

# Automated Generation of Web Application Front-end Components from User Interface Mockups

Oksana Nikiforova<sup>a</sup>, Kristaps Babris and Farshad Mahmoudifar

*Institute of Information Technologies, Riga Technical University, 6A Kipsalas Street, Riga LV-1048, Riga, Latvia*

**Keywords:** User Interface Mockups, Front-End Components, Generation of User Interface Elements, User Experience Automation.

**Abstract:** In modern web development, the design-to-code process is a critical bottleneck, often characterized by inefficiencies and inconsistencies between initial design concepts and implemented user interfaces. Bridging this gap demands innovative solutions that streamline the translation of design artefacts, such as wireframes/mockups, into functional front-end components. Generation of user interface (UI) elements has also been in the spotlight of researchers all along. There are several solutions to support the transformation of different automation processes and elements, but the fact that common methods and tools for generating UIs are still not widely used shows that this problem has not been solved yet. In this paper, we propose model-driven approach to automate the generation of front-end source code from wireframe representations and, thereby facilitating rapid prototyping and enhancing collaboration between designers and developers. The paper offers the conceptual solution for front-end components generation from the mockups developed in the UI design tools, where it is possible to have formal definition of the UI elements source code. The offered solution is approbated on the abstract practical example, where mockups designed in the tool draw.io are used as a source for generation of front-end components of the web application deployed in AngularJS framework.


## 1 INTRODUCTION

Since 2020, digitalisation has demonstrated its need to the modern society (Horgan, et al., 2020). As everyday life changes, the society is more and more ready to use and accept various technological benefits that could help to perform routine tasks more efficiently, as well as increase quality - by automating processes. In this context, COVID-19 acted as an accelerator, growing the forced use of various systems and tools and proving that quick solutions can solve tactical tasks (Amankwah-Amoah, 2021). Despite the fact that many organizations have found it difficult to digitize their processes in a hurry and not all of them have been able to do so, digitization has proved its worth and its progress is now unstoppable.

At the current stage of digitalisation, there is a great demand for IT solutions in business areas such as insurance, tourism, etc. Automation, as an element of digitization, is needed at all stages of software development - from documentation creation and

software requirements generation, software artefact generation, quality aspects, user interface, etc. All of these issues are being researched and brought to the attention of scientists, and solutions and tools are being developed to address these issues and to increase efficiency (Nikiforova, et al., 2021).

The development process for web applications often involves a significant investment in time and resources dedicated to front-end development. This stage translates visual design elements, typically captured in wireframes, into functional user interfaces (UI) using code. While various UI frameworks and libraries streamline front-end development, manually writing code for each UI component remains a time-consuming task. Creating UI from the initial design and providing the features and components that the stakeholder and designer agreed on for the application is a challenging and time-consuming process of development process. This can be even more challenging when this conversion involves a structure that needs to be incorporated into the final application. This paper explores the potential of

<sup>a</sup> <https://orcid.org/0000-0001-7983-3088>

automating front-end component source code generation from mockups. The aim of this paper is to introduce an efficient approach the solution to converting UI mockups and mockups into structure code to build a web application. This article focuses on the automatic generation of a web application front-end components from user interface mockups, which means that we want to automatically generate source code of user interface components from sketches or prototypes of user experience of the business environment, based on a notation or model defined in the business field and resulting in a sketch model that will consist of certain elements.

The structure of the paper is the following – the next section looks into the background and related work performed in the area of user interface design automation. The third section describes a solution offered by authors. We discuss the technical challenges involved and propose an approach that utilizes benefits from model-driven engineering formally translating mockups into corresponding web application front-end components source code. The fourth section demonstrates an application case of the approach, where the offered solution is used and its potential benefits and limitations discussed. The conclusion section gives some discussion about the perspectives of the research.

## 2 BACKGROUND AND RELATED WORK

The User Interface (UI) plays a critical role in modern software development, significantly impacting user experience and satisfaction (Lallemand, Gronier, & Koenig, 2015). A well-designed UI not only attracts users but also enhances engagement, while a poor design can lead to frustration and disengagement (Djamasbi, Siegel, & Tullis, 2010). To excel in competitive markets, developers and designers must prioritize effective and appealing UIs. As the demand for digital products rises, delivering high-quality products that captivate users attention becomes imperative (Jokela, Ojala, & Olsson, 2015). UI design extends beyond web browsers, encompassing various devices with different user experiences, such as mobiles and touch-screen devices.

Considering users requirements and involvement during UI design is crucial for enhancing user satisfaction and efficiency (Norman, 2013). Usability is another vital aspect, ensuring users can accomplish tasks easily and efficiently (Nielsen, 2012). Factors like simplicity, speed, memorability, error

prevention, and enjoyment contribute to user engagement (Nielsen, 2010). Accessibility is also essential, ensuring that all users, including those with disabilities, can use the application (Henry, et al, 2014). Following accessibility guidelines like Web Content Accessibility Guidelines (WCAG) facilitates a comprehensive user experience. Finally, prioritizing user needs and staying updated with design trends are crucial for delivering exceptional UIs in modern applications development (Norman, 2013).

The rapid evolution of software development has led to the adoption of advanced technologies to efficiently meet customer expectations and enhance functionality. UI mockup to code conversion has emerged as a method to streamline production and minimize manual coding efforts. The approach presented by Bouças and Esteves (2020) utilizes deep machine learning to generate HTML and CSS from UI mockups created with the Balsamiq application. The method involves creating a dataset of common components, detecting HTML elements, and converting them to code using hybrid and YOYO algorithms. While the hybrid model yields slightly different results from the initial design, the YOYO approach demonstrates higher accuracy in identifying objects. Similarly, Moran et al. (2020) propose a method combining computer vision and machine learning to generate applications from GUIs. This approach involves detecting GUI components, classifying them using machine learning algorithms, and assembling the application based on component hierarchies and styles. However, complexities and image quality issues can compromise accuracy and consistency. In another study, Lopez (2020) evaluates three tools for converting UI mockups to HTML using image processing and machine learning. The quality of the designed UI image significantly influences the output results, highlighting the importance of image quality and tool selection in this approach (Behailu, 2020).

The web application development industry has been developing software that needs to be automated since its inception. Typically, components of web application are created manually, in parallel with the research and development of user interface wireframes, which is possible to design in special prototyping tools. In recent years, advancements in software development have necessitated the utilization of automation techniques to enhance efficiency. Among the challenging aspects of this process is UI design, prompting developers and researchers to explore new methods for converting initial sketches or UI prototypes into code.

The UI development process typically begins with the creation of wireframes, mockups, or prototypes using various UI design tools such as Figma, to visualize and design the application interface (Staiano, 2022; Mahendra, 2021). These tools enable designers to create layouts, schemas, and UI elements like buttons and text boxes, with Figma being a popular choice due to its extensive features and flexibility. However, while Figma lacks direct source code export capabilities, plugins can be installed for this purpose, albeit with some limitations.

Draw.io is another widely used tool for UI mockup design, offering user-friendly features and the ability to export designs to various formats (Mahendra, 2021). Despite these tools capabilities, there remains a gap between UI designers and developers, leading to discrepancies between the initial design and the final UI. Designers focus on aesthetics, while developers prioritize functionality and performance, resulting in inconsistencies and inaccuracies. Automating this process can mitigate errors arising from such discrepancies (Chen et al., 2020).

The challenge lies in bridging the gap between designers and developers, ensuring accurate translation of UI designs into code (Myers et al., 2008). Collaboration is essential to clarify ambiguities and ensure alignment between design vision and implementation. By automating parts of the UI development process, errors due to inconsistencies can be reduced, leading to improved accuracy and consistency in the final application UI.

UI prototyping encompasses two types: mockups and wireframes (Uzayr, 2022). Mockups are high-fidelity graphical representations of the final product, detailing components and elements with colours, but lack interactivity. Conversely, wireframes are low-fidelity designs focusing on UI structure, created using basic shapes and lines, without visual design details (Chen et al., 2020).

Both mockups and wireframes play a crucial role in UI design, aiding designers in presenting their product and enabling stakeholders to provide feedback before coding begins. This iterative process ensures customization according to stakeholders specifications and user expectations.

However, converting UI mockups to UI designs and code can be challenging due to ambiguous definitions and potential information loss. Unlike Unified Modelling Language (UML), which uses variables and entities for abstraction, mockups rely on real-life data scenarios, increasing the risk of missing information during conversion (Urbietta et al., 2018). To address this, modeling is essential to facilitate a

smooth and accurate conversion process. According to these model-driven approaches, modelling and metamodeling (which is the description of the models) aid the UI development process in terms of increasing accuracy as well as providing an efficient conversion process. This also brings reusability and optimization to the project which is an advantage for the developers. Metamodels serve as mediators between application design and corresponding code, enabling designers to describe all elements of the application at the design level, ensuring understanding by developers or automated tools (Nikiforova, Gusarovs, 2020).

Rivero et al. (2010, 2011) introduced a model-driven approach that transforms UI mockups into models or codes using metamodels and abstract models. These models specify content, navigation, and structural UI elements, facilitating easier recognition for coding or modeling purposes. Saputra and Azizah (2013) presented a solution using metadata models to generate code from UI elements, simplifying the process of modifying web applications. The idea to use model-driven formalisms in UI automation was promising and model-driven approach enhances maintainability and flexibility in UI development. Model-driven development provides a systematic and structured framework for designing, implementing, and managing UI components. By representing UI elements and behaviours in abstract models, developers can achieve a higher level of clarity, consistency, and maintainability throughout the development process. This approach streamlines collaboration between designers and developers, ensuring that the final UI aligns closely with design specifications, enabling developers to create UI components that can be easily adapted and extended across different applications and platforms. By abstracting UI implementation details into models, developers can build flexible and modular UI architectures that accommodate changing requirements and evolving technologies. By defining UI specifications in models, developers can enforce consistency and compliance with usability principles, accessibility standards, and design patterns.

According to system theory, any solution to the problem can be represented as a function that depends on some input data and outputs certain output data. User interface components generation approaches which exists today can be systematized in way that there are some input data (text, picture, mockup, wireframes, etc.) which is needed to produce desired output data (HTML, XML, CSS, application code, etc.) using algorithms to do that.

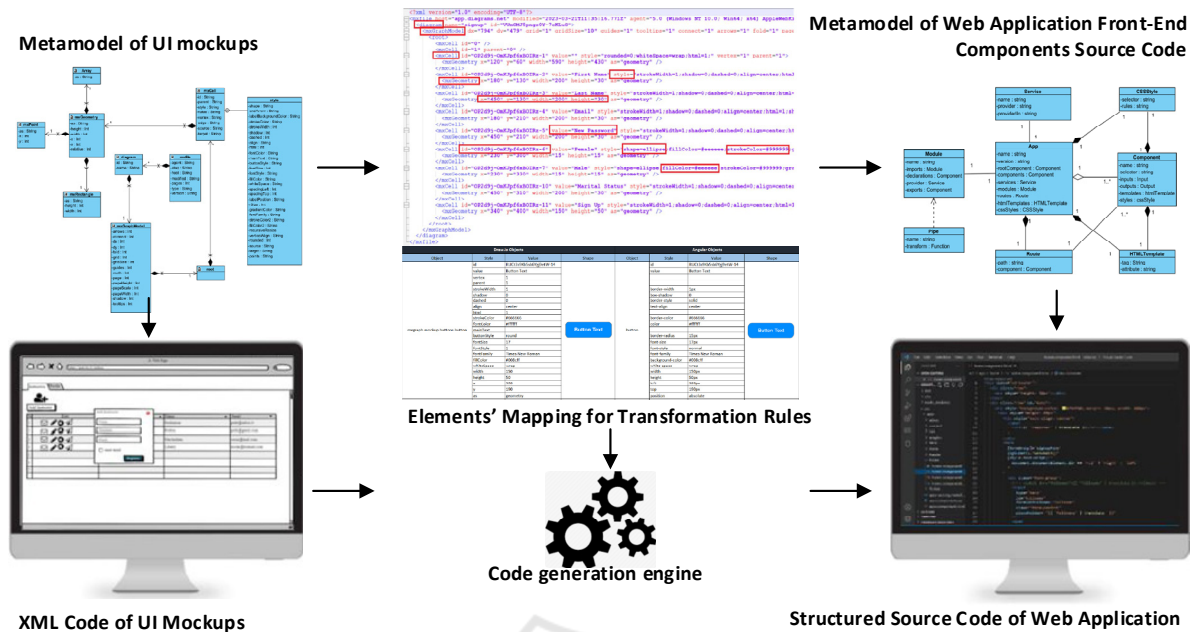


Figure 1: Solution concept.

The analysis of related researches demonstrates that there are no fully automatic approach existing to support full automation chain, because there are problems with input data, proceeding data (have their flaws) or the algorithms which should process input data to output result have their dependencies, specific format needs or must include manual work.

### 3 SOLUTION CONCEPT

The offered solution concept how to generate source code of web application components from designed user experience mockups is shown in Figure 1.

#### 3.1 Source Model

The authors have introduced a method for generation a cohesive final structured web application that integrates all components and their interrelations from the initial UI design in a seamless manner. This strategy leverages the UI mockups source code for the conversion process, ensuring accessibility through programming languages like XML. An automated code generator based on the predefined transformation rules utilizes the XML code of the designed mockups to fabricate the structured web application components based on designated programming languages or frameworks such as Angular, React, and JavaScript.

A model serves as a simplified representation of a system, aiding in understanding and analyzing its logic and design (Knuutila, 2011; France & Rumpe, 2007). Modeling involves simplifying real-world situations to reveal system structure and behavior, facilitating system requirement identification during software development (Stevens, 2000; Seidewitz, 2003; Leimane, Nikiforova, 2019).

In software engineering, metamodels provide guidelines and descriptions for models, serving as a lower-level guide for higher-level models (Seidewitz, 2003; Lumertz, Ribeiro, & Duarte, 2016). Metamodeling offers benefits such as simplifying computation, filtering noise, rendering design space, and error detection (Wang & Shan, 2007).

Model-driven approaches in UI development aim to bridge existing gaps. Rosado da Cruz and Faria (2010) introduce a model-driven development approach using metamodeling to automatically generate UI models for interactive applications. However, limitations such as complexity and limited scalability exist. Similarly, Lumertz, Ribeiro, and Duarte (2016) propose a graph-based metamodel for UIs, representing UI components and their relationships as graph nodes and edges. They identify patterns among UI elements and define control mechanisms for user interaction, along with standards for managing data operations.

Metamodels act as a powerful tool in all stages of the development process and provide an efficient method for automating the different stages in which

UI development can get benefit from employing this tool.

A metamodel is established, delineating components, elements, and their relationships as depicted in the UI wireframe, thus specifying and presenting the UIs structure alongside component specifications. This involves analyzing the source code generated from the UI design application to extract and identify objects recognizable as components within the standard UI metamodel, which will be elaborated upon subsequently. The metamodel of UI mockups is shown in Figure 2.

The establishment of a standard UI metamodel is imperative to encompass nearly all requisite components and elements (text boxes, buttons, menus), and others—for web application development. The UI metamodel yields a set of transformation rules utilized during code generation, effectively forming a dictionary of UI components for conversion. Consequently, comparing the initial

metamodel derived from the UI wireframe with the standard UI metamodel is essential to construct a comprehensive dictionary for recognition within the programming language.

### 3.2 Target Model

A key factor in modelling and metamodeling is the transformation of the models which is the process of converting a model to another model at the same level or transforming to different levels in a system (Czarnecki and Helsen, 2006). Model transformation can be applied by a series of transformation rules to refine the model of the platform, transform a model into another model, or generate code of the application UI from the model. In the context of generation of web application components, the metamodel of front-end components required by AngularJs is shown in Figure 3.

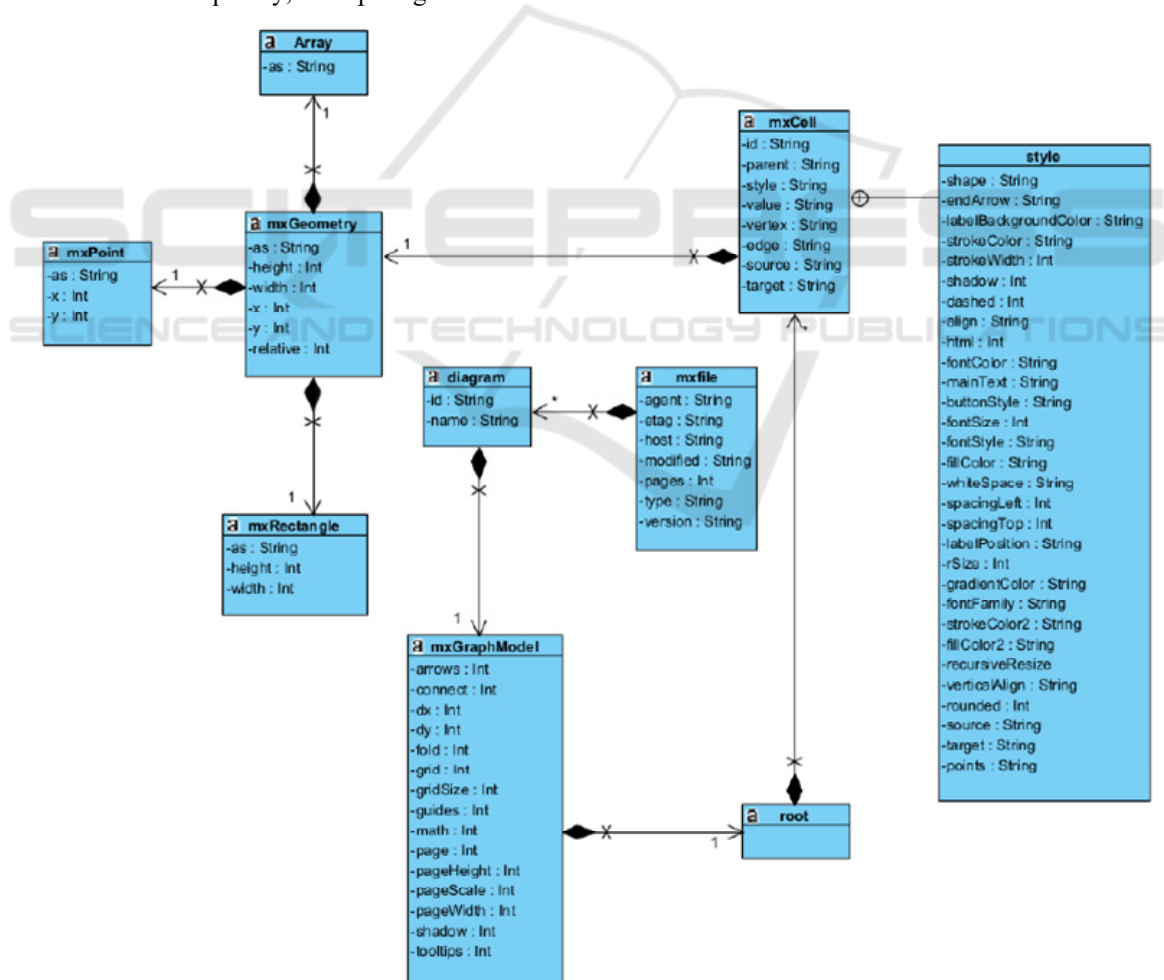


Figure 2: Meta-model of UI Mockups.

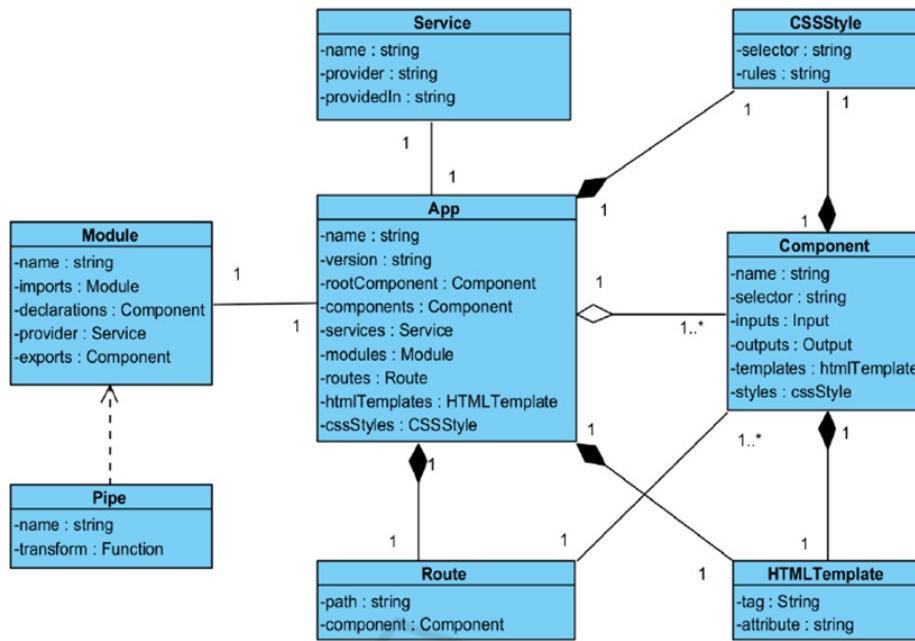


Figure 3: Meta-model of Web Application Components for AngularJS.

Data collection, analysis, and interpretation are required for this research process in this methodology which creates a metamodel from the UI mockup to identify the structure of the UI and determine the components specifications. This metamodel is being made based on analysing the source code of the UI that is exported from the UI design software to identify the objects as the components as well as the structure of the source code. A standard metamodel is required according to the selected code or framework to present the final code structure along with the elements and components required to create a web application. The next step is the comparison of this metamodel with the UI design metamodel for the purpose of creating transformation rules for components as well as its structure that is recognizable by programming languages. Finally, an automatic code generator generates a web application from the identified elements, components, and their relations using these transformation rules.

According to the system architecture, in order to create the transformation rule, it was required to perform a comparison between the standard metamodel, which is the designated code or framework, and the UI design metamodel to map the data. As the second requirement for creating the transformation rule, the standard model for the designated code or framework needs to be created (El Marzouki, et al., 2017). This has been done by identifying the main objects of code and its structure to find out about their connection. This research has

used Angular as the designated framework for creating a metamodel and prototype. The author created the metamodel based on the information provided on the Angular website ([www.angular.io/docs](http://www.angular.io/docs)) and the knowledge and experiences of the author in programming in the Angular framework.

The steps for creating the metamodel includes identifying the main objects according to the Angular structure and then finding their attributes according to their type. These objects were classes for the metamodel. When all the classes with attributes were defined, the connections and interactions with other classes were observed, and these connection types were implemented in the metamodel to reach the final design.

### 3.3 Transformation Rules

The automated code generator employs transformation rules to discern UI elements and relationships, facilitating the generation of the final code and execution of necessary dependencies to structure the exported code akin to the initial UI design within the designated framework.

For transformation rules definition the exported source code of the UI mockups was examined and analysed based on the recognition of the elements and components in order to be able to transfer them to the web application components (see Figure 4). This phase also requires employing a parser to parse the

initial data to organize data like JSON for the purpose of readability and ease of use for the code generator.

The author used a live mapper to apply the changes happening on the model to the frontend component long with wireframe. This provides the ability to change the wireframe by changing the components of the front end. This is a way to handle a complex method by updating the code and model at the same time employing the model-driven approach as well as the transformation of models to reach their desired outcomes. In this way, the model transformation is able to facilitate UI development as well as reduce the risk of inconsistencies for the conversion.

When the source code was analysed and interpreted it was necessary to create a table of the content of the elements, objects, and any information that helps us in creating a dynamic web application, like the connection between the objects. To support different platforms, it would be important to export all elements with their default values even if we could not identify them based on the common web application elements. This helped the process in creating a metamodel of the components in the next phase which is the metamodeling phase.

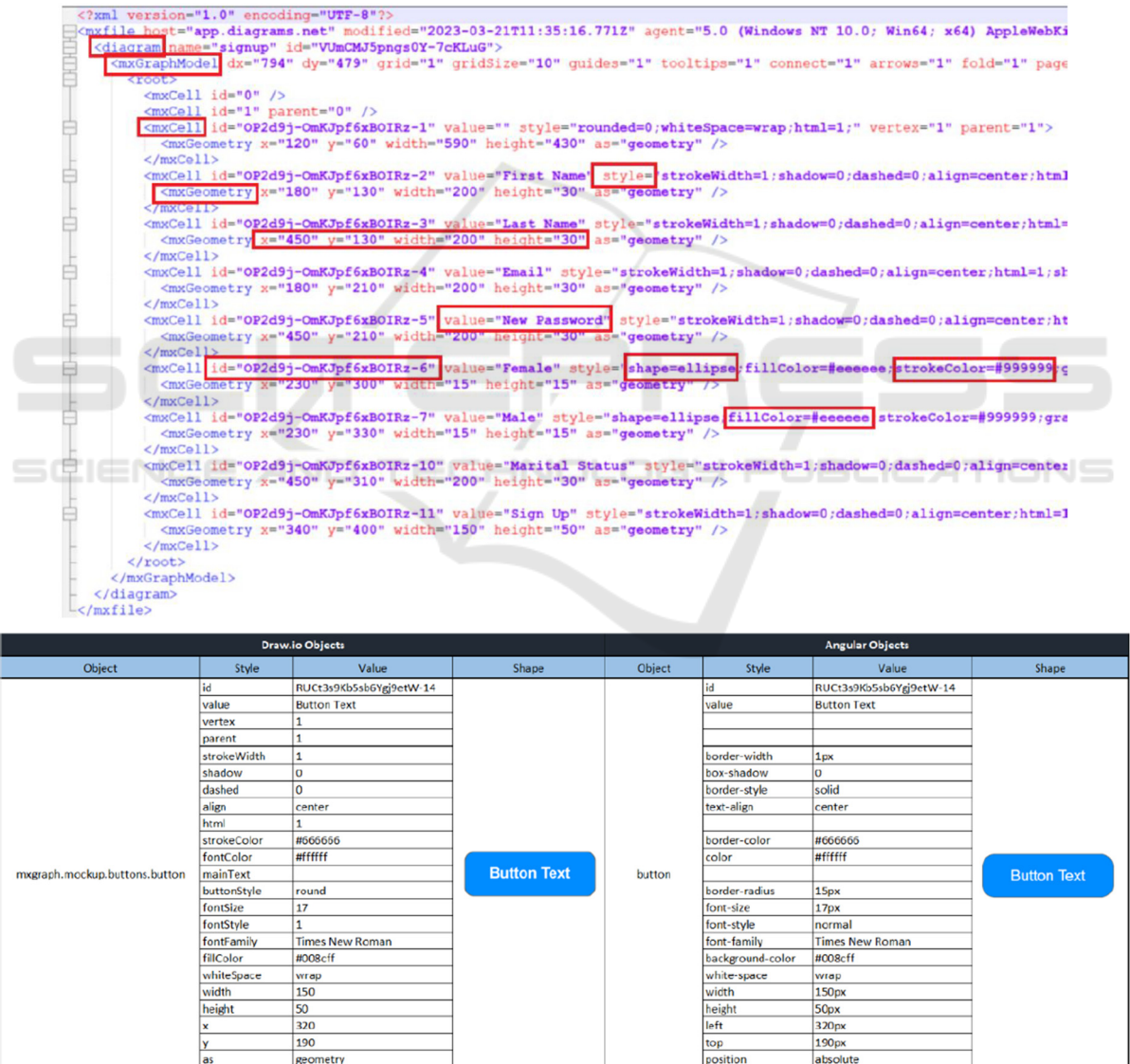


Figure 4: An example of the object identified in UI mockups XML code versus elements required on Web application in AngularJS.

## 4 APPLICATION CASE

This proposed approach streamlines web application development, economizing developers time and effort. Moreover, the resulting output closely resembles the original UI design in structure, fostering accuracy and consistency throughout the development process (de Lange et al., 2020).

In the metamodeling phase, a metamodel is created on the basis of the objects, and relations among them can be done manually or with modelling software that exists for this purpose. The next which is creating the metamodel of the designated output code or framework which can also be done manually, employing existing software or using the already made metamodels in studies. This metamodel plays a significant role in creating transformation rules due to the increase in the accuracy and reliability of this conversion and creates by comparing the designated code metamodel as the standard model with the initial UI metamodel. The transformation rules work as a dictionary for the code generator to identify the object in the UI source code and generate the equivalent code in the designated code or framework.

In terms of feasibility check and making sure that two meta-models are able to be converted to web application source code like HTML, it was necessary to find the equivalent component for each object, The UI mockups are designed using draw.io tool with further export of the elements in XML file and parsing it to json objects. And for web application AngularJS framework was selected. Since most of the elements used in Angular are the same as those in HTML, this can be almost identical for frameworks or codes using HTML such as React and JavaScript, except for the elements referring to the structure of the code.

The metamodels were used to identify the source and destination of elements and the catalogue of objects was employed to create the final code manually to observe and modify the output to reach the expected result. The result should have the same structure as the UI design source code such as components and their containment which is the elements with their specifications and styles. A snapshot of the final output for some of the elements as the mapped code presented in Figure 5 which this mapping for all defined elements accessible in the transformation rules file on the GitHub repository (Mahmoudifar, 2023a,b).

Based on the provided code blocks authors were able to create the web application manually quite similar to the initial design which shows the possibility of this approach which is generating a web application from UI mockup source code with all structure and elements on the initial design. This process of identifying the elements, creating the designated code then installing dependencies, and in the final creating the structure of the web application based on the source code can be done automatically by an automatic code generator which is in phase three. In continuation of the main process, when the objects are identified and metamodels are created, the next step is extracting the transformation rules by comparing the current metamodel with the standard UI metamodel to reach the final code acquired manually, this is being used by the code generator. These transformation rules include mapping and transforming elements and their relations to a final code. This makes it easier for the programming language to recognize the element for generating the code by using it as a dictionary of UI components and elements of the web applications (Batdalov, Nikiforova, 2018).

| Object in Draw.io | Object in Angular              | Source Code in Draw.io  | Source Code in Angular  |
|-------------------|--------------------------------|---|---|
| diagram           | component<br>Module<br>Routing | <pre>&lt;diagram name="Page1" id="0apm80VhK-hvCSe2Skn2"&gt;   &lt;mxGraphModel dx="794" dy="530" grid="1" gridSize="10"   guides="1" tooltips="1" connect="1" arrows="1" fold="1" page="1"   pageScale="1" pageWidth="850" pageHeight="1100" math="0"   shadow="0"&gt;     &lt;/mxGraphModel&gt;   &lt;/diagram&gt;</pre> | <pre>Component: import { Page1Component } from './page1/page1.component'; @Component({   selector: 'app-page1',   templateUrl: './page1.component.html',   styleUrls: ['./page1.component.css'] }) export class Page1Component { }  Routing: const routes: Routes = [{ path: 'Page1', loadChildren: () =&gt; import('./page1/page1.module').then(m =&gt; m.Page1Module) }];</pre> |

Figure 5: The output code for some elements.



In the direction of reaching a transformation rules that work as a dictionary used by the code generator to generate the code from the UI mockup source code, a block of code should be created to do this conversion. These code blocks should be unique for each element or specification that exists in the UI design source code to generate the equivalent element in the final code. By comparing the table provided, which converts each object in Draw.io to HTML elements, which is part of the Angular structure, it was possible to provide conversion code blocks, for example, for a button to be recognized in the source code and identify the style and specification to create the similar button in the final code.

This can be done for almost all objects in the source code depending on the technology that is going to be used as the final code or framework. In this research, Angular as the designated code covers the HTML components in the UI design source code, and also other information like page and relation among elements or even routings can be extracted from the code. This transformation rules are used by code generator to transform the identified elements in UI mockup source code to the application UI elements.

In the last phase, the code generator which works as an automatic code generator installs the initial requirements, dependencies, and libraries required for the web application. Afterward, this code gen According to Lumertz, Ribeiro, and Duarte (2016), UI views are generated based on identified patterns, illustrating relationships among basic elements. Utilizing a graph as the UI structure, source code can be generated from the extracted UI model as part of the metamodel after adding attributes and constraints. Although not part of this thesis process, employing metamodeling and graphs in UI development can enhance the process efficiency (Lumertz, Ribeiro and Duarte, 2016).

Generator takes the simplified JSON file which is the analysed components and elements, using the transformation rule as the dictionary to generate the codes, features, routings, and any other requirements for creating the web application UI and then running the application.

To validate the approach, an evaluation was conducted to ensure the system functions as intended, addressing potential weaknesses and areas for improvement. The goal was to demonstrate that the prototype effectively addresses research gaps and offers practical solutions to real-world problems.

The evaluation encompassed various aspects to assess both strengths and limitations of the method. Beyond merely transferring UI objects to elements, the approach constructs the front-end application

structure based on UI design source code, validating the entire transformation process and its outcomes.

Evaluation criteria included:

- **Consistency and accuracy:** Ensuring that generated code elements accurately reflect the original designs properties, such as type, shape, and color.
- **Performance and maintainability:** Efficiently identifying all XML objects and generating code, with consideration for processing time and code structure. The generated code should be easily maintainable to accommodate UI design modifications.

To assess accuracy and consistency, an image comparison tool, Beyond Compare (www.scootersoftware.com), was utilized. This tool juxtaposes UI mockup and application UI images, highlighting similarities and differences pixel by pixel. By visually comparing the output design with the original, the accuracy and coherence of the output can be evaluated effectively. This tool configures to illustrate the images side-by-side along with a view of the overlying of two images and compares them pixel by pixel. which is used to compare different data files as well as images. The differences are presented with colours and symbols on the overlaid images as shown in Figure 6 for the signup page and for the login page.

The comparison results indicate that all elements from the UI mockup were successfully transferred to the application UI, retaining specifications and styles. Minor differences in element placement and layout, particularly affecting text elements, were observed but deemed insignificant. Notably, the combo box exhibited a different style due to varying default styles between Draw.io and HTML.

Statistical analysis revealed a high degree of similarity between the registration and login pages and their original designs, with similarity percentages of 94% and 93.4%, respectively. This assessment was based on pixel-by-pixel comparison.

To assess the generated Angular code, a code review was conducted, focusing on structure and syntax. The code structure adhered to Angular guidelines, with the main app component and separate components for signup and login pages. Each component comprised HTML, CSS, and TypeScript files, facilitating functionality and styling.

The HTML code accurately reflected the UI elements, generated based on transformation rules and code generator design. However, certain elements like grid, div, and span were not defined in Draw.io, leading to their absence in the output HTML code.

Figure 6: Comparison result for signup page by Beyond Compare.

CSS styles, derived from element IDs, determined element positioning on the HTML page. Notably, absolute positioning was used, detracting from modern web design principles advocating for division and span-based layouts.

Overall, while the generated code adhered to Angular standards and accurately reflected UI elements, improvements are needed to align with

contemporary web design practices, particularly regarding element positioning.

The authors of this paper suggest that employing metamodeling in UI development can improve the process and reduce errors, as evidenced by similar studies. In the study by Dimbisoa, et al. by (2018), the aim is to describe UI generation using metamodels to achieve functionality in components and source code. This approach involves model-driven development to create an abstract model based on the real model, facilitating the transformation process for easier understanding.

Every approach has some limitations and this approach would not be free from limitations. The authors of the paper have identified some factors that might affect this process result such as the complexity of the designed UI, the UI design software, the transformation rules, and the code generator. In general, these factors can be categorized by, the definition and information of the objects in the UI design source code and the design accuracy of the UI transforming section, which can be compromised by complex UI designs.

One more aspect is its accuracy hinges on the precise construction of the UI wireframe. Inaccuracies in construction may lead to imprecise or incomplete code, rendering the method unsuitable for complex web applications requiring extensive customization and versatility.

To recap, the proposed approach presents a promising solution for the challenge of converting UI wireframes into code. A standard UI metamodel serves as a blueprint for structuring components within web applications, ensuring the production of precise and reliable code. Nonetheless, further research is warranted to evaluate its efficacy and limitations across varied scenarios and to validate its reliability.

## 5 CONCLUSIONS

The main hypothesis of this paper is that it is possible to use a metamodel to define the structure of UI mockups by identifying components and elements and their relationships of potential application in order to create transformation rules for source code generation of the application front-end components in predefined programming language. This helps to increase the accuracy and consistency of the automatic code generation based on the provided transformation rules. As a result, it saves time and effort for the developers by automating repetitive

tasks and ensuring that the code is complete and consistent to design specifications.

The research presented in this paper introduces a methodology leveraging metamodeling techniques to dissect the UI structure, identify components, and establish transformation rules for seamless conversion. Through an exploration of existing literature, the pivotal role of metamodeling in UI development is underscored, demonstrating its potential to enhance efficiency, adaptability, and accuracy in both design and code generation processes. By employing metamodeling, the mapping process is refined, leading to the creation of precise transformation rules for UI conversion. Moreover, utilizing source code instead of image processing offers comprehensive insights into design structure, while automated code generation bolsters productivity and minimizes errors.

The proposed research design entails three phases: source and target models analysis, metamodeling, and code generation. Initial data collection involved scrutinizing export source files from prominent UI design applications to identify design components. Subsequently, metamodels for UI source code and designated frameworks were constructed, enabling the extraction of transformation rules based on established relationships. The final phase saw the implementation of an automated code generator to produce application UI based on the analyzed data and transformation rules.

In order to explore the generation of front-end component source code we have investigated the feasibility of model-driven principles, which were actual a decade ago, but still are suitable in the tasks of web development automation and gives an ability to bridge the gap between design and development. As the result, the solution reduces development time, because by generating code from wireframes, developers can focus on complex functionalities and business logic. As well as automation can free developers from repetitive tasks, allowing them to invest their expertise in higher-level aspects of application development. Moreover, the automating code generation can bridge the gap between designers and developers by providing a common ground for communication and iteration. During the research, limitations arose due to difficulties in exporting source code from various UI design applications. Draw.io was identified as a potential solution, yet it also had limitations, as outlined in subsequent sections. Despite these challenges, the proposed solution shows promise in transforming UI development processes. Future research will explore

leveraging Figma, a UI design tool with extensive plug-in integration capabilities.

## REFERENCES

- Amankwah-Amoah, Joseph et al. (2021) "COVID-19 and digitalization: The great acceleration." *Journal of business research*, vol. 136, 602-611. doi:10.1016/j.jbusres.2021.08.011.
- Batdalov R., Nikiforova O. (2018) "Three patterns of data type composition in programming languages." ACM International Conference Proceeding Series, DOI: 10.1145/3282308.3282341
- Behailu, B. (2020) Automatic Code Generation from Low Fidelity Graphical User Interface Sketches Using Deep Learning. Thesis: <http://ir.bdu.edu.et/handle/123456789/11292>
- Bouças, T. and Esteves, A. (2020) Converting Web Pages Mockups to HTML using Machine Learning. DOI:10.5220/0010116302170224.
- Chen, J. et al. (2020) Wireframe-based UI Design Search through Image Autoencoder, *ACM Transactions on Software Engineering and Methodology*, 29(3), p. 19:1-19:31. DOI:10.1145/3391613.
- Czarnecki, K. and Helsen, S. (2006) Feature-based survey of model transformation approaches, *IBM Systems Journal*, 45(3), pp. 621–645, DOI:10.1147/sj.453.0621
- Dimbisoa, W.G., Mahatody, T. and Razafimandimby, J.P. (2018) Creating a metamodel of UI components in form of model independant of the platform, 6(2), p. 5.
- Djamasbi, S., Siegel, M. and Tullis, T. (2010) Generation Y, web design, and eye tracking, *International Journal of Human-Computer Studies*, 68(5), pp. 307–323. DOI:10.1016/j.ijhcs.2009.12.006.
- Draw.io, Diagram drawing tool, <https://app.diagrams.net>
- Figma, The Collaborative Interface Design Tool, <https://www.figma.com>
- Henry, S.L., Abou-Zahra, S. and Brewer, J. (2014) The role of accessibility in a universal web, Henry [Preprint]: <https://dspace.mit.edu/handle/1721.1/88013>
- Horgan D, Hackett J, Westphalen C, B, Kalra D, Richer E, Romao M, Andreu A, L, Lal J, A, Bernini C, Tumiene B, Boccia S, Montserrat A. (2020) Digitalisation and COVID-19: The Perfect Storm. *Biomed Hub*. 2020;5:1-23. doi: 10.1159/000511232.
- Jokela, T., Ojala, J. and Olsson, T. (2015) A Diary Study on Combining Multiple Information Devices in Everyday Activities and Tasks, p. 3912, DOI:10.1145/2702123.2702211.
- Knuuttila, T. (2011) Modelling and representing: An artefactual approach to model-based representation, *Studies in History and Philosophy of Science Part A*, 42(2), pp. 262–271. DOI:10.1016/j.shpsa.2010.11.034.
- Lallemand, C., Gronier, G. and Koenig, V. (2015) User experience: A concept without consensus? Exploring practitioners perspectives through an international survey, *Computers in Human Behavior*, 43, pp. 35–48. DOI: 10.1016/j.chb.2014.10.048.

- de Lange, P. et al. (2020) Integrating Web-Based Collaborative Live Editing and Wireframing into a Model-Driven Web Engineering Process, *Data Science and Engineering*, 5. : DOI:10.1007/s41019-020-00131-3.
- Leimane., L., Nikiforova., O. Results from Expert Survey on System Analysis Process Activities, *Applied Computer Systems*, 2019, Vol. 24, Issue 2, pp. 141.-149., DOI:10.2478/acss-2019-0018
- Lopez, B.S. (2020) Automatic Generation of Synthetic Website Wireframe Datasets from Source Code. Do Porto: <https://repositorio-aberto.up.pt/bitstream/10216/128542/2/412407.pdf>
- Lumertz, P.R., Ribeiro, L. and Duarte, L.M. (2016) User interfaces metamodel based on graphs, *Journal of Visual Languages & Computing*, 32, pp. 1–34. : DOI:10.1016/j.jvlc.2015.10.026.
- Mahmoudifar, F. (2023a) Solution for Generation of Web Application User Interface Components From User Interface Mockup Source Code, Master thesis, Riga Technical University, Latvia
- Mahmoudifar, F. (2023b) Source code repository for the solution for generation of web application UI components from UI mockups: <https://github.com/fmahmoudifar/uitocode>.
- El Marzouki, N., Nikiforova, O., Lakhrissi, Y., El Mohajir, M. (2017) Toward a Generic Metamodel for Model Composition Using Transformation. *Procedia Computer Science*, 2017, Vol.104, 564.-571.lpp. ISSN 1877-0509. Pieejams: doi:10.1016/j.procs.2017.01.173
- Mohamad Yusril Aldiana Mahendra (2021) The Use of Draw.io as Digital Mind Map to Improve Students Creativity and Students Concept Mastery in Learning Human Influence on Ecosystem. Universitas Pendidikan Indonesia. : <http://repository.upi.edu>
- Moran, K. et al. (2020) Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps, *IEEE Transactions on Software Engineering*, 46(2), pp. 196–221. DOI:10.1109/TSE.2018.2844788.
- Myers, B. et al. (2008) How designers design and program interactive behaviors, p. 184. DOI:10.1109/VLHCC.2008.4639081.
- Nielsen, J. (2010) CHAPTER 1 - What Is Usability?, in C. Wilson (ed.) *User Experience Re-Mastered*. Boston: Morgan Kaufmann, pp. 3–22. DOI:10.1016/B978-0-12-375114-0.00004-9.
- Nielsen, J. (2012) Usability 101: Introduction to Usability, Nielsen Norman Group. : <https://www.nngroup.com/articles/usability-101-introduction-to-usability>
- Nikiforova, O., Babris, K., Madelāne, L. (2021) Expert Survey on Current Trends in Agile, Disciplined and Hybrid Practices for Software Development, *Applied Computer Systems*, vol.26, no.1, 2021, pp.38-43. DOI:10.2478/acss-2021-0005
- Nikiforova, O., Gusarovs, K. (2020) Anemic Domain Model vs Rich Domain Model to Improve the Two-Hemisphere Model-Driven Approach, *Applied Computer Systems*, 2020, Vol. 25, Issue 1, pp.51-56, DOI:10.2478/acss-2020-0006
- Norman, D. (2013) *The Design of Everyday Things: Revised and Expanded Edition*. Hachette UK.
- Rivero, J. et al. (2011) From Interface Mockups to Web Application Models, p. 264. : DOI:10.1007/978-3-642-24434-6\_20.
- Rivero, J.M. et al. (2014) Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering, *Information and Software Technology*, 56(6), pp. 670–687: DOI:10.1016/j.infsof.2014.01.011.
- Rosado da Cruz, A.M. and Faria, J. (2010) A Metamodel-Based Approach for Automatic User Interface Generation. : DOI:10.1007/978-3-642-16145-2\_18
- Saputra, D.G. and Azizah, F.N. (2013) A Metadata Approach for Building Web Application User Interface, *Procedia Technology*, 11, pp. 903–911: DOI:10.1016/j.protecy.2013.12.274.
- Seidewitz, E. (2003) What models mean, *Software, IEEE*, 20, pp. 26–32. : DOI:10.1109/MS.2003.1231147.
- Staiano, F. (2022) *Designing and Prototyping Interfaces with Figma: Learn essential UX/UI design principles by creating interactive prototypes for mobile, tablet, and desktop*. Packt Publishing Ltd
- Stevens, P. (2000) *Using UML software engineering with objects and components*. New York: Addison-Wesley [https://www.academia.edu/1465995/Using\\_UML\\_software\\_engineering\\_with\\_objects\\_and\\_components\\_2000](https://www.academia.edu/1465995/Using_UML_software_engineering_with_objects_and_components_2000)
- UML, Unified Modelling Language Specification Version 2.5.1, <https://www.omg.org/spec/UML/2.5.1>
- Urbietā, M., Torres, N., Rivero, J.M., Rossi, G., Dominguez-Mayo, F.J. (2018). Improving Mockup-Based Requirement Specification with End-User Annotations. In: *Garbajosa, J., Wang, X., Aguiar, A. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2018. Lecture Notes in Business Information Processing*, vol 314. Springer, Cham. DOI:10.1007/978-3-319-91602-6\_2.
- Uzayr, S. bin (2022) *Mastering UI Mockups and Frameworks: A Beginners Guide*. CRC Press.
- Wang, G. and Shan, S. (2007) Review of Metamodeling Techniques in Support of Engineering Design Optimization, *Journal of Mechanical Design - J MECH DESIGN*, 129. : DOI:10.1115/1.2429697