TU WIEN Informatics

Survey of current (web) modeling tool development platforms

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Business Informatics

by

Tobias Gacko
Registration Number 11730105

to the Faculty of Informatics

at the TU Wien

Advisor: Assistant Prof. Dipl.-Wirtsch.Inf.Univ. Dr.rer.pol. Dominik Bork

Vienna, 22nd August, 2023

Tobias Gacko	Dominik Bork

Erklärung zur Verfassung der Arbeit

Tobias Gacko

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 22. August 2023

Tobias Gacko

Acknowledgements

I would like to first and foremost express my sincere thanks my advisor Dominik Bork, who provided me with the opportunity to write my first bachelor thesis and the chance to contribute to the field of model engineering. I would also like to express my gratitude for the support and encouragement during the process of writing this paper, as well as the involvements in other projects which have provided me with great memories.

I want to thank Syed Juned Ali for providing support during the deployment of the website, as well as all the colleagues of the Business Informatics group who lended their support.

Abstract

Modeling Tools aim to provide their users with a set of graphical symbols so that a certain concept (e.g. a process, a model or a system) can be abstracted to its core components, providing a comprehensive overview of all the necessary information on first glance. The resulting models generally take the form of a diagram. As of now there is a large variety of modeling tools, with a new trend emerging in the area of web-based modeling tools, also known as "Web Modeling". Web-based modeling tools allow a user to access a modeling application over the network using HTTP. This provides a variety of advantages in some cases, such as as open cloud storage or real time collaboration. An important subset of modeling is also "Metamodeling". It is a modeling language which can be used by so called metamodeling platforms to support efficient development of modeling tools, as it defines the structure, syntax and rules of a model by which other models have to abide to.

In this paper, we will provide an overview of currently the most popular modeling tools with the focus being web-based modeling tools and metamodeling tools. Seeing as there is a large variety of topics which can be modelled, we will also portray how these modeling tools differ in aspects such as what their intended use is, if these technologies are open source, and much more.

Furthermore, we also implemented a website[1] that displays the current modeling tools that were found during this research. Any user visiting the website has the option to explore, search and filter existing tools based on multiple criteria, e.g., which modeling language is supported, whether logging in is mandatory, etc. In addition, anyone has the option to suggest new modeling tools or provide an update on a modeling tool already contained in the list.

[1] List of Modeling Tools: `http://me.big.tuwien.ac.at/`

Contents

CHAPTER 1

Introduction

Diagrams find usage in almost any topic as they provide an overview and a summary over a sequence of events, a hierarchical structure, an engineering problem or anything that can be divided into a set of components based on similar characteristics. The main purpose of a diagram is to translate a large amount of information through visualization. This has led to the large influx of newly developed modeling tools, however not each modeling tool may possess the desired features combined with the usability perks a user might desire. This may depend on many factors, such as is the complexity appropriate for the intended user, does it provide syntax for certain models, can multiple people edit the model at the same time, and more.

There are many applications which can be used for the purpose of modeling a diagram, even though it is not their intended use. A good example is the presentation program "PowerPoint" by Microsoft, primarily designed for presentations of certain topics, but with predefined shapes such as lines or rectangles, a user can create a model. Other apps such "Paint" can be also used to model diagrams, where a user can draw shapes or lend predefined shapes for creating diagrams. The following paper will not take apps that fall into that category into consideration, as they have no notion of what a valid model is, i.e., they do not know or enforce any metamodel but instead focus on different aspects and thus are not suitable for the purpose of modeling.

Though the meaning of a modeling tool is to a certain extend arbitrary as examples above have shown, our goal will be to outline tools with the specific purpose of modeling. We will mainly try to find apps that provide templates for simple diagrams such as mind maps or flowcharts up to diagrams used in software engineering and business informatics, such as UML diagrams, BPMN diagrams or ER diagrams.

This paper will provide an extensive overview of currently the most popular modeling tools with the focus being "web modeling tools" and "metamodeling tools". Furthermore we will attempt to categorize the collected tools based on shared properties.

This paper is organized as follows: Chapter 2 defines the terms used throughout this paper, paired with depicting our research criteria and strategy. Our web application [1] displaying modeling tools collected by us and possibly suggested by the community in the future is shown in chapter 3, whereby the research topics surrounding the surveyed modeling tools and their technologies are further discussed and compared in chapter 4. Chapter 5 mentions further work related to the area of modeling tools and concludes with final reflections in chapter 6.

[1] List of Modeling Tools: `http://me.big.tuwien.ac.at/`

CHAPTER 2

Survey of Modeling Tools and Metamodeling Platforms

2.1 Definitions

2.1.1 What *is* a modeling tool

A modeling tool is an application, framework or a library, that provides the service of constructing a diagram. The manner in which a diagram can be constructed can vary, though the most common options are either graphically, textually, or a mixture of both.

In the case of a graphical modeling tool, a user conventionally has a collection of shapes (e.g., rectangles, circles, lines) at his or her disposal on the side of the window, while the center and majority of the screen consists of an empty white window, onto which shapes can be dragged and dropped. The colors of shapes can usually be edited and the sizes can be adjusted, moved, duplicated and also deleted once they are in the editing window.

Users making use of textual modeling tools however do not need to touch the graphical interface and instead just type into a text-field where a certain syntax is expected to be upheld. Based on predefined terminology, the user can create shapes and connections (e.g., nodes and edges) by typing the respective keywords, which in turn generates the corresponding diagram. If an unexpected or undefined set of characters is detected, the graphical representation fails and the user is given an error message describing the cause for failed compilation.

2.1.2 What is *not* a modeling tool

Presentation programs (e.g., Microsoft PowerPoint) or graphics editors (e.g., Microsoft Paint, Adobe Photoshop) are not modeling tools. Though they provide basic shapes that often cover the basic requirements, its primary purpose is not modeling and generation

of diagrams. Though in a broad sense they could fall into the umbrella of the term "modeling tool", as they provide the basic requirements as defined in section 2.1.1, its primary purpose is not modeling and generation of diagrams. To avoid confusion for users unfamiliar with programs not intended for modeling, they will be left out.

Frameworks and libraries, which are part of aforementioned modeling tools on the other hand are included (e.g., PlantUML).

2.1.3 Web Modeling

Web Modeling in the context of this thesis refers to web-based modeling tools, which in turn are modeling tools that are available on the web and also provide their functionalities on the web. Modeling tools that need to be installed before they can be used are not considered web modeling tools.

2.1.4 Source code generation

When a modeling tool is capable of translating a diagram into code in a language such as C, Java etc., then the modeling tool can be described as providing the service of source code generation.

2.1.5 Cloud service

Cloud services are services, such as infrastructure, platforms or software, which are delivered on demand to companies and customers over the internet. There is a large trend towards cloud services and renowned companies such as Google Cloud Platform, Microsoft Azure and Amazon Web Services already provide cloud platforms. These services can be grouped into three following categories: Infrastructure as a Service (IaaS) offers hardware-related components, such as storage services. Platform as a Service (PaaS) presents development platforms, where cloud applications can run as a service. Lastly comes Software as a Service (SaaS), which provides a complete, ready-to-use software application as a service.

2.1.6 Real time collaboration

Real time collaboration refers to an application that allows multiple users to work on the same project at the same time. If for example one user changes the size of a diagram, the other users can see it in real time.

2.2 Research criteria

There are currently numerous modeling tool applications, frameworks as well as libraries. We have collected the most popular modeling tools based on the results of search engines by

Google[1], Bing[2] and DuckDuckGo[3]. Some of our keywords used during the search included "modeling tools", "web modeling tools", "metamodeling tools", "modeling editors", "model driven web engineering", and more. From the provided results we collected modeling tools that either linked directly to a modeling tool or to a website that presented a collection of modeling tools[Wik, Top]. Furthermore we collected modeling tools found in scientific papers with Google Scholar[4] and the ACM Library[5]. We omitted applications that did not meet our definition (section 2.1), appeared to be no longer in development or only provided basic shapes without any additional diagram specific type models or templates (e.g., UML diagram). Using this query we collected a total of 73 modeling tools.

Adonis	GoJS
Apache OpenOffice Draw	Graphity
Archi	Graphiti
ARGOuml	iGrafx
Astah	JetBrains MPS
BIGER Modeling Tool	JointJS
BIGUML	jslumb
BPMN.io	jsUML2 Editor
Cacoo	Lucidchart
Camunda BPM	MagicDraw
Chartmage	MelanEE
Circuit Diagram	Mermaid.js
ConceptDraw Diagram	MetaEdit+
Creately	MetaUML
dbdiagrams.io	Microsoft Visio
Dia	Mindfusion
Diagram Designer	Miro
Diagramo	Modelio
Diagrams.net	Moqups
DotUML	mxGraph
Eclipse GLSP	Nomnoml
Eclipse Papyrus	ObeoDesigner
Eclipse Modeling Framework	OpenPonk Modeling Platform
Edraw Max	Pencil Project
ER/Studio	pgModeler
FXDiagram	PlantUML
GenMyModel	ProcessOn
GitMind	Rational Rose (IBM)
Gliffy	SCADE

[1]Google: `https://www.google.com/`
[2]Bing: `https://www.bing.com/`
[3]DuckDuckGo: `https://duckduckgo.com/?va=b&t=hc`
[4]Google Scholar: `https://scholar.google.com/`
[5]ACM Digital Library: `https://dl.acm.org/`

Simulink	Umple
sketchboard	Visual Paradigm
Slickplan	WebGME
Software Ideas Modeler	Xtext
Swimlanes	yEd (yWorks)
Enterprise Architect	yFiles
StarUML	ZenUML
UMLetino	

2.3 Properties

For each modeling tool resulting from our research criteria, we collected two types of properties: Properties which are self evident and are explicitly stated and properties that cover a broad set of similar characteristics.

Self evident properties include the name of the modeling tool and a corresponding URL, where the modeling tool is accessible and/or downloadable from. In addition, simple yes or no questions are answered such as whether or not a modeling tool is open source, if it is available as a web application and/or desktop application, etc. We also documented which models and possibly templates of a certain modeling language were made available, who partook as a developer on the modeling tool project, which platforms support the modeling tool and with which programming languages the modeling tool was developed with (see Table 2.1).

Property	**Description**
Name	Name of the modeling tool
Link	Website link of the modeling tool
Open Source	Is it open source?
Web App	Is it available as a web application?
Desktop App	Is it available as a desktop application?
Modeling Language(s)	Name of a modeling language whose models and possibly templates are available
Source code generation	Does it provide source code generation?
Cloud service	Does it provide cloud service?
Log-in	Is log-in required?
Real Time Collaboration	Is real time collaboration possible?
Creator(s)	Who partook in the development of the modeling tool?
Platform(s)	Which platforms support the modeling tool?
Programming Language(s)	Which programming languages were used to develop the modeling tool?

Table 2.1: List of all self evident modeling tool properties

Figure 2.1 below displays yes/no-answerable questions to a property of all modeling tools collected during the survey in the form of a bar chart.

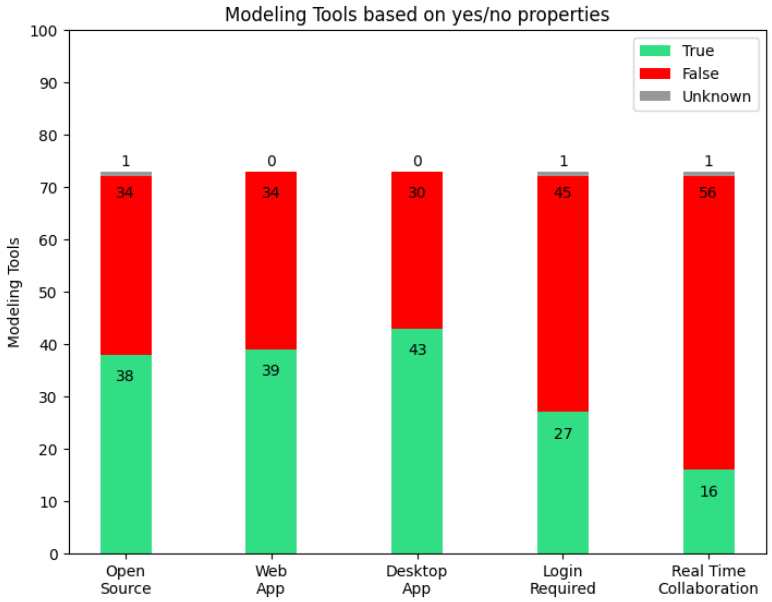

Figure 2.1: Bar graph of modeling tools based on yes or no properties

Umbrella properties consists of "Licenses" and "Technology".

2.3.1 License

The "License" property sums up software licenses that share similar characteristics. We have defined the following three types of licenses: "Free", "Restricted free content, commercial", and "Commercial".

Modeling Tools within the license cateogry "Free" have all of their content made accessible to any user for free.

"Restricted free content, commercial" means some contents within a modeling tool are available to the user, the rest is however only available if the user pays an additional fee.

"Commercial" sums up modeling tools with contents and functionalities none of which are available unless a user pays for an access.

Our survey collected modeling tools of which 33 are freely accessible, 28 have partial free content with the rest hidden behind a paywall, 11 commercial and 1 which is unknown (Fig. 2.2).

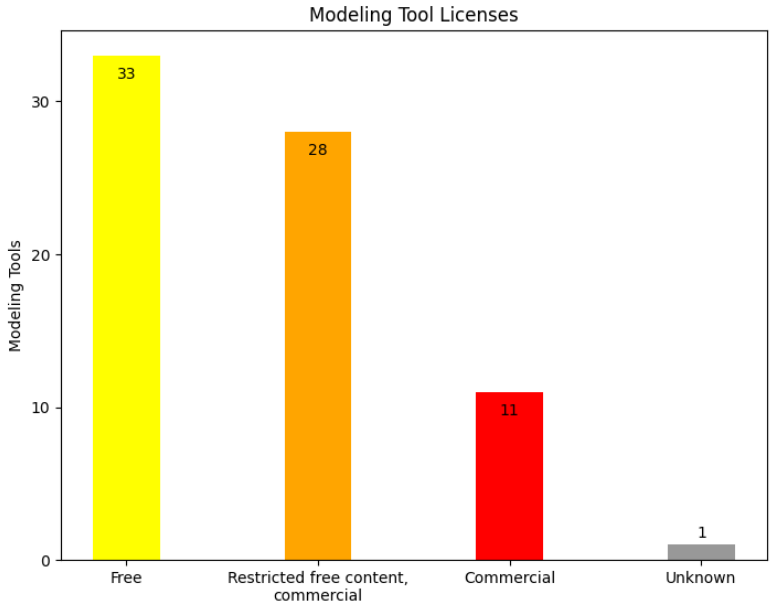

Figure 2.2: Bar graph of modeling tools based on licenses

2.3.2 Technology

The "Technology" property specifies the software type of a modeling tool and can take on the value of "App", "Framework", and/or "Library".

"App" refers to applications that are usable as soon as the user either loads the website of the corresponding modeling tool, or downloads it as a desktop application and can use it afterwards.

"Framework" refers to technology intended for being reincorporated within applications. Unlike a library, it does not contain finished functions, but instead can be described as a program scaffold that provides a blueprint, however is not a finished unit. The user is given the choice in terms of customization. While they are usually also available for usage on the homepage of their developers to allow first impressions, their purpose lies in being reincorporated by other "apps" and therefore are not self alone applications.

Lastly, a "Library" is usually found in software development where its functionalities serve as support within a program and thus are not standalone. It is not an independently executable unit, but an additional module that can be called by a program.

This thesis contains in total 56 apps, 11 frameworks and 6 libraries (Fig. 2.3).

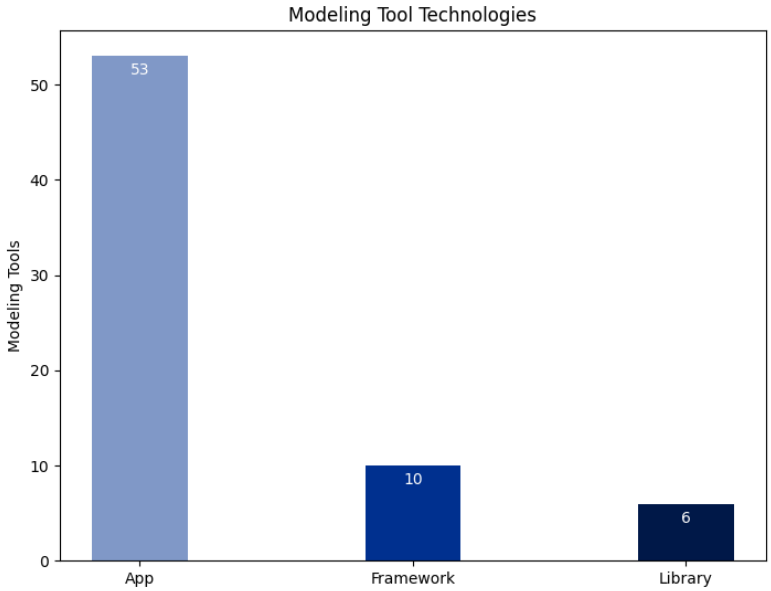

Figure 2.3: Bar graph of modeling tools based on technologies

2.4 Categories

Modeling tools have a diverse amount of use cases. This paired with the fact that we collected in total 73 modeling tools, we have decided to assign to each modeling a category based on the collected properties in section 2.3 Properties. A category in the context of a modeling tool serves as an umbrella term for a set of similar characteristics from which its most common intended use can be derived from. We have defined six categories: Text-based modeling tools, graphical modeling tools, drawing tools, business tools, metamodeling tools and mixed textual and graphical modeling tools.

Figure 2.4 displays the sum of all modeling tools assigned to each category in the form of a bar chart.

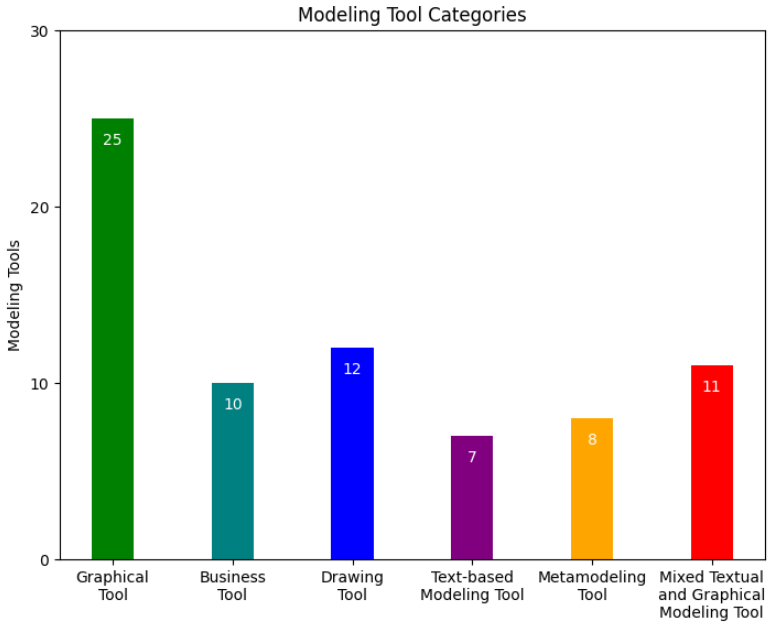

Figure 2.4: Bar graph of modeling tools based on categories

2.4.1 Text-based modeling Tools

Text-based modeling tools are usually either frameworks or libraries which the user can install and include in their projects. While the user can see the output visually, there is no "drag and drop" functionality, meaning a user cannot use the mouse cursor and drag a shape onto the modeling window, but instead must type the the name that refers to the preferred shape. The text is then compiled and displayed as a diagram, if it is syntactically correct.

The following sequence diagram example by PlantUML (Fig. 2.5) showcases how text-based modeling tools work. The code block defines entities and relations among entities. Both Alfa and Bravo are entities who sent among each other messages.

Name	Open Source	Modeling Languages
ChartMage	Yes	UML
ER/Studio	No	BPMN, ER
FXDiagram	Yes	Other
Nomnoml	Yes	UML
PlantUML	Yes	UML
Swimlanes	Yes	UML
Xtext	Yes	Other

Table 2.2: List of text-based modeling tools

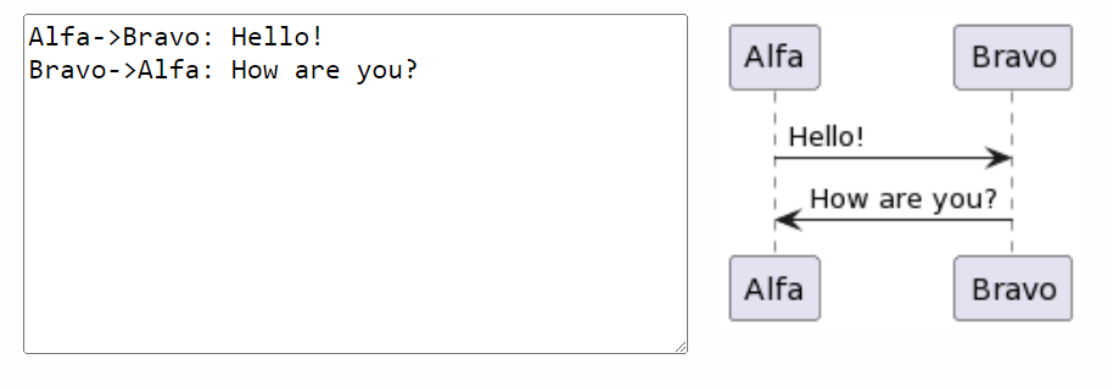

Figure 2.5: PlantUML command and corresponding graphical output

2.4.2 Graphical Modeling Tools

The following category provides support in regards to usability and possibly also syntax when creating a model. The user interface follows the same conventions as in the case of drawing tools (see Sect. 2.4.3) and business tools (see Sect. 2.4.4). The design of diagrams occurs graphically, where the user has a window in the center of the page where the diagram can be modelled. Shapes and arrows are provided in a tool bar on the side of the application window.

Figure 2.6 showcases support for the modeling language BPMN. When the user clicks on a shape, the tool displays a set of possible BPMN related shapes that can be connected with the clicked-on shape. The user can also omit that recommendation by picking a shape from the left sided toolbar.

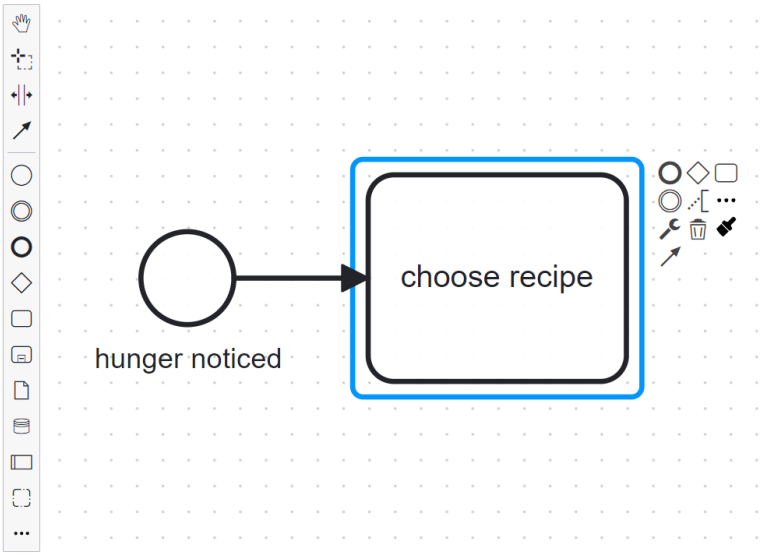

Figure 2.6: BPMN.io screenshot

Name	Open Source	Modeling Language(s)
Adonis	No	BPMN
ARGOuml	Yes	UML
Astah	No	UML, ER
BPMN.io	Yes	BPMN
Circuit Diagram	Yes	Other
Dia	No	Other
DotUML	N/A	Yes
Eclipse Papyrus	Yes	Yes
GenMyModel	No	BPMN, UML, ER
Graphity	No	BPMN, UML, ER
Graphiti	Yes	Other
jsUML2 Editor	Yes	Yes
MagicDraw	No	UML, SysML, BPMN
Modelio	Yes	UML, SysML, BPMN
OpenPonk	Yes	UML, BPMN
pgModeler	Yes	ER
Rational Rose	No	UML
SCADE	No	Other
Simulink	No	Other
Software Ideas Modeler	No	UML, SysML, BPMN, ER
Enterprise Architect	No	UML, SysML, BPMN, ER
Visual Paradigm	No	UML, BPMN
yEd	No	UML, BPMN, ER
yFiles	No	UML, BPMN

Table 2.3: List of Graphical Modeling Tools

2.4.3 Drawing Tools

The main goal of drawing tools lies in providing an easy-to-use application where a user can use shapes available by the modeling tool and position as well connect those arbitrarily. Same as graphical modeling tools (see Sect. 2.4.2) and business tools (see Sect. 2.4.4), the user uses the mouse to drag shapes from the toolbar onto the editing window. Unlike graphical modeling tools however, they do not provide any modeling language related support or warnings if a user designs a diagram that does not follow the rules of a certain modeling language. Drawing tools are usually free and open source as it allows users to submit additional templates as well as shapes and thus reach more users.

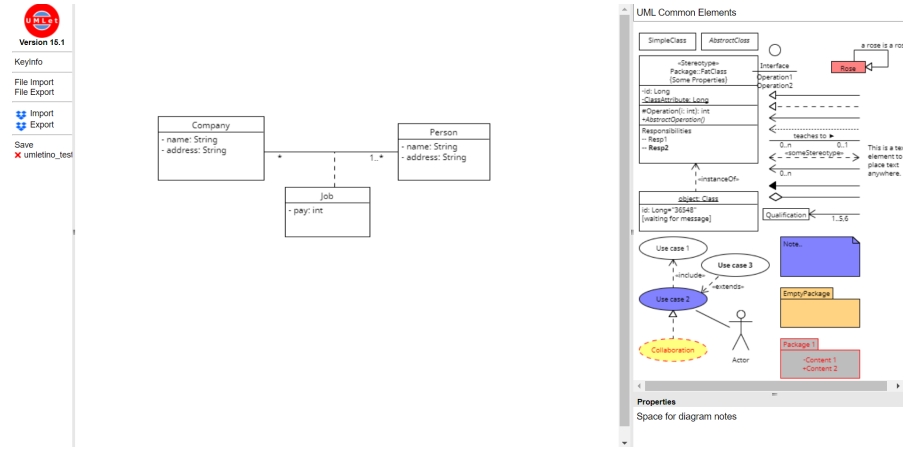

Figure 2.7: Diagram created with UMLetino

Name	Open Source	Modeling Language(s)
Apache OpenOffice Draw	Yes	UML
ConceptDragram	No	UML, BPMN
Diagram Designer	Yes	UML
Diagramo	Yes	Other
Diagrams.net/draw.io	Yes	UML, BPMN, ER
Lucidchart	No	UML, BPMN, ER
Mindfusion	Yes	UML
mxGraph	Yes	Other
Pencil Project	Yes	Other
ProcessOn	No	UML, BPMN
UMLetino	Yes	UML

Table 2.4: List of Drawing Tools

2.4.4 Business Tools

Business tool applications usually contain paid licenses and therefore are also not open source. On the other hand they provide perks such as real time collaboration and connection to other licensed programs. Most business tool diagrams are intended for the most common (modeling language agnostic) use cases that can be modelled with diagrams such as mind maps or flowcharts.

Name	Open Source	Modeling Language(s)
Cacoo	No	BPMN, Flowchart
Creately	No	UML, BPMN, ER
Edraw Max	No	UML, BPMN, ER
Gliffy	No	UML, BPMN
iGrafx	No	BPMN
Microsoft Visio	No	UML, BPMN
Miro	No	UML, BPMN, ER
Moqups	No	UML, SysML, BPMN
sketchboard	No	UML
Slickplan	No	UML, BPMN, ER

Table 2.5: List of Business Modeling Tools

2.4.5 Metamodeling Tools

Metamodeling Tools differ from all the other modeling tools in the regard that they are a platform purpose lies in efficient implementation of new modeling tools. This means the user defines the rules of a certain model, based on which then models can be created. Graphical modeling tools follow the syntax of meta models.

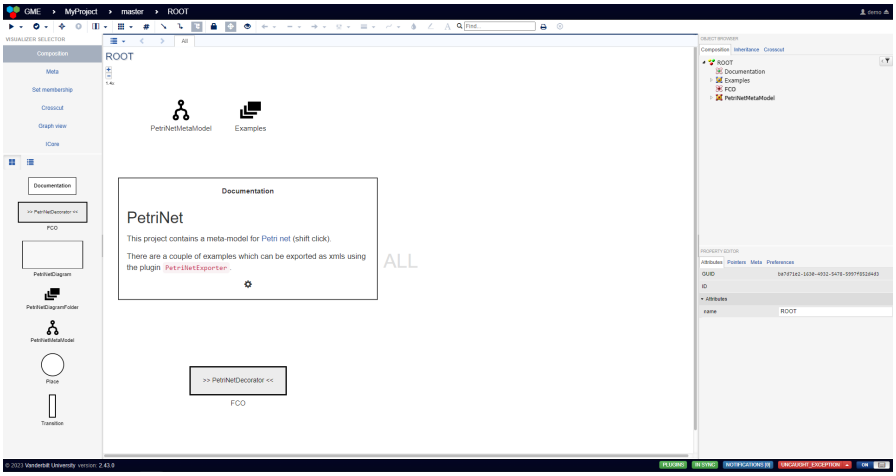

Figure 2.8: Screenshot of WebGME

Name	Open Source	Modeling Language(s)
Archi	Yes	ArchiMate, Metamodeling
Eclipse GLSP	Yes	Metamodeling
JetBrains MPS	No	Metamodeling
MelanEE	No	Metamodeling
Mermaid.js	Yes	Metamodeling, UML
MetaEdit+	No	Metamodeling
MetaUML	Yes	UML
WebGME	Yes	Metamodeling

Table 2.6: Caption

2.4.6 Mixed textual and graphical modeling tools

Modeling tools from this category are a combination of text-based modeling tools (see Sect. 2.4.1) and graphical modeling tools (see Sect. 2.4.2), meaning the choice is up to the user if the model will be created by typing the structure within a text field, or by creating it using the graphical user interface and mouse.

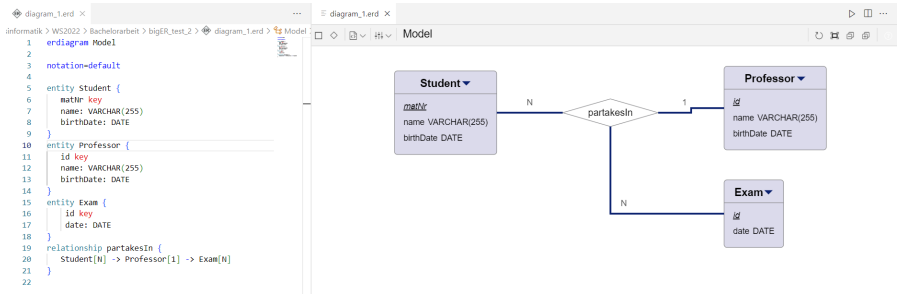

Figure 2.9: An Entity-Relationship (ER) diagram modelled with bigER Modeling Tool

Name	Open Source	Modeling Language(s)
bigUML	Yes	UML
dbdiagram.io	No	ER
jslumb	Yes	Others
JointJS	Yes	Others
bigER Modeling Tool	Yes	ER
Camunda BPMN	Yes	BPMN
GoJS	Yes	UML, BPMN, ER, Flowchart
ObeoDesigner	Yes	UML, SysML, BPMN
StarUML	Yes	UML, ER, SysML
Umple	Yes	UML
ZenUML	Yes	UML

Table 2.7: List of mixed textual and graphical modeling tools

CHAPTER 3

Modeling Tool Survey - Website

To provide an overview over the current modeling tools in use, we have decided to create a website which is available at `http://me.big.tuwien.ac.at/`, whereby the repository is also accessible on GitHub[1]. The user is presented with a short introduction to the topic and is also shown the current modeling tools in the form of a table, where each row contains a modeling tool and each column represents a property. A table cell is the status of a modeling tool property of the corresponding row and column. Any user can suggest a completely new modeling tool or propose an update to multiple properties of a modeling tool.

3.1 Use Case

The person visiting the website[2] can be either a regular user or an administrator. Any user can view the list of modeling tools, download all modeling tools in the form of CSV and/or JSON. Furthermore any user can suggest a new modeling tool as well as suggest a modeling tool edit. Modeling tool suggestions get stored and displayed on the website after an administrator has confirmed the suggestion (Fig. 3.1).

[1]Repository: `https://github.com/me-big-tuwien-ac-at/modeling-tool-repo`
[2]List of Modeling Tools: `http://me.big.tuwien.ac.at/`

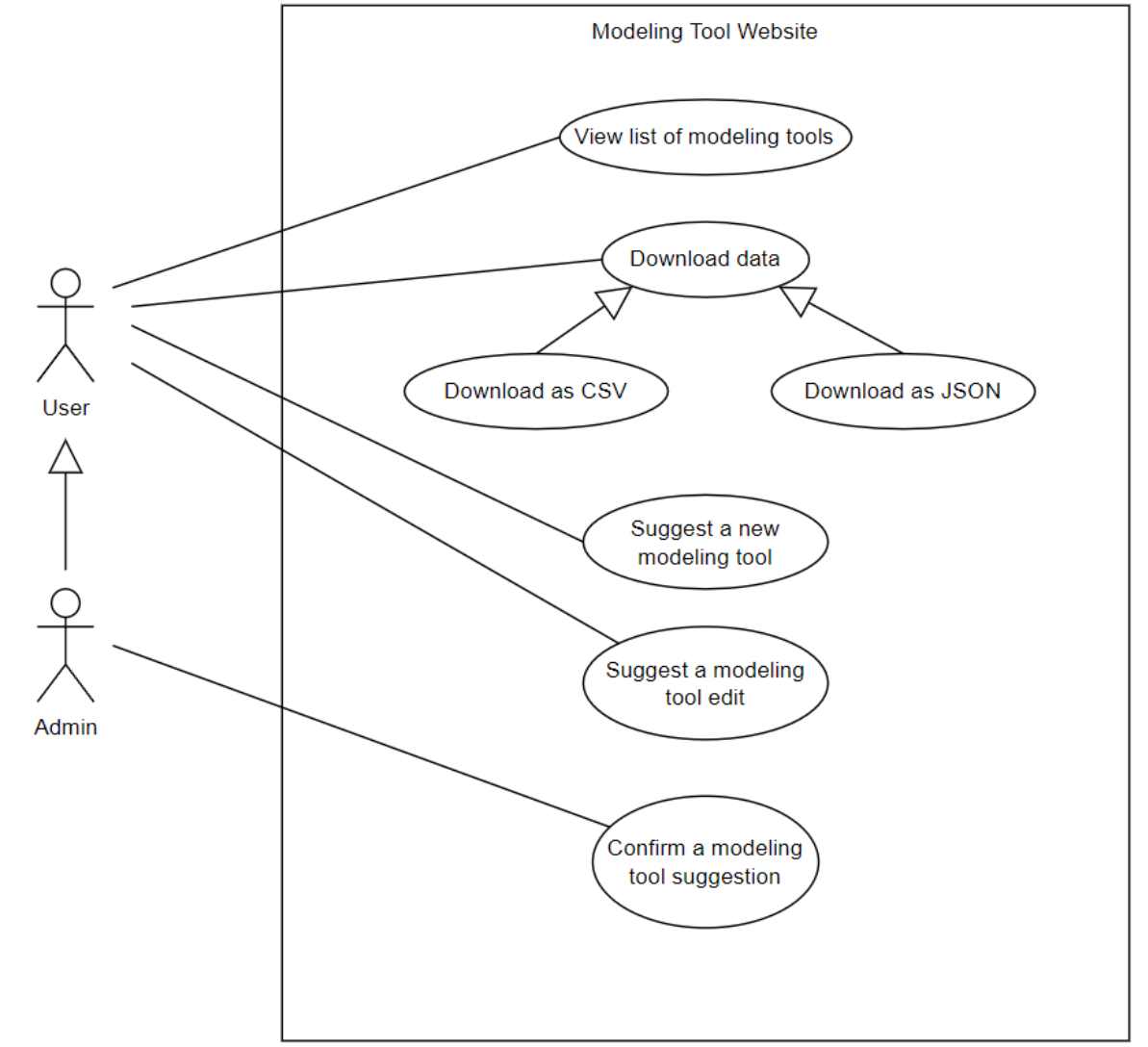

Figure 3.1: UML Use Case Diagram of the Modeling Tool Website.

3.2 Technology

The web application[3] is divided into a backend and a frontend. The backend represents the data access layer and handles the server side, while the frontend is the presentation layer and handles client requests. The application was programmed with the programming language Java OpenJDK 17 and uses the java-framework Spring Boot 3.0.1. It uses the build automation tool Maven 3.x.x and the database engine H2. For testing purposes,

[3]List of Modeling Tools: `http://me.big.tuwien.ac.at/`

the text-framework is Junit 5.x.x. The frontend-framework is Angular 14.x.x, paired with the build & dependency manager npm 9.x.x. Version control is handled by GitHub.

Table 3.1: Overview over all technologies used by the web application

Programming Language	Java OpenJDK17
Java-Framework	Spring Boot 3.0.1
JavaScript Runtime	Node.js 18.x.x
Frontend Framework	Angular 14.x.x
Database	H2
Test-Framework	Junit 5.x.x
Build & Dependency Management	Maven 3.x.x, npm 9.x.x
Versioning	Git

3.2.1 System structure

The storage of collected modeling tools and its corresponding properties within the persistent data store is handled by the server (backend). As a user can not only suggest a modeling tool, but also new properties contained within the new modeling tool, we implemented two abstract classes: *ModelingTool* and *Property*. *ModelingTool* is extended by *ModelingToolVerified* and *ModelingToolSuggestion*, while *Property* is extended by *ModelingLanguage*, *ProgrammingLanguage* and *Platform*.

Each *Property* contains a unique ID, a Boolean attribute named *deletable* and a *name* property of type String. The field *deletable* determines, whether a modeling tool property should be deleted once it no longer refers to any *ModelingTool* entity.

The class *ModelingTool* is abstract as its main purpose is to cover properties shared by both *ModelingToolVerified* and *ModelingToolSuggestion*. Each *ModelingTool* possesses a unique ID-property that functions as a primary key in our database. Properties name and link are of type String, whereby *name* property refers to the name of the modeling tool and *link* represents the website link. The property *creator* is a String array and contains all people who partook on the development of the modeling tool. Properties which can be answered with yes or no answers are of type Boolean and include *openSource*, *webApp*, *desktopApp*, *sourceCodeGeneration* (created diagrams can be translated into code), *cloudService*, *loginRequired* and *realTimeCollab* (users can edit a diagram together in real time). The category, license and technology properties are enumerations of type *Category*, *License* and *Technology* array respectively. Lastly, a modeling tool has a many-to-many (N:N) relationship with classes *ModelingLanguage*, *ProgrammingLanguage* and *Platform*, whereby all three are subclasses of the abstract class *Property*.

ModelingToolVerified represents all modeling tools that have been verified by the administrator, meaning the modeling tool is not a copy and all of its properties correspond to the modeling tool attributes it is referring to. *ModelingToolSuggestion* is any modeling tool suggested by the user and contains the additional information *modelingToolId* capturing

the unique ID of the modeling tool it is referring to. Furthermore we store the email of a user (*userEmail*) for security purposes, e.g., spam, and also an token (*adminToken*) which can be redeemed by the admin if the modeling tool is meant to be verified

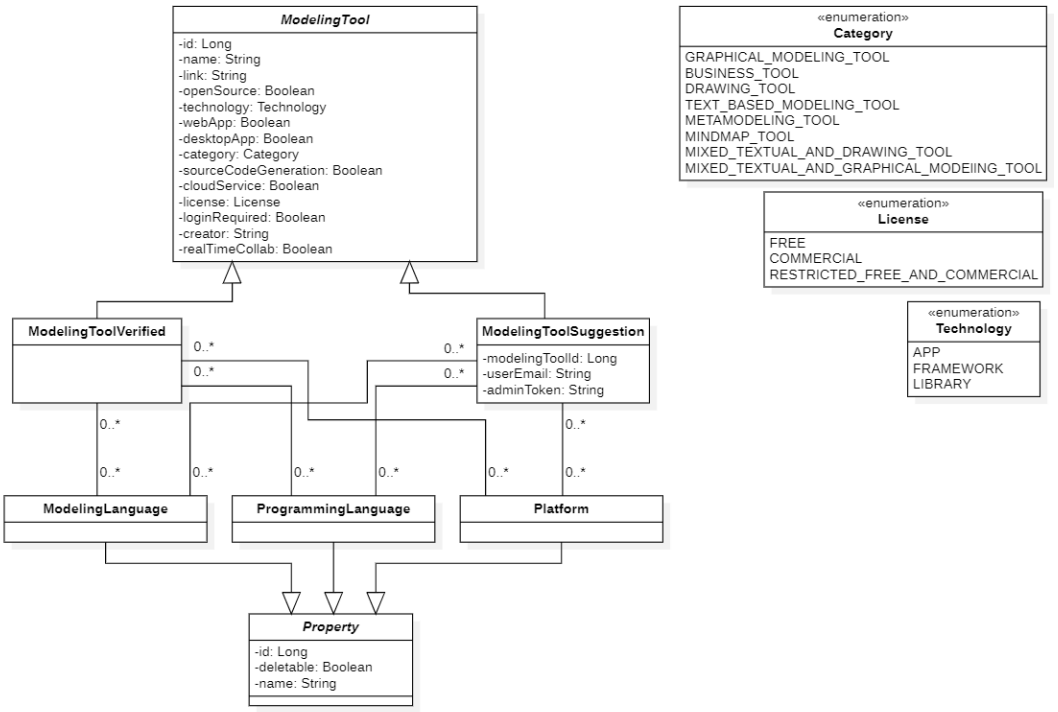

Figure 3.2: UML class diagram displaying the structure of the backend within the web application

3.3 Webpage

Our web application supports[4] CRUD operations for both the users and the administrators. Any user can suggest a new modeling tool on our website, or submit an edit for an existing modeling tool. Additionally, the user can also suggest new properties when submitting a new suggestion. As such we have divided our web application into different pages for their main purposes.

3.3.1 Homepage

When loading the web page for the first time, the user is met with the title of the web site "Modeling Tools" and an expanded section called "General", comprised of sets of definitions regarding modeling tools. Next to the definition is a screenshot of a diagram

[4]List of Modeling Tools: `http://me.big.tuwien.ac.at/`

created with the modeling tool BIGER[5]. The "General" section is followed by sections "Examples", "Table Columns" and "List of Modeling Tools" sections. Above the main body is a disclaimer and a header. The disclaimer elaborates by whom and what purpose this project was developed. Below the disclaimer is a header that is divided into three sections horizontally. The left sections consists of two buttons titled "Suggest a Modeling Tool" and "Edit a Modeling Tool", which when clicked on reroute the user to the corresponding pages. Within the centre of the header is the web site icon, which sends the user back to the homepage when it is clicked. The right section displays a sun or a moon icon and serves as a theme switch (Fig. 3.3). The bottom of the page contains a footer providing the email handling requests and suggestions by the user, a link to the GitHub-repository[6] and an image of the BIG TU Wien institute linking to their official page[7]. The footnote ends with a copyright disclaimer (Fig. 3.13).

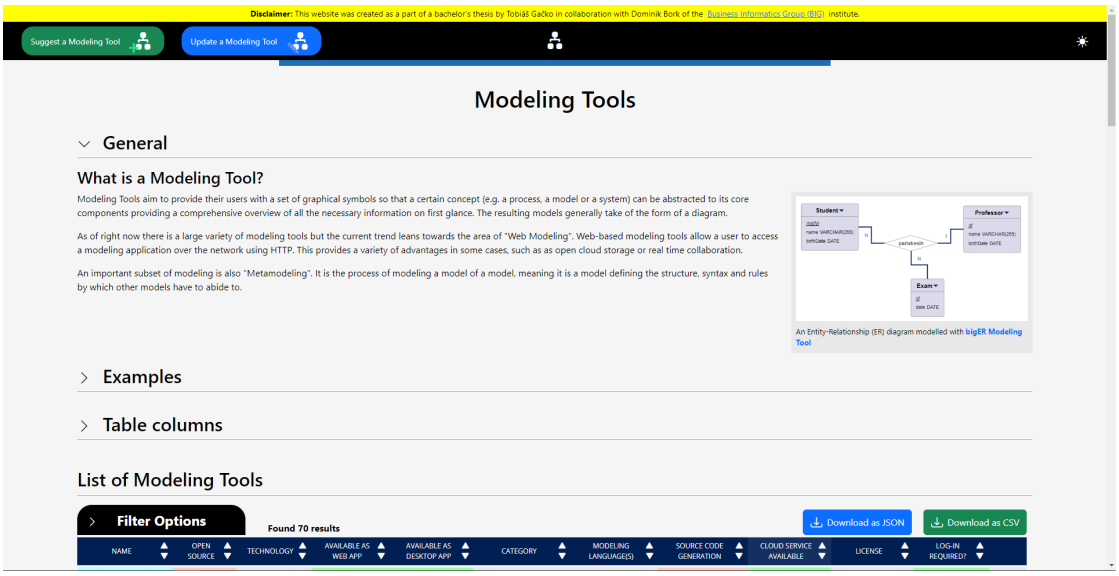

Figure 3.3: Web Application - Homepage start (1920 × 1080 window)

[5]bigER: https://github.com/borkdominik/bigER/blob/main/README.md
[6]Repository: https://github.com/me-big-tuwien-ac-at/modeling-tool-repo
[7]Business Informatics Group (BIG): https://www.big.tuwien.ac.at/

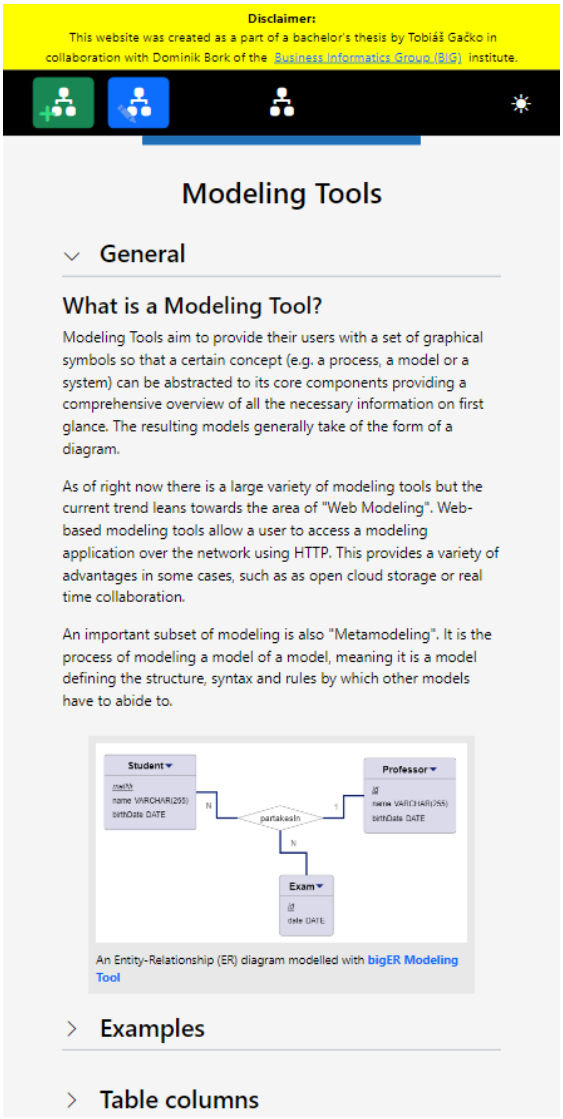

Figure 3.4: Web Application - Homepage start (600 × 1200 window)

Clicking on the sun within the header sets the theme of the page to dark, clicking on the moon switches it back to light. The default theme is set to light (Fig. 3.5).

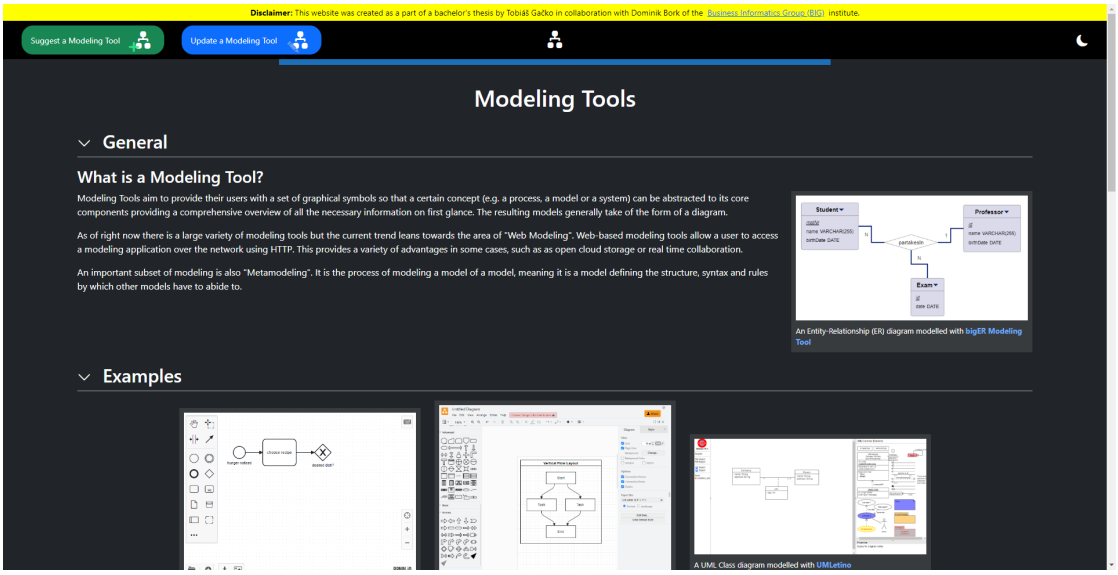

Figure 3.5: Web Application - Homepage start dark themed (1920 × 1080 window)

The examples section contains screenshots of diagrams made with various modeling tools (Fig. 3.6). Currently there are four images that have been made with BPMN.io[8], app.diagrams.net[9], UMLetino[10] and DotUML[11]. The user can zoom in on the picture by clicking on it and exit the window by either clicking on the × button in the upper right corner or simply outside of image window (Fig. 3.7).

[8]BPMN.io: `https://bpmn.io/`
[9]app.diagrams.net: `https://app.diagrams.net/`
[10]UMLetino: `https://www.umletino.com/umletino.html`
[11]DotUML: `https://dotuml.com/index.html`

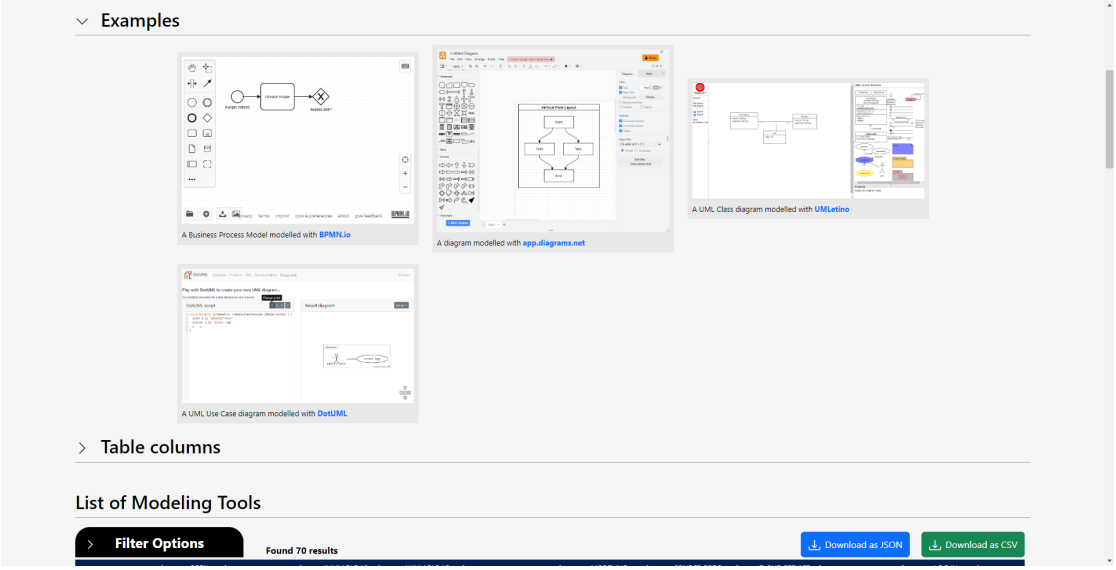

Figure 3.6: Web Application - Homepage section "Examples" (1920 × 1080 window)

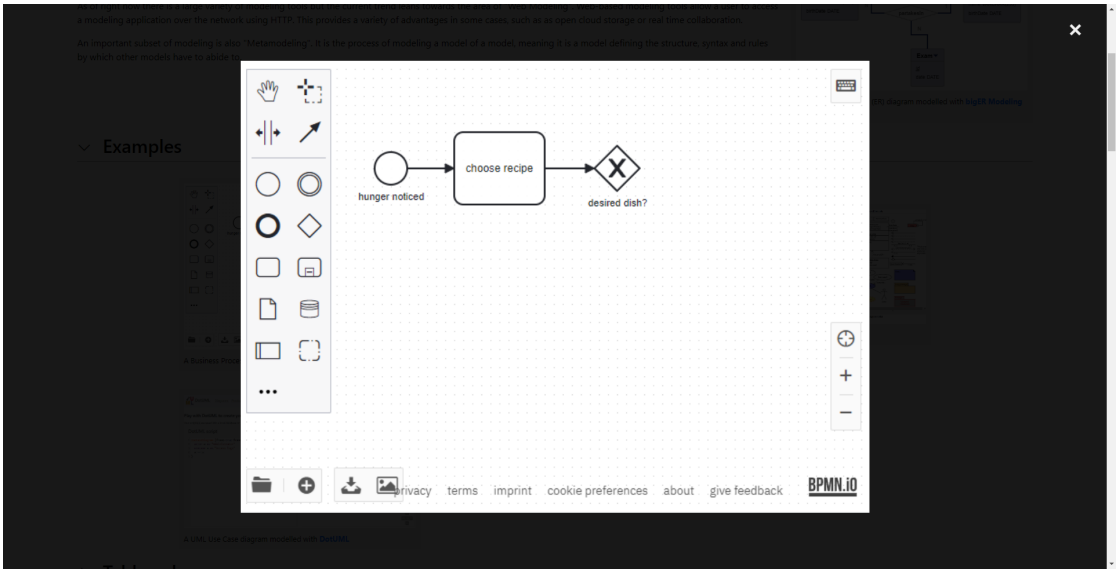

Figure 3.7: Web Application - Homepage example zoomed in (1920 × 1080 window)

The section "Table columns" explains in further details certain columns we have opted to add to our modeling tools table.

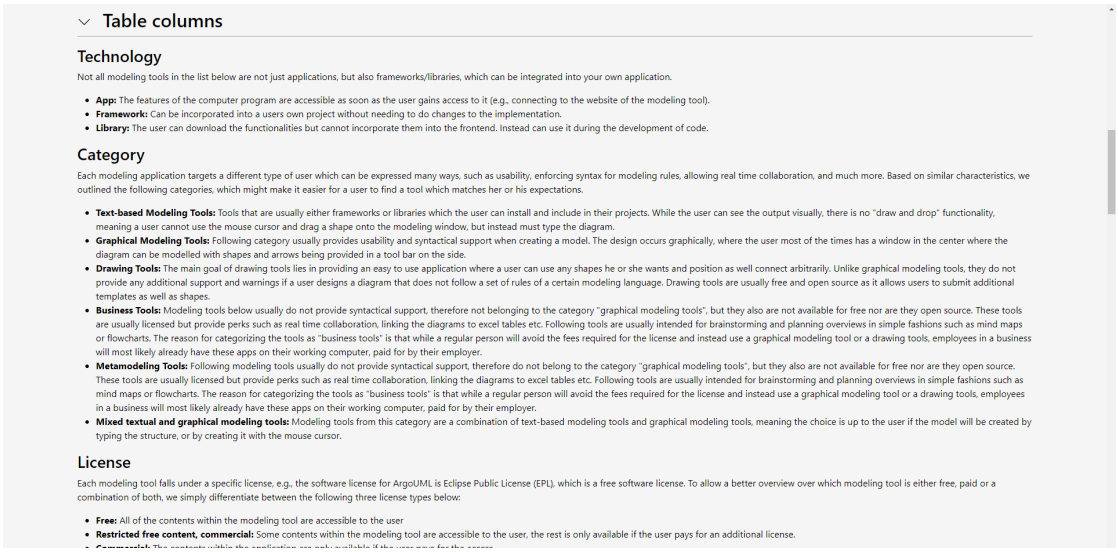

Figure 3.8: Web Application - Homepage section "Table columns" (1920 × 1080 window)

The "List of Modeling Tools" section displays every collected modeling tool in a table (Fig. 3.9). Each row consists of a modeling tool and each column represents a property, though the last column is a button represented with a pencil for the purpose of editing a modeling tool. The user is redirected to the "Edit" window of the respective modeling tool by clicking on the aforementioned button. The user can order modeling tools based on the properties by clicking on a property (Fig. 3.10).

Above the table is a collapsed "Filter Options" button, which when clicked on expands and displays the options to filter columns or to filter modeling tools to the left of the table (Fig. 3.11). If the user ticks too many columns within the "Filter Table Columns" window and they exceed the width of the table, then the user has to scroll horizontally to see the hidden entries. The user can click on the button "Optimize Columns", which ticks the maximum amount of columns that fit into the table based on the user's window size. This is done in the order of top to bottom. The user can also filter modeling tools by their properties, which is done using conjunction (AND operator) (Fig. 3.12). Names of modeling tools are matched if the typed character sequence is a subset of the modeling tool name stored in the persistent data store. The search is case insensitive and ignores trailing spaces. The user can click on reset to clear the query parameters (Fig. 3.13).

Right of the "Filter Options" button the user is shown the amount of matches found each time the page is loaded or a query is made. Above the table to the right, the user is met with the options to download all modeling tools either as CSV and/or JSON.

Figure 3.9: Web Application - Homepage section "List of Modeling Tools" with filters collapsed (1920 × 1080 window)

Figure 3.10: Web Application - Homepage section "List of Modeling Tools", table ordered by modeling tool name alphabetically in reverse order (1920 × 1080 window)

Figure 3.11: Web Application - Homepage section "List of Modeling Tools" with filters expanded (1920 × 1080 window)

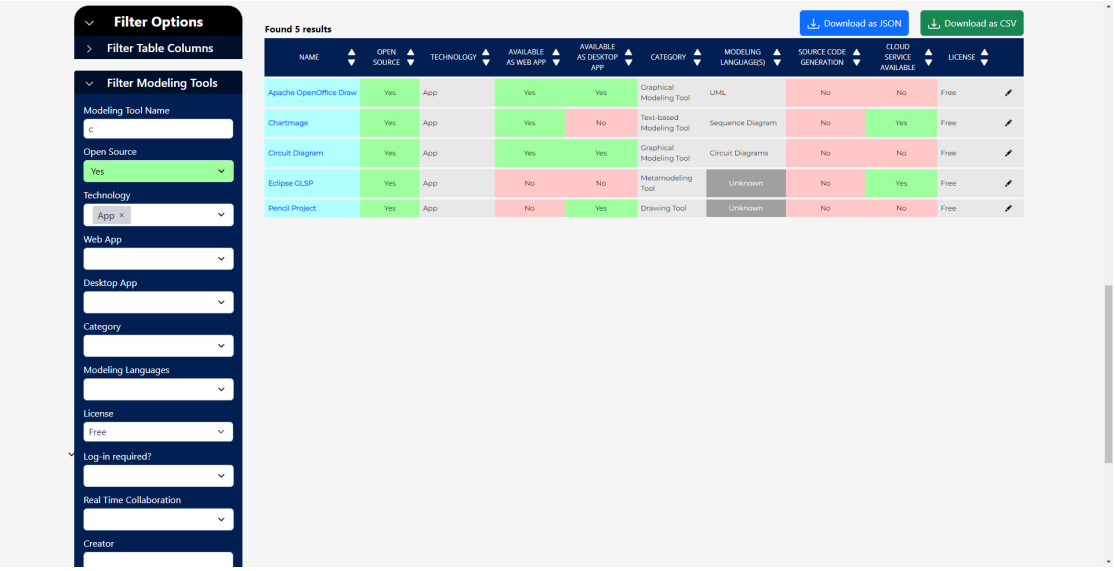

Figure 3.12: Web Application - Homepage section "List of Modeling Tools" with filtered entries. Searching for modeling tools which have the letter "c" in them, are open source, fall under the technology category "App" and are a free license (1920 × 1080 window).

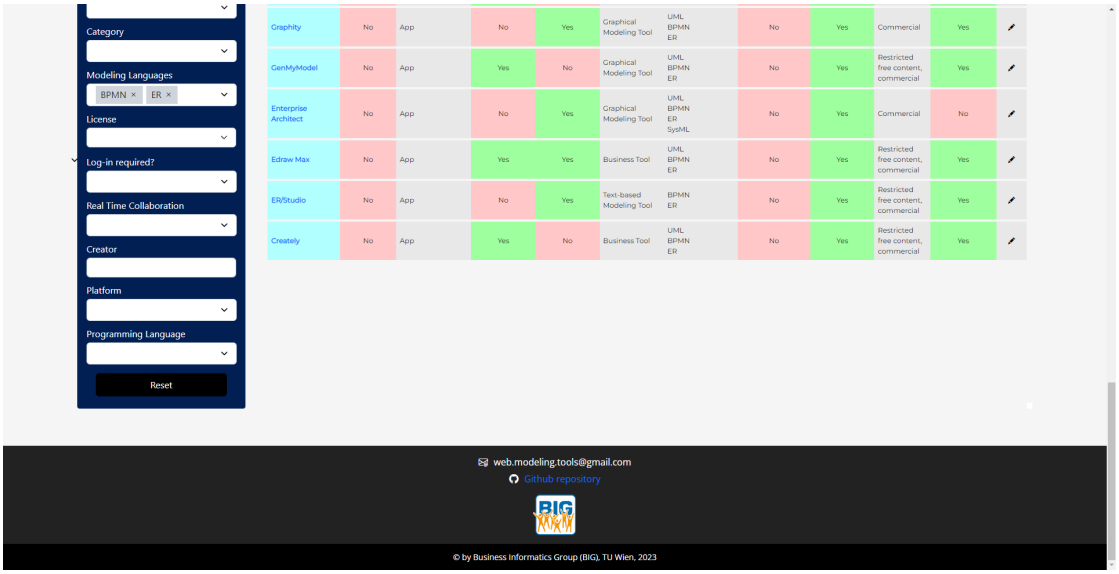

Figure 3.13: Web Application - Homepage section "List of Modeling Tools" reset filters option and footer (1920 × 1080 window).

3.3.2 Suggesting a new modeling tool

The user can suggest a new modeling tool by clicking on the green button in the top left within the header (Fig. 3.3) which reroutes the user a modeling tool form (Fig. 3.14).

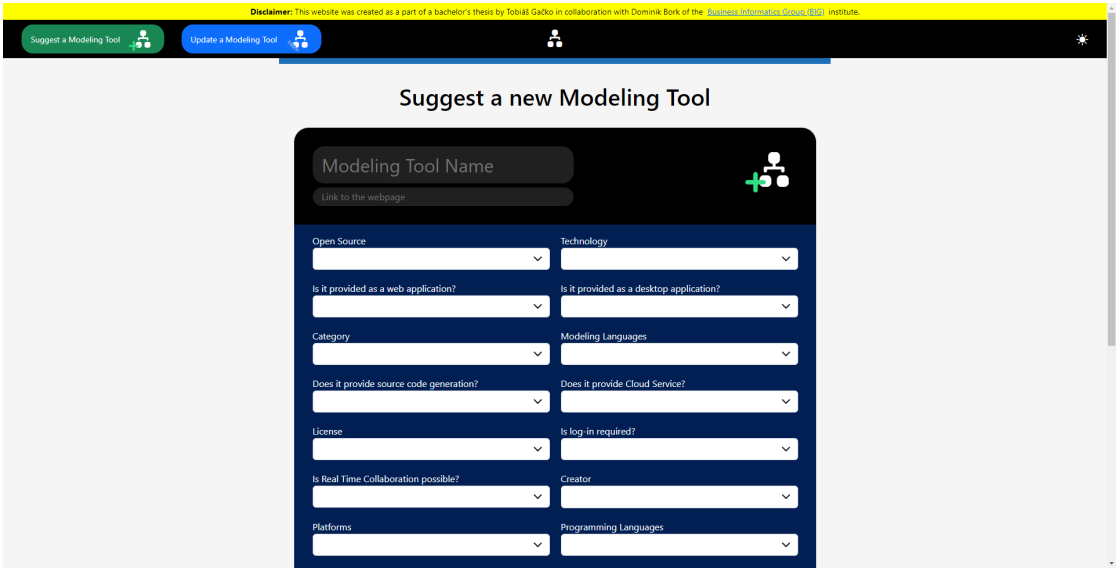

Figure 3.14: Web Application - Suggest a new Modeling Tool (1920 × 1080 window).

The user can specify for the following modeling tool information: modeling tool name,

web-page of where the modeling tool application can be accessed, is it open source, technology, is it provided as a web application, is it provided as a desktop app, what category does it fall under, which modeling languages does it support, does it provide source code generation, does it provide cloud service, as what type of license can it be classified as, is log-in required, is real time collaboration possible, who partook in the creating of the modeling tool, which platforms does it support and with which programming languages has it been developed with. Furthermore, any user can provide additional information or remarks (Fig. 3.15).

Before a modeling tool can be submitted, the user has to provide a name of the modeling tool, the web-page link, his or her own valid email and pass the reCAPTCHA test (used for avoiding spam from bots). The name of the modeling tool must be unique. If a user provides a name for a modeling tool, with which another modeling tool is already named with, then the user is notified and given the option to switch to edit mode (Fig. 3.16).

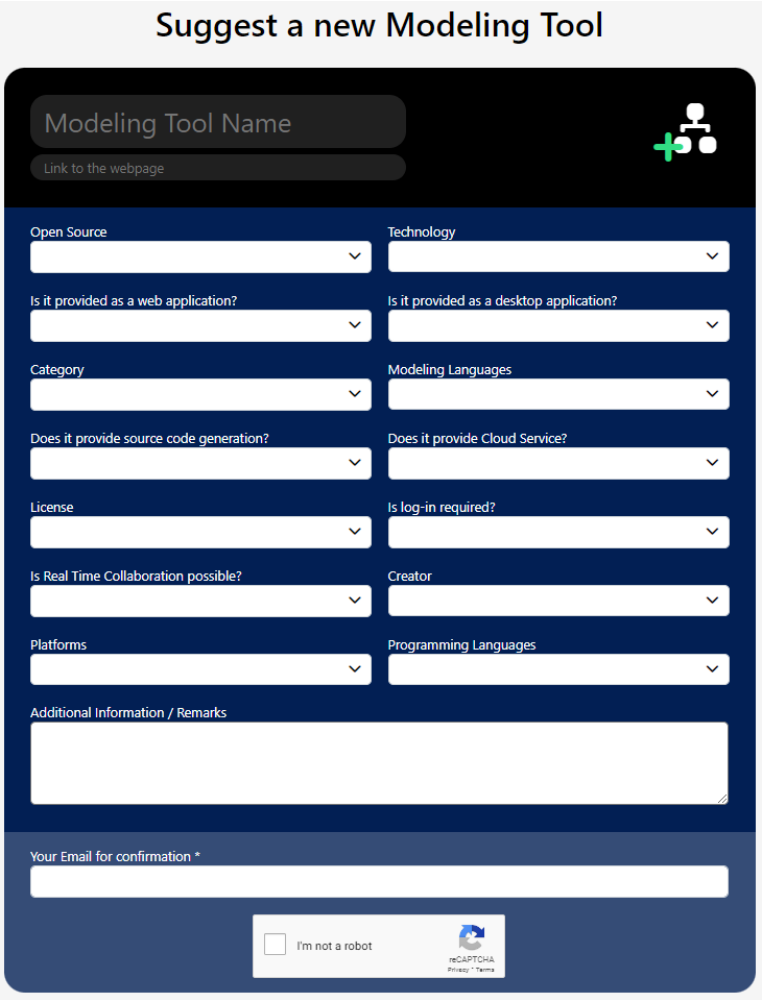

Figure 3.15: Web Application - Suggest a new Modeling Tool form.

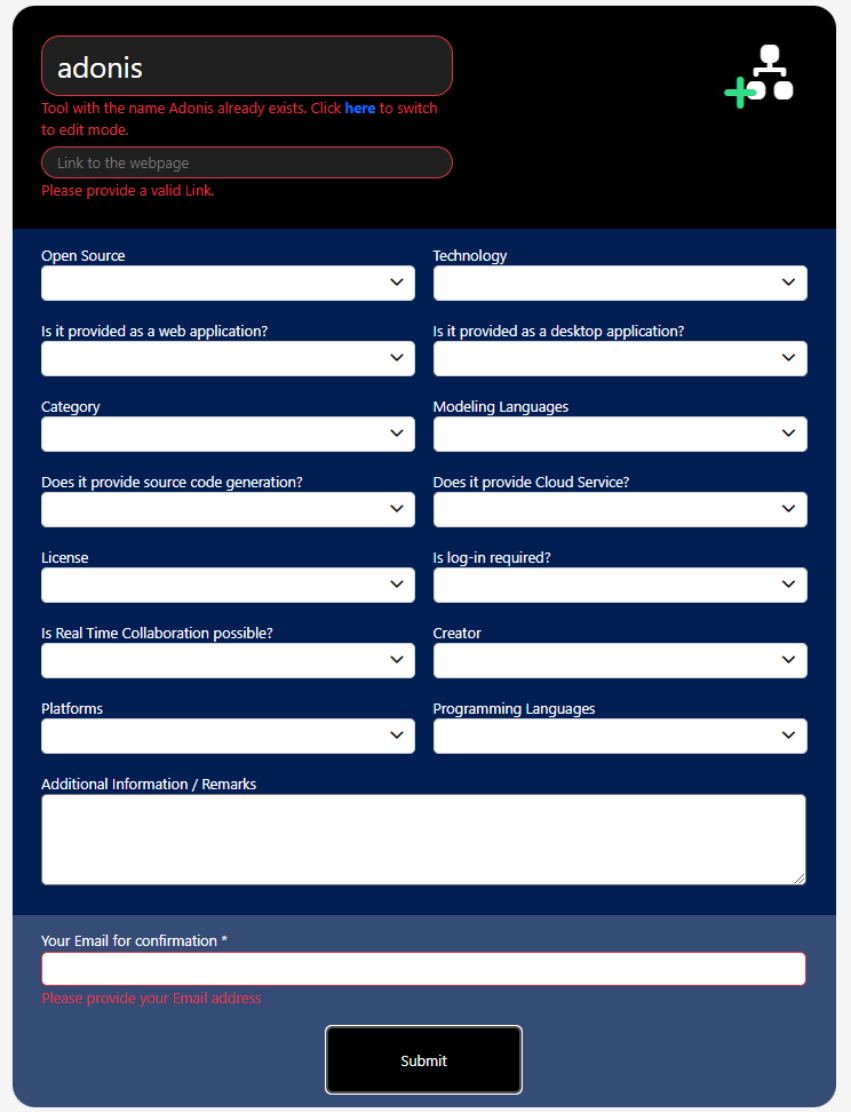

Figure 3.16: Web Application - Invalid modeling tool suggestion.

After successfully submitting a modeling tool suggestion, the user is redirected back to the homepage and thanked for the contribution (Fig. 3.17). In addition, the user also receives an email displaying the sent suggestion in a JSON format (Fig. 3.18) as does the admin (Fig. 3.19). The admin has the option between confirming the modeling tool suggestion and storing it as a modeling tool, or to dismiss the suggestion and delete it from the persistent data store. The links within the admin email contain tokens which when clicked on activate the desired action.

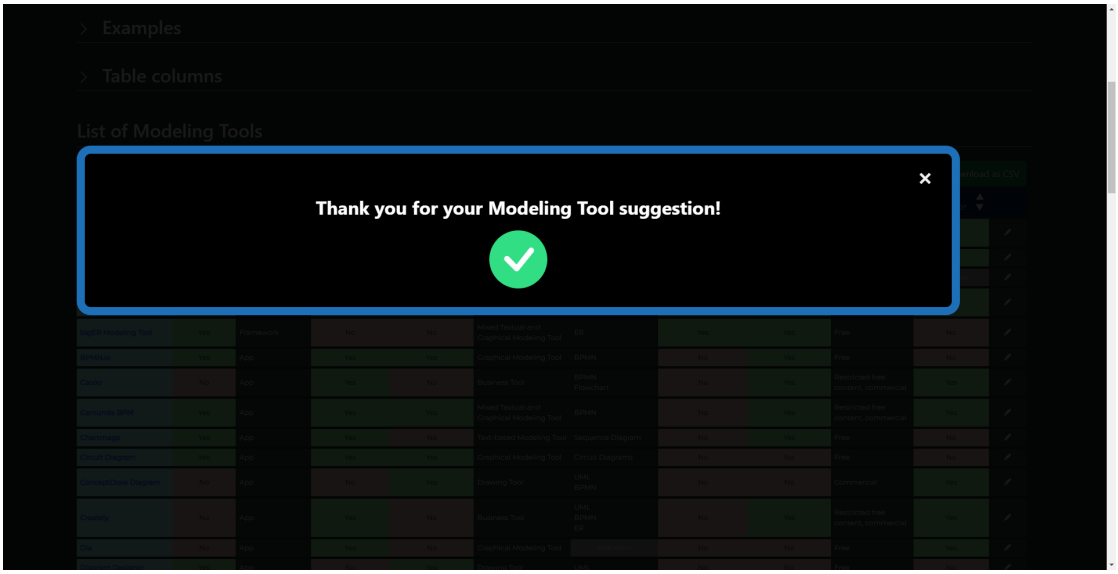

Figure 3.17: Web Application - Response after submitting a new modeling tool or a modeling tool edit.

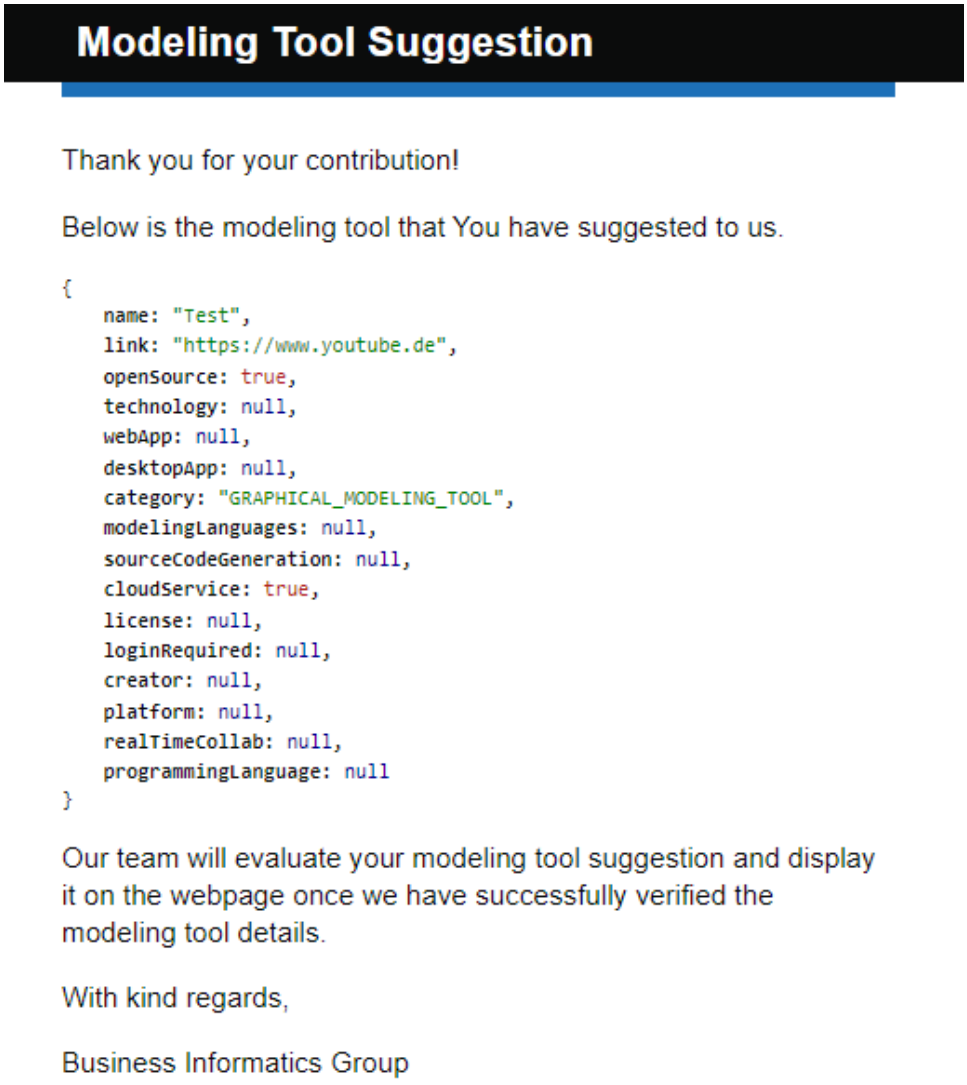

Figure 3.18: Web Application - Email response to the user after submitting a new modeling tool

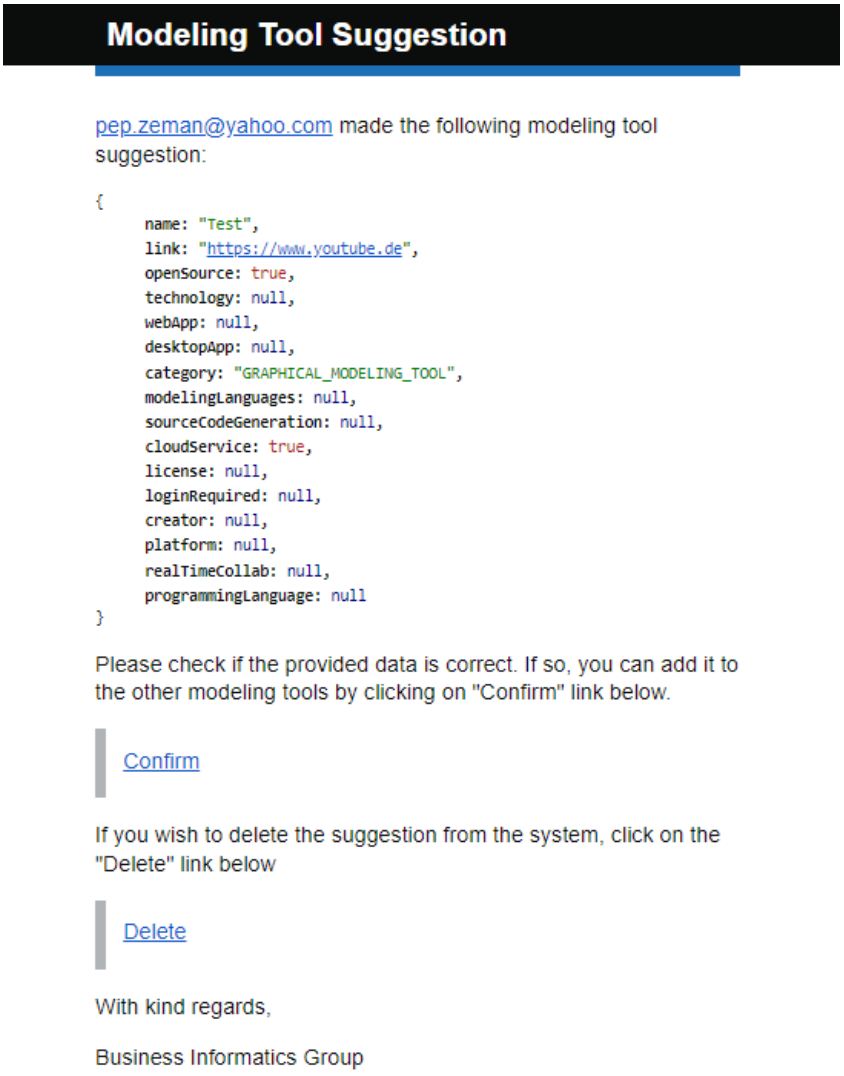

Figure 3.19: Web Application - Email response to the admin after a user has submitted a new modeling tool

3.3.3 Suggesting a modeling tool edit

There are two ways to propose an edit to a modeling tool - either by clicking on the blue button within the header on the left (Fig. 3.3), or clicking on the pencil button of a respective modeling tool within the table in the section "List of modeling tools" (Fig. 3.9). Choosing the former option, the user is redirected to a search field where the modeling tool which is expected to be edited, can be found. The search query is case insensitive and matches all modeling tools which contain the character sequence provided by the user in its name (Fig. 3.20). Clicking on one of the rows redirects the user to the editing

form, which has the same format as when a user attempts to suggest a new modeling tool (Fig. 3.3.2).

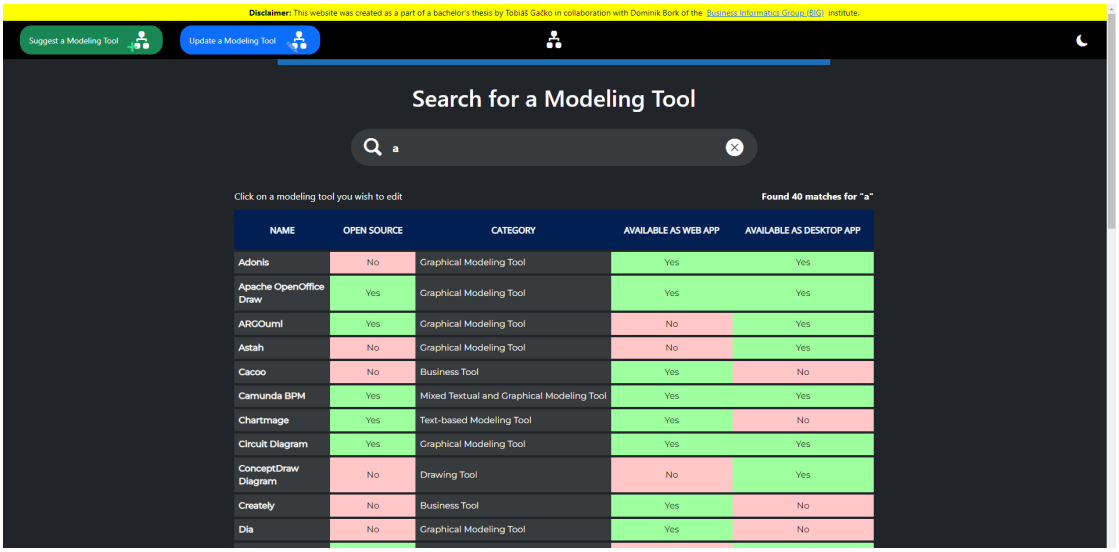

Figure 3.20: Web Application - Searching for modeling tool that contain the character "a" intended for the purpose of suggesting an edit.

The entries of the modeling tool that the user attempts to edit are filled in within the form. The user also has the option to reset the entries to their initial values after editing them (Fig. 3.21). Before the user can submit an edit proposal, either at least one entry needs to be edited or the feedback field must not be empty (Fig. 3.22).

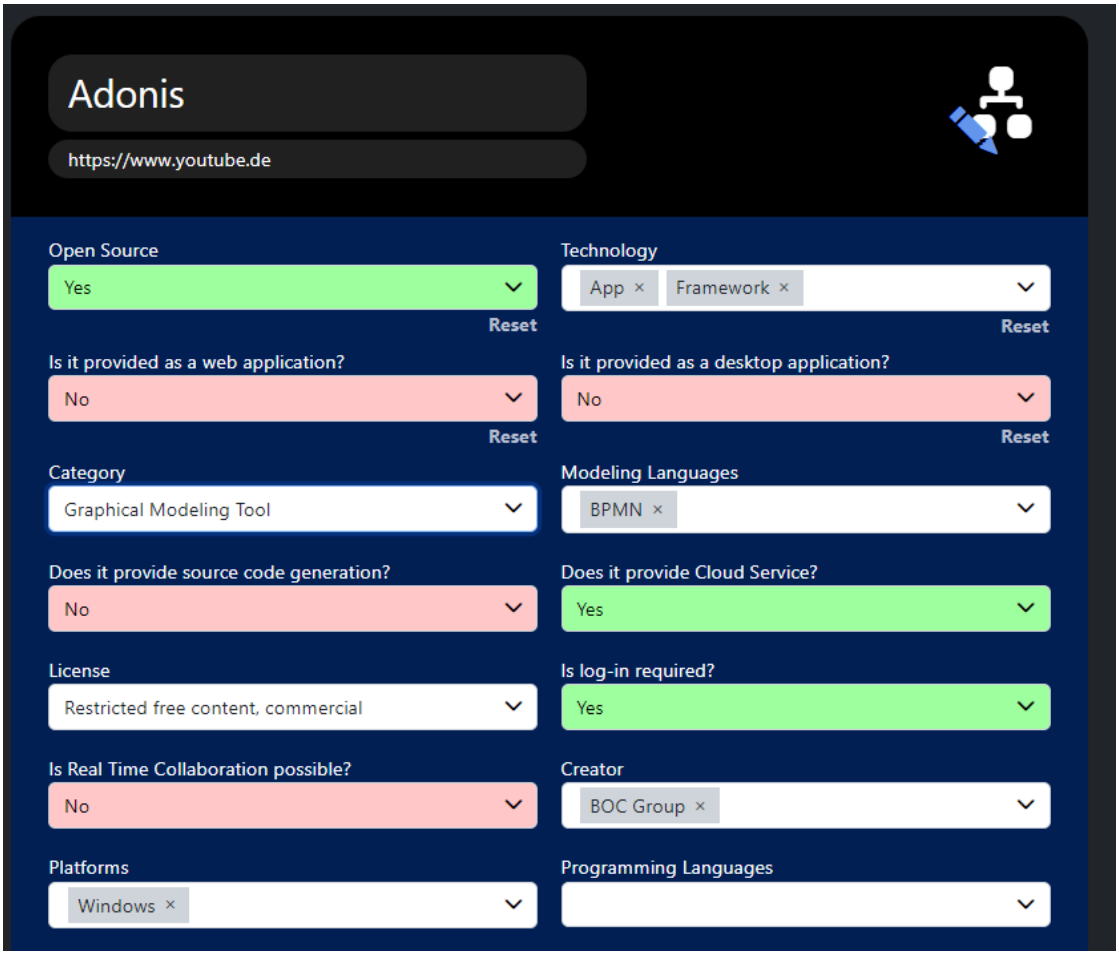

Figure 3.21: Web Application - Options to reset values after editing them (in this example Adonis).

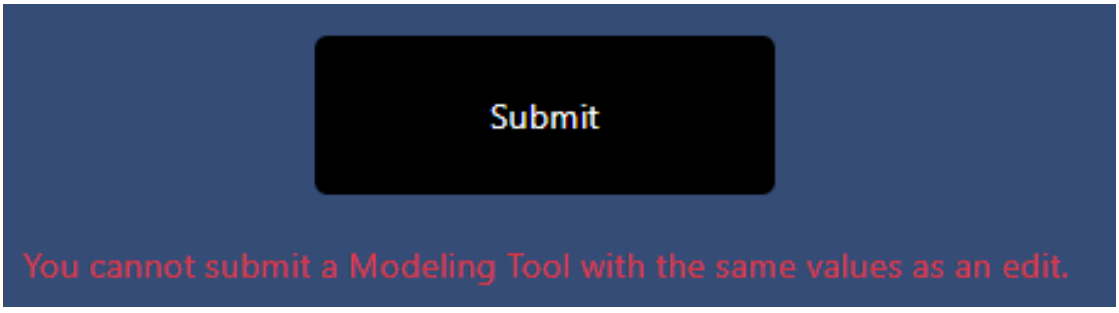

Figure 3.22: Web Application - Warning message when attempting to submit a modeling tool edit with no entry edits.

After submitting a valid modeling tool edit, the user is redirected back to the homepage

and thanked for the contribution (Fig. 3.17).

CHAPTER 4

Observations

While syntactical correctness and correctness of a system within modeling tools is irrelevant in the case of creating diagrams with modeling tools or business tools, this becomes the opposite in the case of diagrams expected to generate corresponding source code, be translated to other platforms or creating a functioning software. However there are very few modeling tools that manage to provide fully syntactical checks, which is why drawing tools such as diagrams.net are still a popular choice among experienced users, as it provides all the necessary templates, does not require login and all of its content is free.

There are however few efforts to curb this issue, for example the Language Server Protocol (LSP)[MB23], a protocol which intends to simplify the development of IDEs by dividing its components into language-specific servers and language-agnostic clients. This means that the servers are responsible for programming language-specific features, such as code completion, syntax highlighting etc., while the client simply provides frontend editing support. However, LSP only addresses textual languages and neglects graphical languages.

Both the research community [REIWC18b] and the open source community at Eclipse proposed to fill this void using the Graphical Language Server Platform (GLSP)[CLB22][BLO23], an extensible open-source framework for building custom diagram editors based on web technologies, allowing users to develop modeling tools similarly to editors which make use of LSP. Any developer can integrate the editor into web applications, but also into tool platforms such as Eclipse Theia and VS Code. GLSP uses the same client/server architecture, but since it also needs to take graphical challenges into account, the server also handles the tasks such as from moving from plain editing operations consisting of simple edit operations such as changing characters to complex operations such as connecting two nodes with an edge, thus establishing a relation between two objects [MB23]. Some of the tools listed in our surveys making use of GLSP include open-source

frameworks bigUML[1][MB23] and bigER[2][GB21][GHHB22]. One of the disadvantages of GLSP is that it is a relatively new framework, and thus there is bigger uncertainty among researchers and developers to create new web modeling tools using technology that is not yet established in the area of web modeling.

Another area that overlaps this topic is Model Driven Engineering (MDE)[BRDR+14][HWR14][PA18], an approach to software engineering with the goal of converting formal models into functioning software. MDE highlights and constructs abstract representations of the activities contained within an application domain, instead of computing concepts. One of the challenges MDE is faced however is that modeling tools are often separate software packages that do not provide support for reuse of existing modeling artifacts, therefore developers often build their own tools from scratch.

A further subset of MDE is blended modeling. As previously mentioned, the goal of MDE is engineering a system at high levels of abstraction before realization. To handle the complexity deriving from this approach that is faced with multiple formalisms, notations and computer mechanisms, blended modeling wants to provide an approach where engineers can freely choose between different notations for the same domain-specific concepts captured in a Domain-specific Modeling Languages (DSML).

Most modeling tools using syntax highlighting and validation use Domain-Specific Languages (DSL), which are used to specify domain models using textual concrete syntax. In comparison to general purpose programming languages (GPL) such as Java, DSLs are used to narrow down a problem area. Both LSP as well as MDE largely implemented using DSL and DSML.

While none of the aforementioned developments are yet fully implemented due to the sheer complexity deriving from graphical structures, there is still potential especially in software development. Future development could not only provide reliable code validation, but also be used for the generation of entire projects.

[1]bigUML: `https://github.com/borkdominik/bigUML`
[2]bigER: `https://github.com/borkdominik/bigER/blob/main/README.md`

CHAPTER 5

Related Work

Saheed Popoola et al.[PCG17] conducted a survey in the area of Model-Driven Engineering. As mentioned in 4 Observations, modeling tools are developed separately and therefore do not provide the possibility to reuse modeling artifacts. To solve this obstacle, Saheed Popoola et al.[PCG17] collected common functionalities among modeling tools based on which they propose the approach called as Modeling-as-a-Service (MaaS), a cloud-based modeling service.

Aníbal Iung et al.[NVN+12] also conducted a survey with the aim to gain independence from GPLs and develop a new DSL that offers higher level of abstraction and merges implementation concerns such as usability or security, to assist in the development of those concerns independently of the coding technology. However, implementing a new DSL is challenging and therefore Aníbal Iung et al.[NVN+12] identified and analyzed a total of 59 modeling tools, Language Workbenches or frameworks which can support language construction when creating a new DSL.

Istvan David et al.[DLP+22] searched for the most suited modeling tools in the area of blended modeling. The paper examined and found in total 26 modeling tools.

In the area of metamodeling, Miklós Maróti et al.[MKK+14] present WebGME, a web- and cloud-based (meta)modeling tool based on the Generic Modeling Environment (GME)[MBL07]. While GME allows users to defined new modeling languages using UML-based metamodels, WebGME provides further improvements such as collaborative modeling and analysis of large-scale information systems.

The use and development of web modeling tools as well as metamodeling tools goes beyond the need to create diagrams. Rodriguez-Echeverria et al.[REIWC18a] emphasize the challenges of syntax highlighting as well as validation in the development of modern IDE's in regards to textual and graphical languages. With the aim of simplifying new graphical modeling tools, the paper presents the approach of language-agnostic clients and language-specific servers, using LSP.

CHAPTER 6

Conclusion

In this paper we presented a survey of the current modeling tools, with our focus lying on web-based modeling tools which are emerging as the de facto standard when it comes to software aimed at constructing models and diagrams. As our research shows, there are currently many modeling tools already established in the market such as MS Visio from Microsoft, as well as new applications emerging from various developers and volunteer contributors. This paired with the fact that there are many aspects to be considered when developing a modeling tool, such as whether it merely serves the purpose of providing the functionality of combining any shapes in a smooth manner, or the tool is intended for engineers to collaboratively develop a model for their next project, this paper reflects that not any modeling tool is the same as another. The term "modeling tool" currently encompasses many applications, whereby many of those can be grouped into further subcategories based on common characteristics and intended usage, which has been shown in this paper.

We believe that our survey as well as our web application[1] could be of help to any user whose interest is to find modeling tools that meets his or her demands the closest. As we lay out all relevant details and properties which could be important in a users decision making, such as whether it is open source or if a user can reliable develop ER-diagrams, we provide an overview of the current modeling tools which under normal circumstances would have been found only through personal research or even personal testing, both of which require time investment which most users would preferably avoid.

Our web application[1] also provides the feature by the users of adding new modeling tools, as well updating existing ones. Furthermore, anyone can also provide feedback. Through user interaction we hope to extend our list in both existing tools as well as level of detail.

[1] List of Modeling Tools: `http://me.big.tuwien.ac.at/`

List of Figures

List of Tables

Bibliography

[BLO23] Dominik Bork, Philip Langer, and Tobias Ortmayr. A vision for flexibile GLSP-based web modeling tools. *CoRR*, 2023.

[BRDR+14] Francesco Basciani, Juri Rocco, Davide Di Ruscio, Amleto Di Salle, Ludovico Iovino, and Alfonso Pierantonio. Mdeforge: An extensible web-based modeling platform. In *Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud co-located with the 17th International Conference on Model Driven Engineering Languages and Systems, CloudMDE@ MoDELS 2014, Valencia, Spain, September 30, 2014*, volume 1242, 09 2014.

[CLB22] Giuliano De Carlo, Philip Langer, and Dominik Bork. Advanced visualization and interaction in glsp-based web modeling: realizing semantic zoom and off-screen elements. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, MODELS 2022*, pages 221–231. ACM, 2022.

[DLP+22] Istvan David, Malvina Latifaj, Jakob Pietron, Weixing Zhang, Federico Ciccozzi, Ivano Malavolta, Alexander Raschke, Jan-Philipp Steghöfer, and Regina Hebig. Blended modeling in commercial and open-source model-driven software engineering tools: A systematic study. *Software and Systems Modeling*, 06 2022.

[GB21] Philipp-Lorenz Glaser and Dominik Bork. The biger tool - hybrid textual and graphical modeling of entity relationships in VS code. In *25th International Enterprise Distributed Object Computing Workshop, EDOC Workshop 2021, Gold Coast, Australia, October 25-29, 2021*, pages 337–340. IEEE, 2021.

[GHHB22] Philipp-Lorenz Glaser, Georg Hammerschmied, Vladyslav Hnatiuk, and Dominik Bork. The biger modeling tool. In Sebastian Link, Iris Reinhartz-Berger, Jelena Zdravkovic, Dominik Bork, and Srinath Srinivasa, editors, *Proceedings of the ER Forum and PhD Symposium 2022 co-located with 41st International Conference on Conceptual Modeling (ER 2022), Virtual Event,*

Hyderabad, India, October 17, 2022, volume 3211 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

[HWR14] John Hutchinson, Jon Whittle, and Mark Rouncefield. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*, 89:144–161, 2014.

[MB23] Haydar Metin and Dominik Bork. On developing and operating glsp-based web modeling tools: Lessons learned from bigUML. In *Proceedings of the 26th International Conference on Model Driven Engineering Languages and Systems, MODELS 2023*. IEEE, 2023.

[MBL07] Zoltán Molnár, Daniel Balasubramanian, and Ákos Lédeczi. An introduction to the generic modeling environment. In *Proceedings of the TOOLS Europe 2007 Workshop on Model-Driven Development Tool Implementers Forum*. Citeseer, 2007.

[MKK+14] M. Maróti, Tamas Kecskes, R. Kereskényi, Brian Broll, Péter Völgyesi, L. Jurácz, T. Levendoszky, and Akos Ledeczi. Next generation (meta)modeling: Web- and cloud-based collaborative tool infrastructure. *CEUR Workshop Proceedings*, 1237:41–60, 01 2014.

[NVN+12] Leandro Nascimento, Daniel Viana, Paulo Neto, Dhiego Martins, Vinicius Garcia, and Silvio Meira. A systematic mapping study on domain-specific languages. In *The Seventh International Conference on Software Engineering Advances (ICSEA 2012)*, 11 2012.

[PA18] Parsa Pourali and Joanne M. Atlee. An empirical investigation to understand the difficulties and challenges of software modellers when using modelling tools. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, MODELS '18, page 224–234, New York, NY, USA, 2018. Association for Computing Machinery.

[PCG17] Saheed Popoola, Jeffrey C. Carver, and Jeffrey G. Gray. Modeling as a service: A survey of existing tools. In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2017.

[REIWC18a] Roberto Rodríguez-Echeverría, Javier Luis Cánovas Izquierdo, Manuel Wimmer, and Jordi Cabot. An lsp infrastructure to build emf language servers for web-deployable model editors. In *MoDELS*, 2018.

[REIWC18b] Roberto Rodriguez-Echeverria, Javier Luis Cánovas Izquierdo, Manuel Wimmer, and Jordi Cabot. Towards a language server protocol infrastructure for graphical modeling. In *Proceedings of the 21th ACM/IEEE*

International Conference on Model Driven Engineering Languages and Systems, MODELS '18, page 370–380, New York, NY, USA, 2018. Association for Computing Machinery.

[Top] Top online uml modeling tools (also including web-based tools for er and bpmn diagrams). https://modeling-languages.com/web-based-modeling-tools-uml-er-bpmn/. Accessed: 2022-10-01.

[Wik] List of unified modeling language tools. https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools. Accessed: 2022-10-01.