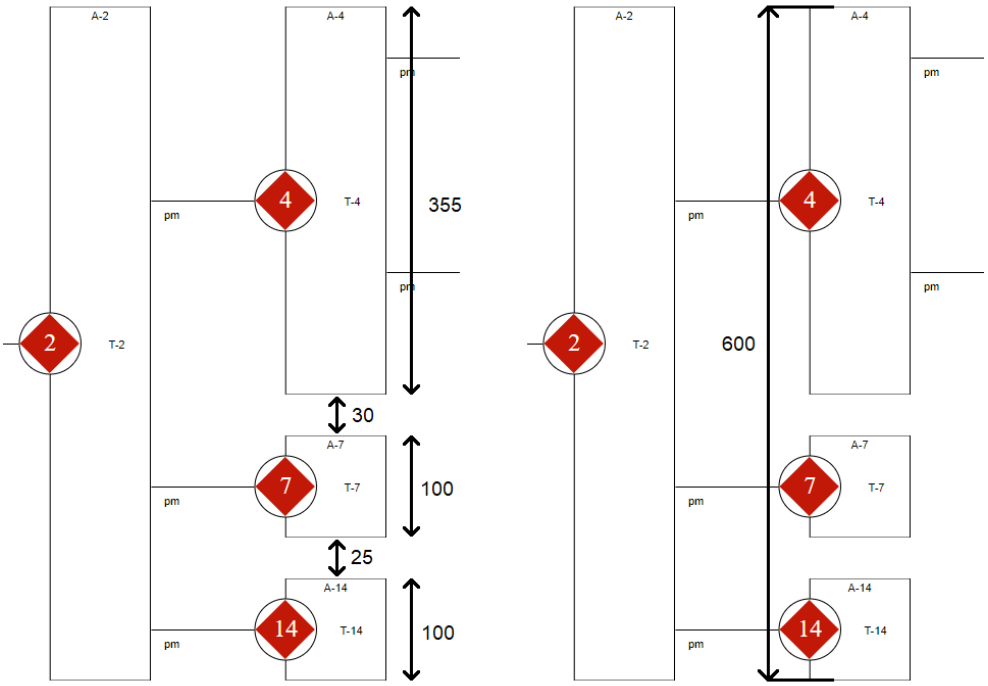

Figure 4.2: Differences in height using height adding method and coordinate difference method

The property bar implementation is straightforward. Thanks to Angulars Material Design form controls and two-way data binding, only the communication between the canvas component and the property bar component has to be implemented. Whenever there is a change in any of the control forms, all the control forms current values are sent to the canvas component, where it is used based on what element is currently selected. When a selection of element changes, the canvas component sends required data about the selected element to the property bar component and the control forms are shown and filled with values based on that element.

The biggest challenge was implementing the canvas component since the mxGraph library used for drawing the diagrams has an incredible amount of features, and learning how the library works takes a long time. SVG images are used to represent all the elements in the diagram to make the graph scalable and the properties changeable through code. To make mxGraph use the SVG images, they have to be taken as a string, the color and the text properties of the image have to be replaced based on the element properties, and the final string is prepended with "*data:image/svg+xml,*" to specify that the image is in the SVG format. In Figure 4.1 there is an SVG template for elementary

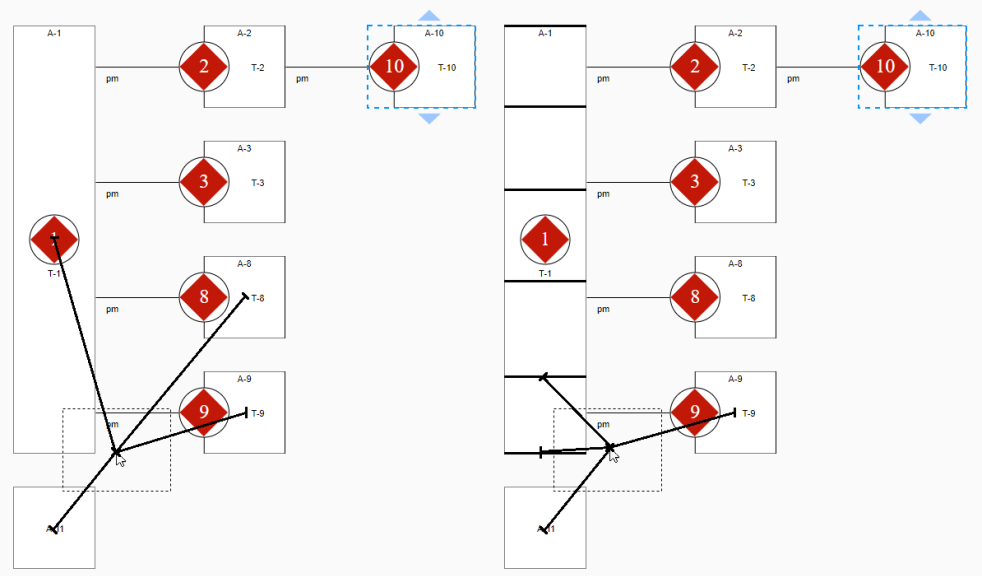

Figure 4.3: Differences in closest element calculation methods

transactor along with some graphical representations of the elements. Some of the height related information about the SVG image is in percentage units so that it automatically resizes when the element height changes.

The height change based on children elements was one of the more difficult tasks. Whenever an element is added, moved or removed, the algorithm has to resize all elements on the path to the root of the OCD tree. The first approach when resizing an element to be the same height as all of the children was to iterate the children and add up all of their heights plus a fixed gap between them. This approach caused problems because the gaps were not always the same size based on the placement of nearby elements, which would then start accumulating the height error with every level of the tree. The second approach was to take the y-coordinate of the uppermost element and the height plus y-coordinate of the bottommost element and subtract them. This approach was tested to give accurate results. Figure 4.2 shows the different results of the two methods with the correct height being 600. Note that transactor 4 (on the left) already accumulated an error of 5 from its children. The difference is only 10, and it is not noticeable, but for big models, the error accumulates, and the edge starting points get misaligned.

Another problem worth mentioning was element swapping and appending. When a user holds left click on an element, he can drag it around and trigger swap and append actions. When the dragging is going on, a distance to the closest element is needed to show the appropriate append actions (user can append between two elements on the same level or append to an element when

to the right of it). The first approach was to take the center of every element placed in the diagram and calculate the distance to the mouse coordinate. This method works well when all elements are approximately the same height. However, when an elements height increases, the center of the element can be further than another element that has a smaller height. The left part of Figure 4.3 shows the problem with calculating the closest element as the distance from the center. The self-activating transactor should be chosen as the closest element because the mouse is next to its border. However, the actor element is chosen instead, and the self-activating transactor is the fourth closest when it should be the first. The solution was to sample elements whose height is bigger than the default height so that this problem does not occur. As seen in Figure 4.3 on the right, the self-activating transactor is sampled every 100 pixels and additionally at the bottom of the element because it is common to append in between two elements. This way, the closest element is chosen more accurately without sacrificing many additional resources.

4.3 Integration process

The new modeling tool is integrated into a bigger project which consists of multiple modeling tools. All of the modeling tools modify the Contract meta model. The state management of the Contract meta model is part of a different thesis and will be treated as a black box. Every change to the Contract meta model has to be sent to the state management service, and every time the state management service sends an update to the modeling tool, the modeling tool has to redraw the entire model. In return, the state management implements undo/redo operations and cross-browser synchronization, which allows collaborative modeling.

When the new modeling tool receives an update from the state management service, the update carries the entire Contract meta model and some graphical information such as positions of transactors in the OCD tree and coordinates of line bends for inspection links and wait links. The graphical information is minimal because all elements have set positions within the diagram. When updating the diagram, it has to be cleared completely, and afterward, the new Contract meta model is parsed and drawn into it.

4.4 Testing

Since the project uses the Angular framework, every component has its file for unit tests. The mxGraph library has been in production for over ten years and should not contain many reproducible bugs. A lot of the features in the implementation part are difficult to create tests for because they usually require user input such as mouse movement and clicking. The created tests cover the component initialization such as initializing the graph with all required

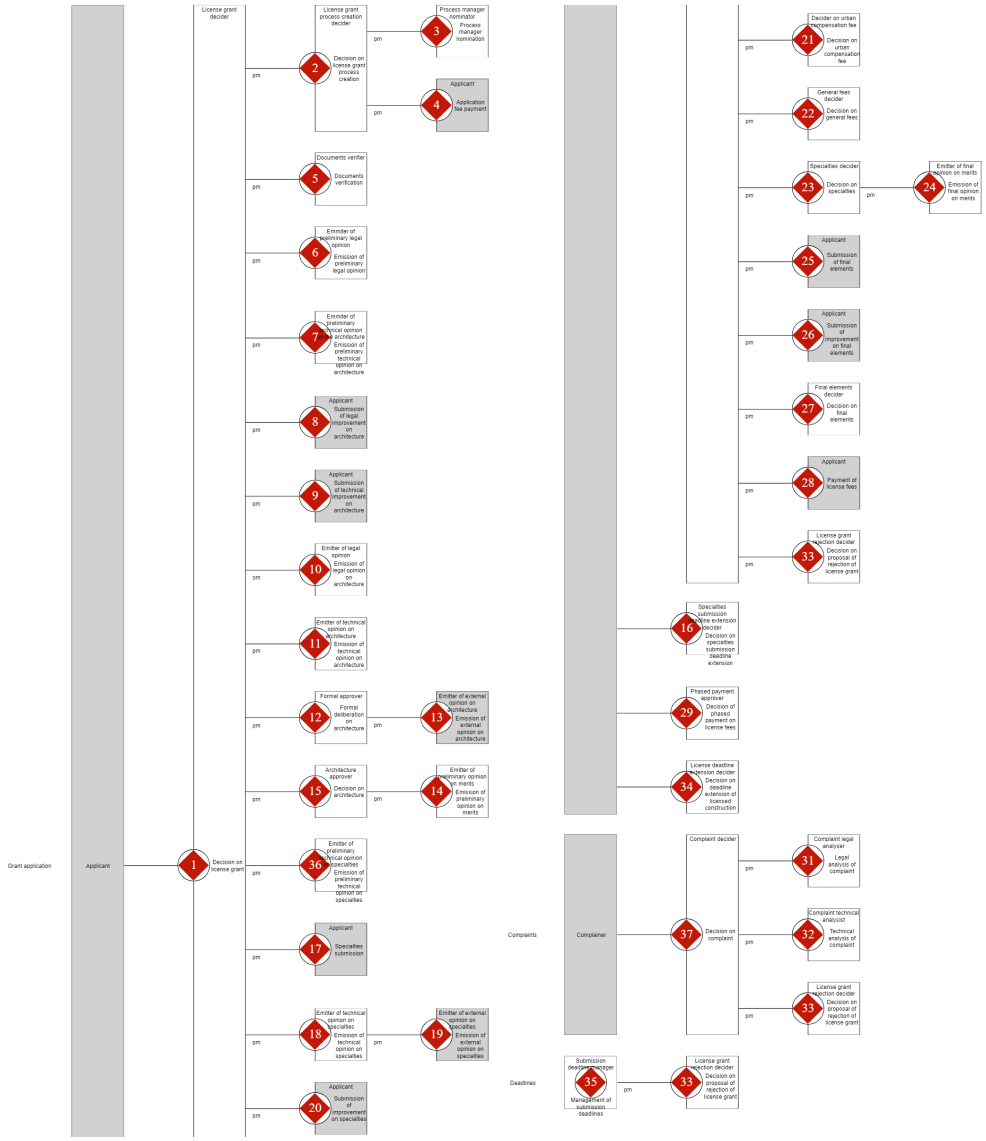

Figure 4.4: OCD of the *Construction licensing* [21] separated into two parts

properties, adding elements and removing elements. After that, the tests try reproducing simple communication between the canvas component and the property bar component using the OCD communication service.

Table 4.1: Time taken (minutes and seconds) to create the OCD of GloLog enterprise

	New modeling tool	Draw.io
Subject 1	5:09	11:54
Subject 2	3:17	13:26
Subject 3	8:14	17:41
Subject 4	10:00	20:00
Subject 5	6:35	13:50
Average	6:39	15:22

4.5 Modeling capabilities

The OCD of *Construction licensing* [21] that can be seen in Figure 4.4 has been created in the new modeling tool to demonstrate the modeling capability and handling of complex models. The new modeling tool fully covers the OCD for the DEMO 4 specification, and because of the coordination acts on initiator links and wait links, it also covers the majority of the PSD in a single diagram.

4.6 Modeling speed

To measure how much faster a user can create models compared to existing modeling tools, a small case study has been made. This case study compares the new modeling tool to the Draw.io modeler. The Draw.io modeler was chosen because it was one of the highest rated modeling tools in the review of existing modeling tools. While DEMOWorld was rated the fastest, it only allows ten transactions per project, which is not enough to measure the modeling speed accurately. Several users across the DEMO community were asked to create the OCD of GloLog enterprise in both modeling tools. Table 4.1 shows all participants along with the time taken to create the OCD in each tool. The study has shown that the average time is 6:39 for the new modeling tool and 15:22 for Draw.io. On average, the user should spend 2.3 times less time in the new modeling tool compared to Draw.io.

Conclusion

The goal of this thesis was to compare the existing modeling tools for DEMO and based on that design and implement a prototype of a new user-friendly modeling tool supporting the DEMO 4 specification.

The review of the DEMO 4 specification in the chapter Theoretical foundations revealed how the model is created and which constraints need to be enforced. The review of the existing modeling tools revealed features that allow fast modeling and good user experience.

The implementation is available at [22] with the full source code released under the MIT license at [23]. The solution uses Material Design which makes the application look nice. In the chapter Proof of concept implementation, a small case study measured the new modeling tool to be 2.3 times faster in modeling OCD when compared to the best scoring application in the chapter Review of existing modeling tools. The design of the application along with the intuitive controls and automatic element creation and connection make the application easy to use and user-friendly.

In the future, the PSD could be extended, and a feature allowing simulation of the model could be added into the project.

Bibliography

1. AVEIRO, David; TRIBOLET, José; GOUVEIA, Duarte. *Advances in enterprise engineering VIII*. 1st edition. New York: Springer, 2014. ISBN 978-3-319-06504-5. Available from DOI: 10.1007/978-3-319-06505-2.
2. DIETZ, Jan. *DEMOSL-3 DEMO Specification Language* [online]. 2017 [visited on 2019-04-19]. Available from: http://www.ee-institute.org/en/documents/21/methodology-documents. Technical report. Enterprise Engineering Institute.
3. DIETZ, Jan. *The Essence of Organisation: An Introduction to Enterprise Engineering*. 2nd edition. Sapio Enterprise Engineering, 2015. ISBN 978-9-081-54494-8. Available also from: https://books.google.com/books?id=XtyEAQAACAAJ.
4. DIETZ, Jan. *The OMEGA theory - understanding the construction of organisations* [online]. 2017 [visited on 2019-04-17]. Available from DOI: 10.13140/RG.2.2.22806.24642. Technical report.
5. HUŇKA, František. *Podnikové ontologie* [online]. Ostrava, 2012 [visited on 2019-04-14]. Available from: http://www1.osu.cz/~hunka/vyuka/javaOOP/XXPONT.pdf. Technical report. Ostravská univerzita v Ostravě.
6. DIETZ, Jan. *Enterprise Ontology: Theory and Methodology*. Springer Berlin Heidelberg, 2010. ISBN 9783642067150. Available also from: https://books.google.com/books?id=ZXuccQAACAAJ.
7. JGRAPH. *draw.io* [online]. 2018 [visited on 2019-04-20]. Available from: https://www.draw.io/.
8. DUNAEVSKIY, Sergey. *Design & Engineering Methodology for Organisations (DEMO) palettes for draw.io* [online]. 2018 [visited on 2019-04-20]. Available from: https://github.com/dunaevskiy/DEMO-drawio-palette.
9. FORMETIS. *About DEMOworld* [online]. 2019 [visited on 2019-04-20]. Available from: https://www.demoworld.nl/Portal/Home/About.

10. HOMMES, Bart-Jan. *Modelworld* [online]. 2011 [visited on 2019-04-20]. Available from: `http://www.modelworld.nl/index.php?page=documentation`.

11. SPARX SYSTEMS. *Enterprise Architect* [online]. 2019 [visited on 2019-04-20]. Available from: `https://sparxsystems.com/products/ea/index.html`.

12. MPHEE. *mDemosyne* [online]. 2019 [visited on 2019-04-20]. Available from: `http://www.mphee.nl/demo-tool/`.

13. THE UMLET TEAM. *UMLet* [online]. 2015 [visited on 2019-04-20]. Available from: `https://www.umlet.com/`.

14. SKOTNICA, Marek. *Lecture notes in Software Team Project: Functional Specification - PSI Contract Designer*. 2019.

15. JGRAPH. *mxGraph* [online]. 2019 [visited on 2019-04-24]. Available from: `https://jgraph.github.io/mxgraph/`.

16. JGRAPH. *mxGraph - NPM* [online]. 2019 [visited on 2019-04-24]. Available from: `https://www.npmjs.com/package/mxgraph`.

17. MICROSOFT. *TypeScript* [online]. 2019 [visited on 2019-04-24]. Available from: `https://github.com/Microsoft/TypeScript`.

18. GOOGLE. *Angular - Architecture overview* [online]. 2019 [visited on 2019-04-24]. Available from: `https://angular.io/guide/architecture`.

19. FREEMAN, Adam. *Pro angular 6*. 3rd edition. New York, NY: Springer Science+Business Media, 2018. ISBN 978-1-4842-3648-2.

20. GOOGLE. *Material Design* [online]. 2019 [visited on 2019-04-24]. Available from: `https://material.io/design/`.

21. AVEIRO, David; PINTO, Duarte. Devising DEMO Guidelines and Process Patterns and Validating Comprehensiveness and Conciseness. In: *Enterprise, Business-Process and Information Systems Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 408–423. ISBN 978-3-662-43745-2.

22. SKOTNICA, Marek. *Enterprise Designer* [online]. 2019 [visited on 2019-05-08]. Available from: `https://smartcontracts.azurewebsites.net`.

23. SKOTNICA, Marek. *Enterprise Designer Repository* [online]. 2019 [visited on 2019-05-08]. Available from: `https://dev.azure.com/CCMiResearch/_git/VSContract`.

Appendix A

Acronyms

AM Action Model

BCT Bank Contents Table

CM Construction Model

DEMO Design & Engineering Methodology for Organizations

EA Enterprise Architect

GloLog Global Logistics

OCD Organization Construction Diagram

OFD Object Fact Diagram

PDF Portable Document Format

PM Process Model

PSD Process Structure Diagram

SM State Model

SOI Scope Of Interest

SVG Scalable Vector Graphics

TPD Transaction Pattern Diagram

TPT Transaction Product Table

UI User Interface

UML Unified Modeling Language

XML eXtensible Markup Language

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `application` ..... implementation sources
    - `readme.txt` . list of files from application that belong to this thesis
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format