University of Twente

Faculty of EEMCS

MSc Human Media Interaction

3D Interactive Visualisation Framework for Simulated Systems in Large Scale Print System Design

Author:
Yannick Donners

Supervisors:
ir. H. Koppelman
dr.ir. H.J.A. op den Akker
ir. E. Schindler PDEng

July 4, 2017

UNIVERSITY OF TWENTE.

océ
A CANON COMPANY

Abstract

Even with the digitalisation of today's culture, development in advanced large-scale print systems is a growing activity at Océ. In the design of the large-scale print systems multiple disciplines work together to produce quality products. The multi-disciplinary teams in Océ combine their knowledge to define the print system and its behaviour. Present challenges in supporting the Océ engineers in their productivity through visualisations are caused by the lack of a single visualisation framework that facilitates the different disciplines. Overcoming these challenges can be achieved by presenting a novel visualisation framework, that is optimized in terms of user experience and productivity, to aid the Océ engineers in multiple disciplines. In this report we present the development and evaluation of such a novel visualisation framework. The evaluation of the visualisation framework is done using the System Usability Scale (SUS) followed by a semi-structured interview. The results from the SUS indicated that, with a SUS score of 78, the visualisation framework is well within the range of acceptable (70-100). In addition, the participants (N = 6) rated the visualisation framework with a 7.3 out of 10 indicating the visualisation framework performed as expected but requires improvements to fulfil all their needs. Overall, the visualisation framework proved sufficient in supporting the engineers in configuring and observing simulators throughout the print system design.

Acknowledgements

This thesis presents the work done during my graduation project at the University of Twente. I am thankful I had the opportunity to do my graduation project at Océ technologies. My deepest gratitude goes to all people that supported and accompanied me throughout the project.
Firstly, I would like to thank my supervisors: Herman Koppelman, Rieks op den Akker; and my daily supervisor at Océ technologies: Eugen Schindler. Throughout the process Herman and Rieks guided me in making the correct decisions with regards to the users and strived for bettering my academic writing skills. Thank you for the strong guidance throughout the project. Eugen, thank you for trusting in my capabilities and allowing me to explore new territories throughout the project. You motivated me to achieve higher goals.
I would also like to thank Joost van Pinxten for the advice throughout the project. With the discussions for potential solutions, the project reached a higher level.
In addition, I would like to thank my family and girlfriend for their strong support throughout the entire project.
Finally, on a personal note, I would like to thank my colleague and friend Roel van der Tempel for our discussions during the 7 months of the project, which made the journey towards graduation much more fun and memorable.

Yannick Donners
July 4, 2017

Contents

1 Introduction

In the contemporary era where a trend is present towards the digitalisation of print, physical prints still have a place in the society [31]. Océ [6] is a multinational company that manufactures and sells high performance printing solutions for businesses. They aim to accelerate new digital print technologies and transform them into local printing products and services for blue-chip multinationals around the globe and creative studios around the corner. Within Océ the development of innovative print systems is a growing activity.

In the early stages of software development in print system design, physical prototypes are not present due to unknown hardware requirements and costs. The absence of physical prototypes weighs down on the possibility to develop control software for hardware components to make sure that the print system, after production, behaves as it should. This challenge is overcome by modelling and simulating the print system behaviour in a virtual environment. The software developed by engineers communicates with software-generated models of physical components. An example is a piece of software that controls the location of a simulated print head. This provides insights into possible errors that might occur when running the software on a physical prototype.

In a traditional scenario, engineers, designers and architects from chemistry, physics, mechanical engineering, electrical engineering and software engineering would make their own models to do individual specialistic analyses. However, within Océ multidisciplinary teams are formed containing engineers from all disciplines. The engineers provide models and simulations about possible print system behaviour specific to their discipline. To aid the engineers within a team to communicate and collaborate effectively, graphical representations, also known as visualisations, of print system behaviour are used.

In the current situation, frameworks for physical visualisations and charts are present for representing print system behaviour in Software-In-the-Loop and paper-handling simulations and models. These visualisation frameworks provide physical visualisations for sheet behaviour. The visualisation frameworks are built by the engineers to fulfil their direct needs. Currently, first prototypes of visualisation frameworks have been made, but the engineers have need for a more productive visualisation framework. In addition, user experience and productivity of the present visualisation frameworks are not considered in the development of these visualisation frameworks.

As the multidisciplinary teams rely on communicating concepts between disciplines, a new, evolving visualisation framework based on modern technologies should be developed that provides a fluent and natural user experience and supports the productivity of the Océ engineers in multidisciplinary teams.
Resulting from the goal of this report the following research question is stated:

- *What visualisation and user interface does an expert require for configuring and observing simulators (such as software-in-the-loop plant simulation, high-level behaviour simulation and physics simulation) for the support of print system design?*

In this report, we design and develop a visualisation framework and through user evaluations verify its capabilities of supporting the Océ engineers in configuring and observing simulators in print system design. The research question above focuses on determining the requirements and the implementation of the requirements. When the requirements are implemented, we determine whether the visualisation framework is suited to support the Océ engineers by answering the following research question:

- *To what extent is the developed visualisation framework able to support the experts in configuring and observing simulators in print system design?*

Prior to this research [15], we focused on developing an understanding of what user interface aspects are required by the Océ engineers. Through two iterations in a user study, multiple graphical user interface (GUI) prototypes were presented to the engineers, resulting in useful insights on GUI aspects required by the Océ engineers to aid their productivity.

This thesis focuses on the design, development and evaluation of a novel visualisation framework that provides a natural and fluent user experience and aid the Océ engineers of multiple disciplines to achieve increased levels of productivity. We implement the GUI designed in the research prior to this report and develop the functionalities as required by the Océ engineers.

In this report, we first recap on the discussed literature and results obtained in Donners [15] in Chapter 3. In Chapter 4 related work is presented with similar goals in terms of developing visualisation frameworks. Then, in Chapter 5 the development of the visualisation framework is presented in terms of requirements, design decisions and system architecture. Chapter 6 discusses the evaluation of our visualisation framework within the workflow of the experts and presents the results. In Chapter 7 the overall development process and the evaluation of the visualisation framework is presented. Then, a set of ideas is mentioned for future work on the visualisation framework in Chapter 9. Finally, conclusions are made in Chapter 8 that verifies the success of the visualisation framework.

2 | Context

In this chapter we discuss the struggles present to the Océ engineers, the current solution for multi-disciplinary development in large-scale printing system design and why it is relevant to resolve the problem. In addition, relevant terms and concepts are explained in section 2.1.

In chapter 1 we already briefly presented the problem at hand and how a resolution to problem will be achieved. However, we first need to understand the current situation of the Océ engineers in order to improve their workflow. Also mentioned in chapter 1, the engineers use simulators to perform a behavioural analysis of the printing system and work towards solution of behavioural errors. The different disciplines among the engineers, mechanical, electrical and software, each use their own models to perform such behavioural analyses. Because each discipline has their own models, communication and collaboration among the disciplines is a challenge to overcome.

Currently, a 3D physical visualisation framework and a 2D visualisation framework are present to these engineers. However, these frameworks are limited to a single field, namely sheet behaviour, across the disciplines. There are other fields (e.g. ink handling and fixation) which require a novel visualisation platform that stimulates communication and collaboration across the mechanical, electrical and software engineers. Such a platform can facilitate in the overall productivity as engineers can easily explain their concepts through shared visualisations.

2.1 Printing Domain Background

In this section we will elaborate on Canon Océ as a multinational company, the functioning of a large-scale printing system and the simulators used within the development of these large-scale printing systems.

2.1.1 Océ

Océ, a Canon company, is a multinational company that manufactures and sells high performance printing solutions for businesses. Océ operates a global network of R&D centers to connect emerging digital print technologies to future markets. Spread among the R&D centers, Océ has around 24,000 employees in 25 countries. The

Océ headquarters together with main R&D site is located in Venlo, the Netherlands. Océ has document printing systems families as well as large format printing systems product families. Document printing systems are usually multi-functional devices which provide scanning, printing, and copying functionality. Large format printing solutions are commonly used for printing blueprints or advertisements.

2.1.2 Large-scale Printing System

In figure 2.1 a decomposition of the VarioPrint i300 is shown. The VarioPrint i300 is the flagship cut-sheet printing system of Océ. This system is used for extremely fast printing of black-and-white and color on A3 or A4 sheets. The figure shows the different modules present in most contemporary business oriented printers. Each of these modules fulfils a well-defined task and can be developed separately. The modules are described in more detail below (numbering corresponds with the figure):

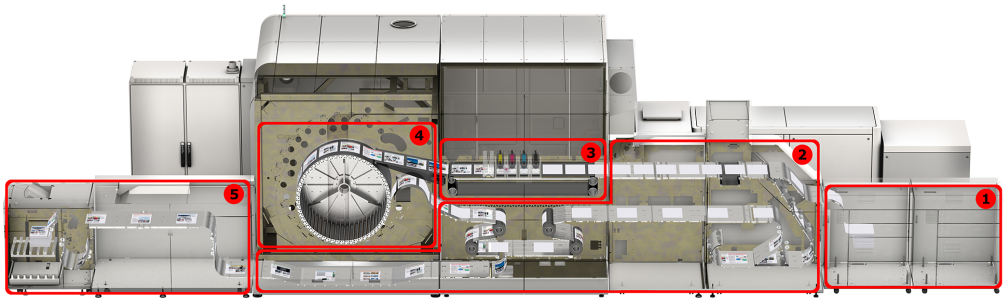

Figure 2.1: A decomposition of the VarioPrint i300. 1. Paper input module, 2. Paper transport module, 3. Printing process module, 4. Fixation, 5. Paper output module

1. **Paper input module (PIM)**: This module provides paper from the outside world into the system.
2. **Paper transport**: This module is concerned with the transportation of a paper sheet throughout the different modules in the printing system, taking into account that the paper sheet is at the correct location at any given moment in time.
3. **Printing process unit**: This module ensures the print head position and correct conditions for effective printing to print ink onto the paper.
4. **Fixation**: This module takes care of the drying process of the ink on a paper sheet.

5. **Paper output module**: This module provides a buffer before post-printing operations such as stapling and binding can be performed in a finisher. Finishers are usually attachable modules and multiple finisher modules can be chained to achieve more complex post-printing operations. Once passed through the attached finisher, the sheet moves to an output tray.

Each of the previously discussed modules is built up using hardware components. These hardware components ensure the correct functioning of the module and are controlled by software developed by the engineers. This means that predicting the functioning of the hardware components is extremely important and is achieved through simulators. In the following section we discuss the different simulators used in the development process.

2.1.3 Simulators

Simulators are applications with the sole purpose of running simulations to predict possible behaviour for input defined by the user. One of the more commonly applied simulations within Océ is a software-in-the-loop simulation to examine the behaviour of the embedded control software. Another simulation is concerned with the timing of the sheets (i.e. ensuring that the sheets are at the correct locations at all times). This section provides some context on the simulation environments used in Océ.

In a Software-In-the-Loop (SIL) simulation environment, the behaviour of a device is examined [26] [17]. The simulation runs embedded software on simulated and idealized hardware. The goal of the SIL environment is to test the behaviour of the embedded control software. The simulation is not entirely realistic as mechanical and other real-world problems do not occur. However, user-specified errors can be induced manually to create such problems. The simulation gives a good indication of general software behaviour under normal scenarios as well as scenarios that require error recovery. As mentioned in 1, using simulators rather than physical prototypes for testing, is very cost-effective and time-efficient since every engineer can have a simulation set-up on their workstation compared to testing with only a limited amount of physical prototype time available to such projects.

Another simulation environment is the Timing Design environment. This simulation environment is used within the early stages of the project to calculate and animate the paper flow within the paper path. Embedded software nor hardware interaction are prerequisites for the use of this type of simulation.

3 | Prior Research on Design of Graphical User Interfaces

In the research prior to this report [15], we performed a study to better understand the needs of the Océ engineers involved with print system development and work together with them towards a graphical user interface (GUI) that would adhere to their needs. In order to do so, a literature study was performed to understand what is required when designing a GUI and a natural interaction according to Nielsen [22] and Shneiderman and Plaisant [29]. In addition, state-of-the-art techniques for visualizing 3D elements were researched to understand how to implement a 3D visualisation into a GUI. To develop a set of requirements for the visualisation framework and the GUI of the tool, the knowledge acquired through the literature survey was combined with an user analysis to properly understand their needs and preferences for a visualisation framework. The requirements (found in Appendix C in this report) formed the basis for a two-step, iterative process in designing a preferable GUI for the engineers. The Océ engineers eligible for a new visualisation framework can be divided into two disciplines: timing design and Software-In-the-Loop (SIL). Both disciplines require different aspects to be visualized as well as feature to allow for an efficient workflow. The timing design domain is mainly concerned with ensuring the correct location of a sheet at each moment in time. Which requires focus on the visualisation of the sheets and the paper path. SIL, on the other hand, is not specialized to a single domain, but extends to multiple sub-domains concerning the behaviour of the sheets, behaviour of moving parts and multiple other domains. Each of these sub-domains requires different aspects of a printing system to be visualised. However, by combining the visualisations of each sub-domain into a single visualisation, multiple disciplines can work closely together as their disciplines are represented in every visualisation. In both iterations the participants were presented with a set of mock-ups and through a modified system usability scale (SUS) survey the results were measured [11],[10]. The final outcome of the user study showed that they are quite satisfied with the designs presented in the second iteration (Fig 3.1) however some improvements should be made in order for the engineers to implement it in their workflow. The major changes proposed by the engineers were the adjustability of the GUI in terms of resizing the different GUI elements and additional features to the 3D viewport as well as the

additional features to the diagram.

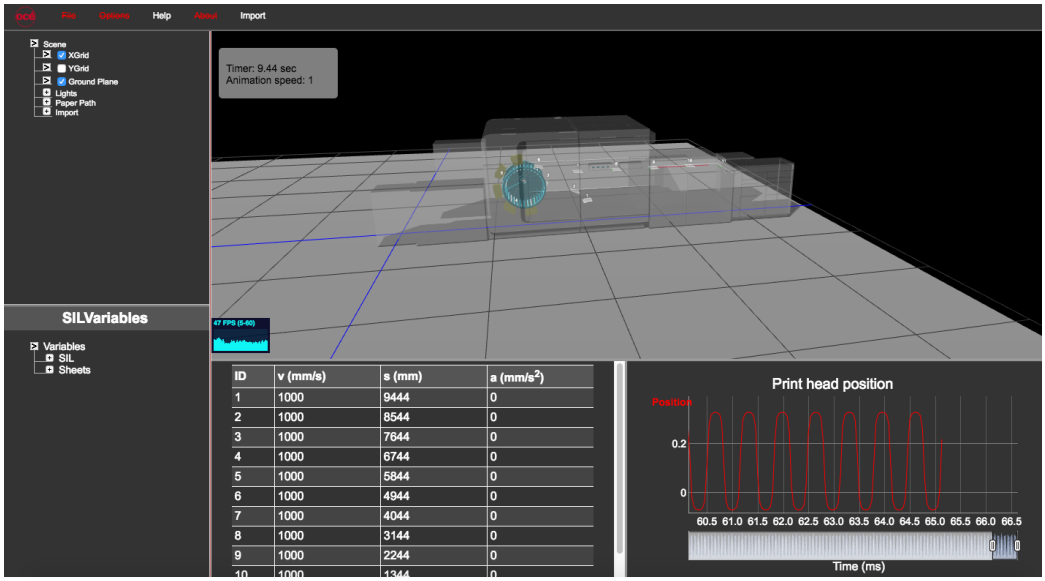

Figure 3.1: The outcome of the graphical user interface used in the second iteration of research performed previously to this thesis. The graphical user interface is divided in five windows containing an overview of present objects in the visualisation, a 3D visualisation, a diagramming facility, a table rendering facility and additional information about variables relevant to the simulation.

4 | Related Work on Contemporary Visualisation Systems

3D visualisations techniques are currently used in three fields of study, namely: biology, geography and mechatronics. These areas benefit largely from providing visual insight into large datasets created in these fields of study. Rego and Koes [25] and Pettit and Marioni [23] provided novel and successful methods for visualizing biological matters using WebGL. WebGL allows for the implementation of a 3D elements in a web browser and thus losing operating systems restrictions as well as losing restrictions for plug-ins for web browsers.

Rego and Koes [25] provided a visualisation as an alternative to existing visualisation frameworks for visualising interactive molecular data. The benefits of using a WebGL visualisation systems are the easy implementation of their framework in other websites, sharing the visualisation through a simple link and easy alteration of the source code to adapt the framework to specific tasks. They proved that a WebGL solution is comparable in performance as existing native applications in addition to being compatible with all operating systems.

Pettit and Marioni [23] used a similar approach. Their main goal in the development of the visualisation framework was to provide a free, cross-platform solution accessible to non-expert users to provide essential insights in biological data. Thus BioWeb3D is developed to be a simple and quick way to view 3D data focused on biological applications. Using HTML5 capabilities of reading local data rather than having to upload to a server and wait for the response, increases the ease of use for the users. Although using a WebGL solution may have limitations in the rendering power compared to native applications, it suffices in representing 3D data quickly in a visualisation for non-expert users.

In another area, geography, using 3D visualisations is becoming more common. In geography, there are 2 sub-areas in which visualisations are quite common: urban planning and geographical information systems (GIS). Lloret et al. [20] introduced a new method for visualising interactive 3D maps for urban planning. Combined with data obtained through GIS procedures, a set of 3D icons (e.g. buildings) are developed and can be readily placed within Google Earth to provide a quick and low cost visualisation of a possible urban planning scenario. By using Google Earth, the visu-

alisation can easily be distributed and shared among involved parties to the projects. Another approach is provided by Pouke et al. [24], where an experimental visualisation framework is developed to better understand public transport in urban areas. In their framework, Virtual Bus, they have opted to use WebGL for its easy access to many users. Through a simple server-client pipeline, real-time information can be obtained about the public transport and be directly visualised in their framework.

In the latter area, mechatronics, Dey et al. [14] developed a framework to enhance E-learning using virtual laboratories. These so-called virtual laboratories for simulated mechatronical environments are gaining in popularity due to its worldwide accessibility, easy implementation and reduction of high machinery or hardware costs. In the framework, they opted for using a server independent web solution based on WebGL for building the 3D geometry used in the visualisation. A model for simulating possible behaviour for the robotic arm used in their framework has be added to the web solution rather than connecting to a server or local client.

Common among the discussed research is the use of WebGL for easy access and distribution of the visualisation framework. Comparing the literature mentioned above to the literature on 3D visualisations reviewed in Donners [15], we can see a move towards WebGL solutions for developing visualisation systems. It shows that WebGL is capable of providing enough performance to deal with larger cities and molecular structures. In addition, WebGL removes client side dependencies and thus increase accessibility for the users. Data can be provided through either a server-client pipeline or using the HTML5 file reading capabilities.

5 | Work Method

In this chapter the method applied for developing the novel visualisation framework is discussed. Firstly, we present the development strategy applied to the development of the visualisation framework. Then, we discuss the method applied for engineering the requirements that fits the development strategy chosen for the visualisation framework. In Section 5.3 we present concepts used for the implementation of 3D visualisation in the visualisation framework. In addition, we discuss the design decisions that have been made throughout the development and the system architecture of the visualisation framework. Finally, we reflect on the implementation of the requirements.

5.1 Development Strategy

In Royce [27] three major development strategies are mentioned:

- **Waterfall:** In a waterfall development strategy, a sequential process is described following a pre-determined set of stages. Most waterfall projects suffer from inefficient integration and late discovery of design issues. In recent years, a trend has developed in which people tend to move away from the traditional waterfall development strategy.

- **Iterative:** In an iterative process cycle a set of demonstrable releases are planned throughout the development of the software. This method is also referred to as a mini-waterfall model as every iteration requires to be fully completed before the next is started. Using an iterative method allows for exposing design issues in early stages of development and resolving them in the next iteration.

- **Agile:** The agile process aims to reuse available assets to kick-start the development of the new application. Similarly to an iterative development process, agile applies iterations throughout the development of the software. In addition, agile implements an incremental process in order to maintain working software. Reusing assets and incrementally developing software allows the team

quickly develop deliverable products for testing, allowing to iterate effectively and efficiently.

In recent years, a move from the conventional waterfall model towards agility is being made among software development teams. When moving from a traditional waterfall strategy towards an iterative strategy, the ease of revising functionalities increases and the need for overhead planning reduces. Modern, iterative development enables better insight into quality because system characteristics that are largely inherent in the architecture (e.g. performance, fault tolerance, adaptability, interoperability, maintainability) are exposed earlier in the process where issues are still correctable. The iterative approach forces integration into the design phase through a progression of demonstrable releases, thereby exposing architectural uncertainties early and allowing them to be resolved efficiently in the context of iterations. Presenting the demonstrable releases early and often is also encouraged by Fullerton et al. [16] and Lundgren [21] for the development of interactive software. By allowing users to be involved early and often throughout the development of the system, we can detect problems early and reduce the amount of scrap and rework. The result is a more robust and maintainable product delivered with a higher probability of success. As interactivity of the system is important as well as a high probability of success, we apply an iterative development strategy to include the users early and often throughout the development of the system.

In figure 5.1 an overview is shown of how an iterative development strategy is executed. An initial plan and set of requirements are defined, followed by presenting a design that adheres to the requirements. The design is then developed, tested and sent to the user for evaluation. On the basis of the feedback received from the user, a new cycle is started to implement the feedback into the design of the software. An iterative design strategy is typically indicated by a subset of principles defined by Royce [27]:

- **Architecture-first approach:** This principle aims for integrating a demonstrable application in the design phase in order to expose and resolve significant errors that might cause design breakage.

- **Iterative life-cycle with early risk confrontation:** It is hard to define the entire problem, design the entire solution, build the software and test the end product in sequence. Therefore, applying an iterative design strategy allows for refining the problem, designing components of the entire solution and focus on testing the components rather than the entire solution as shown in figure 5.1.

- **Component-based development:** Applying component-based development reduces the amount of human-generated lines of code, as a component is predefined structure that can be easily implemented into the software solution.

- **Supporting round-trip engineering:** This principle allows for redefining and adding requirements throughout the process cycle in order to address the problems arising during development.
- **Addressing intermediate artefacts through a demonstration-based approach:** By providing multiple demonstrable versions of the software solution throughout the development cycle, the different functionalities can be tested. During these tests, errors and feedback can be collected to redesign the functionalities as required.
- **Plan intermediate releases in groups of usage scenarios with evolving levels of detail:** This principle involves the user directly in the design process. Each release of the software solution has an increased amount of functionalities (i.e. increased levels of detail) and should be presented to the users for evaluation within the operational context of the system.

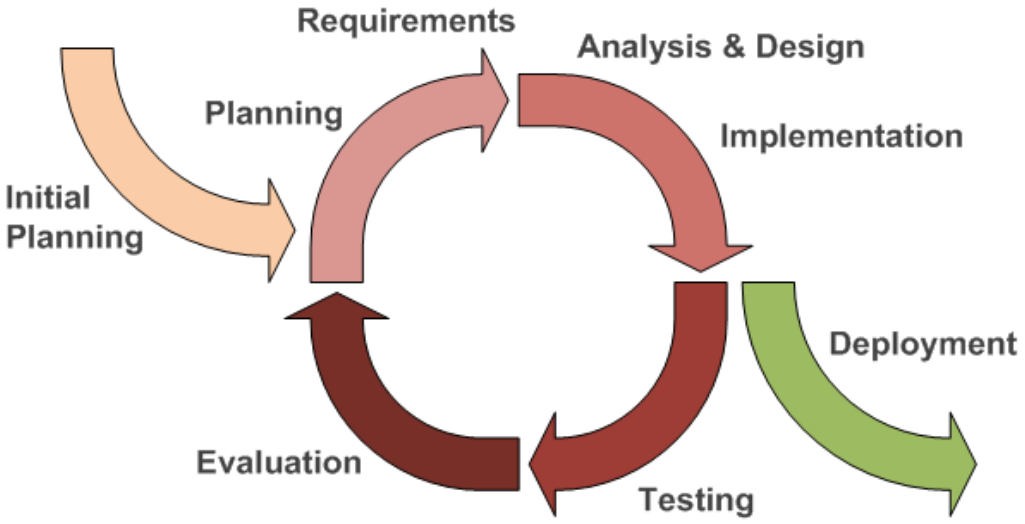

Figure 5.1: The iterative design strategy.

These principles motivate the cyclic workflow as shown in figure 5.1 where users are involved early and frequently throughout the development of software. The early involvement of the users helps to identify issues early on components of the software rather than the entire application and adjust the requirements towards their needs. This type of requirements engineering is referred to as iterative requirement engineering. In most of the organizations reviewed by Cao and Ramesh [12] a high level of adoption is present for iterative requirements engineering. In an iterative requirement engineering strategy only a set of critical requirements are defined before the development of the software is started. Throughout the development of the software,

requirements are added, revised, extended or removed. Benefits of applying iterative requirements engineering are [12], [18]:

- A higher level of customer satisfaction due to direct involvement in the development of the software;
- requirements are clearer and more understandable due to the direct contact with the customers;
- iterative requirements engineering facilitates joint (between developers and customers) discovery of potentially interesting solutions.

However, iterative requirements engineering also present a set of challenges that should be considered. These challenges are defined by Cao and Ramesh [12] and Inayat et al. [18] as:

- Cost and schedule estimation as only a limited set of requirements are present at the start of a project;
- neglecting of nonfunctional requirements as the focus in the early stages is on ease of use rather than stability and security;
- minimal documentation due to the mainly verbal agreement on requirements.

Since the potential challenges that accompany iterative requirements engineering are known beforehand, we are able to counter them early. From the start of the development we define a scope to which this project has to adhere. Throughout the project, we are able to extend to scope and add additional functionalities. Because we define the scope early, focus on stability and security is present even though the scope might be extended. Finally, we create a set of requirements that are maintained and extended throughout the development of the visualisation framework.

In the iterative design cycle, intermediate releases of the software are presented to the users for immediate feedback resulting in the addition, extension or removal of existing requirements. Throughout the project the list of requirements changes and so do the priorities of these requirements. In Section 5.2 we present the requirements as they are by the end of the development of the visualisation framework.

In research prior to this report [15], the first steps of an iterative design strategy have been taken. At first, a user analysis was performed to which a set of initial requirements were developed. From these requirements, two iterations of graphical user interface (GUI) designs were developed. In this report, we continue on the basis of the previously acquired GUI designs, and continue the iterative development of a novel visualisation framework. Throughout the development of the visualisation framework, a modular architecture is adopted (explained in more detail in Section 5.4.2) in order to breakdown the problem into smaller chunks. The smaller chunks

can then be translated into components of the system containing certain functionalities. In addition, intermediate releases of the visualisation framework are tested in a natural setting by a single user. The users is an expert with regards to timing design. Although it being a single user in a single domain, errors and design breakage are detected early before releasing a final version of the visualisation framework.

5.2 Requirements

In this section we present the list of requirements in the state current to the end of the development of the visualisation framework. The requirements are formed based on the feedback received from the user. We distinguish between functional and non-functional requirements. In the research prior to this report we already stated a set of functional and user requirements relevant to the visualisation framework [15]. These can be found in Appendix C. We define functional requirements to contain information about what the system should do, whereas nonfunctional requirements specify how the system should perform a certain function. The requirements are formed in accordance to the MoSCoW method [13], where priorities are defined through must have's, should have's, could have's and would have's. In tables 5.1 and 5.2 the functional and nonfunctional requirements that are defined throughout the iterative development are presented. Each of the requirement contains a rationale explaining why the requirement is important to the development of the system.

Functional Requirements	
F1	The system must aid in the print system development.
Rationale	*Throughout the system data from simulators is visualized with the goal to assist the users in understanding the outcome of certain scenarios. Aiding the users is a reason to use the system.*
F2	The system must run on Microsoft Windows.
Rationale	*Microsoft Windows is the most common operating system among the users and therefore the application should function properly on this operating system.*
F3	The system must be extensible to use with different types of data from models and simulators.
Rationale	*Frequently used simulators in print system design are Software-In-the-Loop (SIL) simulators and timing design for verifying software and sheet behaviour. The system must include the functionality of these models and simulators, but not limit to these two.*
F4	The system must provide an option to choose to which simulator will be used.
Rationale	*Both simulators are to be connected to the same system and use different types of input. Thus, an option for choosing the preferred simulator must be present.*

F5	The system must be able to load files.
Rationale	*Both simulators provide static information concerning the paper path through XML and JSON files. In addition, the Timing Design simulator provides information necessary for the visualization through XML files.*
F6	The system must be able to visualize a print system paper path.
Rationale	*The paper path is important to both simulators as it belongs to the core of print system design. Therefore, the system must provide a visualization of the paper path.*
F7	The system must contain a 3D visualisation area.
Rationale	*A 3D visualisation allows for inspection of print system behaviour from multiple angles and thus providing additional insights.*
F8	The 3D visualisation must be interactive through keyboard and mouse input.
Rationale	*Most common input devices for desktop computers are keyboard and mouse. Allowing for keyboard and mouse to interaction with the 3D visualisation is important, e.g. for changing the camera angle to closely inspect component behaviour in the print system.*
F9	The system must contain a diagramming facility.
Rationale	*Providing data obtained from the simulators in a diagram allows for advanced data analysis.*
F10	The system must contain a table rendering facility.
Rationale	*Displaying data obtained from the simulators in a table provides a quick overview of the visualized data from the 3D visualisation area.*
F11	The table rendering and diagramming facilities must be interactive through mouse input.
Rationale	*Tables and diagrams are considered 2D visualisation elements and must be interactive through mouse input. This allows for closely inspecting data within the 2D visualisation.*
F12	The system must be able to load 3D objects.
Rationale	*For SIL simulations, 3D models of the components of print systems are used to visualize the behaviour of the components. For timing design, it proves useful to better understand the relative location of the paper sheet within the print system. As the solution is present in WebGL, a JSON file is a natively supported format for loading files.*
F13	The system must provide an overview of the present 3D objects in the 3D visualisation.
Rationale	*Providing an overview of 3D objects present in the 3D visualisation is relevant information to the users because they need to know what is visualized.*

F14	Through a button or draggable source additional diagrams and tables should be able to be added to the system by the user.
Rationale	*The option for displaying multiple diagrams and tables was highly requested by the users. By allowing the users to manually add additional diagrams and tables, they are not bound to a pre-set number of diagrams and tables.*
F15	The system should not affect the performance of the timing design or Software-In-the-Loop simulators.
Rationale	*The simulators already require complex calculations that require high performance. If performance is affected by the system, calculation might take longer and thus reducing the real-time factor of the simulations because the system and simulations run on the same machine.*
F16	The system should be based on open source software.
Rationale	*Using open source software increases the accessibility due to the reduction of costs. In addition, in open source software access to the source code is available and thus can be tailored towards the specific needs of the system.*
F17	The system could be able to alter the colours of segments in a paper path.
Rationale	*Allowing for indicating segments with different colours within the paper path could increase the effectiveness of the visualisation.*
F18	The system would be extensible for use with virtual reality.
Rationale	*In addition to traditional displaying methods, allowing for virtual reality of the 3D visualisation could increase the immersion of the user with the print system behaviour and allow for close inspection of the different components.*

Table 5.1: Iteratively defined functional requirements.

Nonfunctional Requirements	
NF1	On every cycle data from the Software-In-the-Loop simulator must be requested.
Rationale	*Requesting data on every cycle provides up-to-date information to the user and a smooth visualisation of the simulated data.*
NF2	Only information relevant to the 2D and 3D visualisations must be requested from the Software-In-the-Loop simulator.
Rationale	*Only requesting data that is required for visualising 3D objects and data present in the diagrams reduces processor load and increases performances. A higher performance allows for a more smooth and direct interaction with the system.*
NF3	Navigation throughout the 2D and 3D visualisations must be smooth.

Rationale	*For a natural and pleasant user experience, it is important that navigation for the 2D and 3D visualisations is smooth and direct. Stuttering of the animation might cause confusion as it impacts the visualisation of the simulated print system behaviour.*
NF4	Errors and warning encountered within the system should be logged and be accessible to the user.
Rationale	*Logging errors and warning encountered throughout the visualisations increases the serviceability of the system.*
NF5	The system should have a modular design.
Rationale	*Applying a modular software architecture to the system allows for easy maintainability of the system because only a single module requires maintenance rather than the entire system. In addition, easy extensibility of the system is achieved because additional modules can be developed and implemented without altering the core of the system.*
NF6	Cross-platform support could be implemented.
Rationale	*Allowing the system to be accessed from any operating system increases the accessibility of the system.*

Table 5.2: Iteratively defined nonfunctional requirements.

5.3 Implementation of 3D Visualisation

In research prior to this report [15], we defined that it suits Océ engineers' best interest to allow for some abstraction concerning the 3D visualisation as well as high detailed models of the actual physical models present in the printing systems. Considering that definition, we discuss the use of geometric primitive objects for 3D visualisation in this section as well as the use of the high-detail CAD models.

5.3.1 Geometric Primitives

In 3D modelling, geometric primitives are the simplest of geometric objects that the system can handle. 3D objects such as a cube, cylinder and sphere are referred to as geometric primitives. For remainder of this report when we refer to a *primitive*, we talk about a geometric primitive. In this section we make a distinction between primitives representing physical objects and primitives representing properties of physical objects. The former primitives can be used to either provide a more user-friendly approach to some abstract data or to bring down the vertex count opposed to using a high-detail CAD model. For some objects in the CAD model, e.g. sensors, it might prove a more suitable solution to use a primitive to represent the behaviour of a sensor. The second use of primitives is used to display additional information to already present models by showing information such as text. Firstly, we discuss the primitives relevant to a print system 3D visualisation:

- **Coordinate System** A coordinate system uses one or more numbers, often referred to as coordinates, to uniquely determine the position of a point. A distinction can be made between global and local coordinate systems. There can only be a single global coordinate system present, whereas multiple local coordinate systems can be present in a single global coordinate system. Local coordinate systems work similar to global coordinate systems (i.e. a vertical y-axis), but can be translated, rotated or scaled within the global coordinate system. In figure 5.2 the relation between global and local coordinate systems is shown.
  - **3D Coordinate System** A Cartesian coordinate system which describes 3 axes. These axes are referred to as x, y and z.
  - **Curve Coordinate System** A local linear coordinate system, ranging from zero to one, present to a curve. This coordinate system is based on a per curve basis. Each curve has its own local coordinate system in which an offset (in a range from zero to one) can be specified, and is translated to a coordinate in the global coordinate system. In figure 5.3 the relation between a curve and global coordinate system is shown.
- **Coordinate** A position describing a point in a coordinate system.
  - **Curve Coordinates** A coordinate in the Curve Coordinate System is defined by a curve identifier as well as a curve-relative position. The curve-identifier provides information to which curve the coordinate is local. The curve-relative position is determined by an offset from the start of the curve in comparison to the relative length of the entire curve and is represented by a single number. E.g. a curve coordinate of 0.5 is located halfway the specified curve.
  - **3D Coordinate** A coordinate defined in a 3D Coordinate System. The coordinate's location is determined by x, y and z coordinates.
- **Bodies** This is a collection of all possible objects present in a 3D environment. Bodies are defined by a set of parameters, shaping the body to a certain object. We distinguish two types of bodies for the purpose of this 3D visualisation framework.
  - **Curve Body** This is a deformable object that is animated along a curve. The object contains information on which curve it is located and its curve-relative position. On the basis of the previously mentioned, the start and end point of the object are determined and the vertices are placed on the curve. The curve-relative position determines the position of the start of the interval of the curve body, as well as the end of the interval. In between the curve body is interpolated over the curve and when animated the curve

body will follow the flow of the curve. An example of a Curve Body is a sheet of paper.

– **Rigid Body** This is a non-deformable object which appearance is defined by a 3D mesh. Rigid Body transformations consist of translations, rotations and scaling that can be applied. The distances between vertices of this body maintain an equal ratio to one another. Examples of Rigid Bodies are printer components.

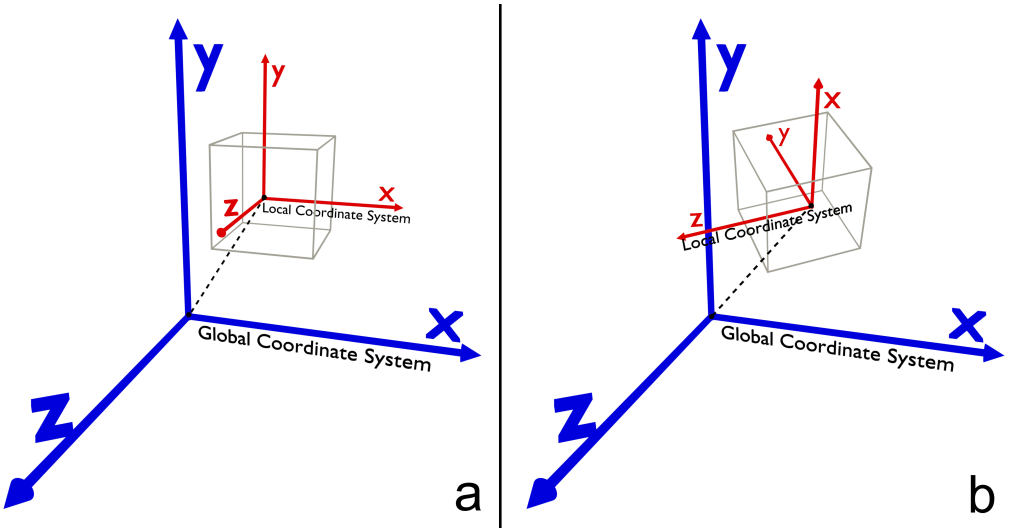

Figure 5.2: A visualisation of a global and local coordinate system. In (a) an object is placed within the global coordinate system with its own, local, coordinate system. In (b) the object is rotated and the local coordinate system (of the object) thus rotates along.

5.3.2 CAD Models

CAD objects belong to the primitive category of *Rigid Bodies* and are thus static objects in the sense that the vertex ratios (the relative distance between the vertices) remain unaltered throughout the visualisation. The objects acquired from the CAD models are used for animation in terms of translation, rotation and altering the color of the object. CAD models require a form of pre-processing in order to be presented in the form of a set number of vertices. A CAD model is a representation of a 3D model using surfaces rather than vertices. In order for a CAD model to be correctly rendered by a rendering engine, a conversion to a vertex representation of the 3D model is required. This is often achieved by exporting the CAD model to a standard format such as Wavefront Obj or Filmbox (FBX).

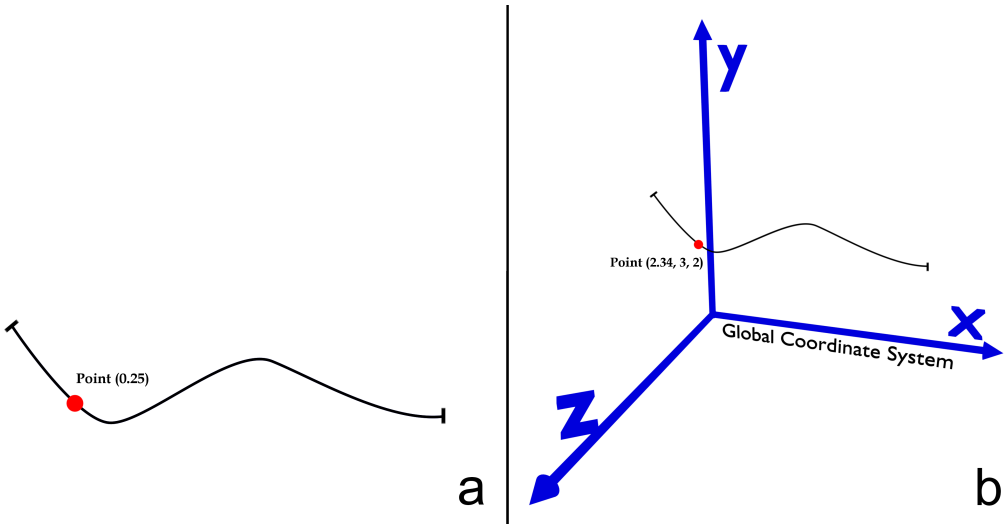

Figure 5.3: (a) shows a curve coordinate (ranging from zero to one) on a curve. (b) shows the curve coordinate converted to a coordinate in a global coordinate system.

5.4 Design Decisions

In this section we elaborate on important design decisions that are made during the development of the visualisation framework. As discussed in the previous section (section 5.1), the visualisation framework is developed as part of an iterative development process. This method allowed for rapid development of experimental version of the visualisation framework before choosing the best option.

5.4.1 Development Platform

In choosing the best framework upon which the visualisation framework is built, multiple platforms were explored in order to find the best-fitting solution. One of the big preferences of this project would be to use Open Source software to prevent any licensing costs. In the exploratory phase for choosing the right platform, a number of game development platforms were considered including Blender [3], Three.js [8], and (although not Open Source): Unity [9]. In figure 5.4 we present a matrix outlining the features of the before-mentioned game development platforms. In section 5.3 we defined the necessary geometric primitives and concepts that are required for a successful 3D visualisation framework. When looking into the matrix (figure 5.4), we can see that Three.js is the best fit for the 3D visualisation framework due to its advanced implementation of the curve system.

In light of using Three.js as the rendering engine for the visualisation framework, we reviewed related work using WebGL solutions creating 3D visualisations. WebGL is a cross-platform, royalty-free web standard for a low-level 3D graphicas, exposed to ECMAScript (i.e. Javascript is an implementation of ECMAScript) via the HTML5

	Blender	Three.js	Unity
Open Source	Yes	Yes	No
Allows for the development of an interactive application	Yes	Yes	Yes
3D coordinate system	Yes	Yes	Yes
Curve coordinate system	Yes	Yes	No, available through plugin.
Curve animation	No, not available in game development mode	Yes	No
Browser Integration	No, achievable through Blend4Web	Yes	Yes
Implementation of additional functionalities through scripting	Yes	Yes	Yes

Figure 5.4: A matrix displaying the benefits of each of the game development platforms.

Canvas element [2]. By using WebGL as basis for the visualisation framework the conditions for a cross-platform support, as stated in the requirement *NF6*, are met.

As Three.js is a WebGL solution for rendering 3D elements, it is the preferred option due to its accessibility and cross-platform support. In addition, Three.js showed that it provides a rigid system for dealing with curves and curve bodies. A downside of WebGL might be performance as Javascript runs on a single thread in the web browser. However, in Chapter 4 we found that WebGL only suffered from longer loading times rather than overall performance issues [25]. In addition, since JavaScript is only running on a single thread, it ensures it won't take away from the performance of any processes (e.g. simulations) running in the background. As system performance will only slightly be affected by the visualisation framework, it conforms to functional requirement *F15*. Using WebGL inherently means that the visualisation framework will be implemented in a webpage using HTML and CSS. Due to connectivity to the SIL simulator, which requires a native library for communication on the localhost, it is not possible to run the visualisation in a web browser such as Google Chrome or Mozilla Firefox. Therefore, a local server has to be hosted to which the SIL simulator can connect. In order to achieve this in combination with WebGL, the visualisation framework should be converted to a "native" application using Node.js [1] and Electron [4]. Node.js hosts a local server whereas Electron allows the JavaScript application to run in a windowed version as a native application to the operating system.

5.4.2 Modular Architecture

In iterative development, a modular system architecture is part of the development process as found in the principles presented by Royce [27] in Section 5.1. A modular architecture splits the application into a number of components that can be linked together through means of an interface. In one of the iterations, two experimental versions of the visualisation framework were developed: timing design and Software-In-the-Loop (SIL). However, with some minor alterations, the entire design could be merged into a single visualisation framework being able to connect to multiple simulators through interfaces. This is explained in more detail in section 5.5. A modular architecture is present as requested in the requirements *NF5*, and thus increasing the maintainability and extensibility of the system. The ease of maintaining the visualisation framework in a modular design comes down to resolving issues in a single component to ensure the functioning of the entire framework. Similarly, when extending the framework with additional features, the only requirement present is that it provides the data in a similar way as the framework requires in order to function. This would mean that if a component (e.g. the diagram) would take an array as input, a new component linking to the diagram would need to output its data in the form of an array.

5.4.3 Modular Graphical User Interface

Although the design of the GUI limited to the features provided by HTML and CSS, an adjustable and modular design can be achieved through using JavaScript libraries such as GoldenLayout [5] and React [7]. GoldenLayout provides a framework that allows for the interface to be adjustable in terms of scaling, moving and docking various elements. Combined together with React, each of the GoldenLayout elements can be filled with a React component. A React component is an object containing HTML elements. Characteristics of objects in programming are that instances of the objects can be created. This characteristic also applies to the React components and thus, multiple instances of a single React component can be created. An example would be a GUI where a single 3D visualisation area is present and multiple diagrams and tables as instances of diagram and table React components are present in the GUI. Ultimately, we allow the user to create such instances of React components through an action and add these to the GUI. In figure 5.5 the current implementation of this feature is shown. Allowing users to manually add additional diagrams and tables to the GUI is in line with the functional requirement *F14*.

5.5 Architecture

In this section we discuss the architecture of the visualisation framework. As discussed in the previous section, a modular architecture is implemented which translates to

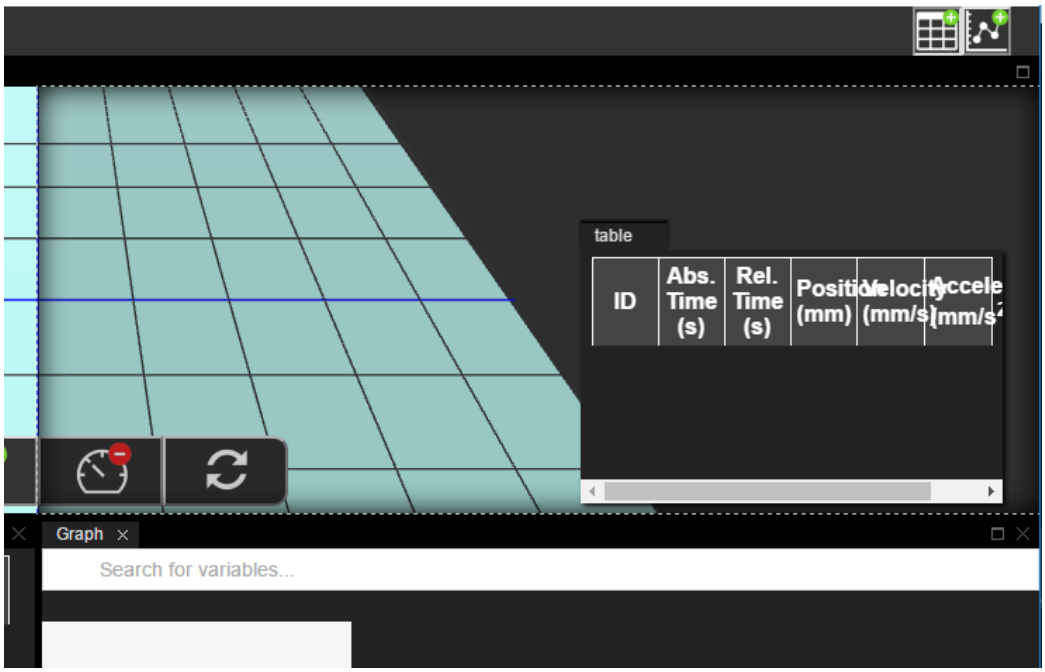

Figure 5.5: An additional table is added to the GUI by dragging the table icon in the upper right corner onto the GUI.

using different components. The architecture of the visualisation framework consists of the different components that will form the visualisation framework. Firstly, we discuss the different components and their functionalities. Secondly, we discuss how these components are deemed to connect. Finally, we present the overall architecture of the visualisation framework in its context and how these compare to the requirements defined in Section 5.2.

First off, we define a set of components (in figure 5.6 the visual components are presented) that are required for the visualisation framework in accordance to the defined functional requirements:

- **3D visualisation area** is a component that is required for rendering 3D objects. A rendering engine processes information about vertex locations and translates this information into objects that fit within the coordinate systems. This engine is also responsible for the animation of the objects throughout the visualisation (*F7*).

- **Diagram** provides data in a two-dimensional graph. This component translates data obtained from the simulator into a two-dimensional representation of a set of selected variables (*F9*).

- **Table** This component provides the data obtained from the simulator in an easy to read format in which only current (and thus real-time) information is

presented (*F10*).

- **Simulators** These components provide the information that requires visualisation (*F3*). In this report we discuss the use of two different simulators:
  - **Timing Design:** This type of simulator is concerned with providing information about the sheet location at certain moments in time. The files are generated beforehand and are provided as input for the visualisation framework.
  - **Software-In-the-Loop:** This simulator is concerned with the software and hardware aspects of print system design. Software-In-the-Loop enables engineers to test their software against a simulated form of hardware. This means that the software thinks it is talking to hardware components, when it is actually a simulator. This simulator provides real-time data as input for the visualisation framework.
- **Interface**: This component translates the data obtained from either simulator into a data format that is eligible for use with the previously mentioned components. For each of the simulators an interface is required to be able to work with the visualisation framework.

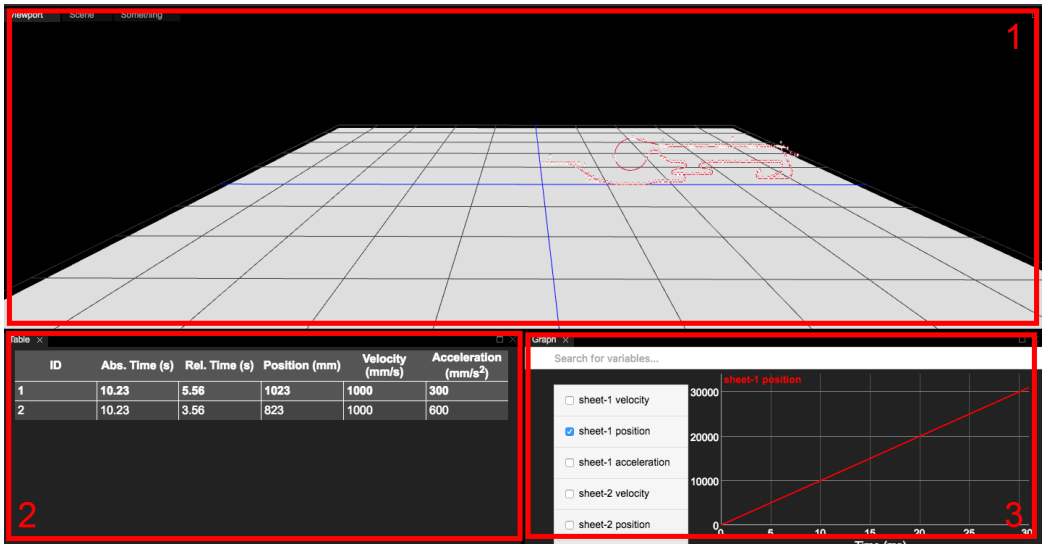

Figure 5.6: An outline of the visual components present in the visualisation framework. 1) 3D visualisation area, 2) Table, 3) Diagram.

In fig 5.7 we present a schematic overview of how the previously mentioned components work together. In the current design of the visualisation framework, a single simulator component is connected to the framework. We defined that there are multiple simulator components that must work with the visualisation framework (*F3*).

In order to tell the visualisation framework which type of data is expected, a choice is presented to the user to choose the preferred simulator (requirement *F4*). In figure 5.8 we present a GUI that provides the selection between two simulators: timing design and SIL. Through an interface the data obtained from the simulator is converted to a format that is eligible to work with the 3D visualisation area, diagram and table. Since the aforementioned are developed as components in a modular architecture, the maintainability and extensibility of the visualisation framework is increased. Generally errors are likely to only occur in a single component which then requires repairing rather than the entire framework. In addition, new simulators can more easily be implemented as the components have predefined input formats to which an interface between the new simulator and the 2D and 3D visualisation components must adhere. Similarly, 2D and 3D visualisation components can be added by altering the interface between the simulator and visualisation components to take the new component into account.

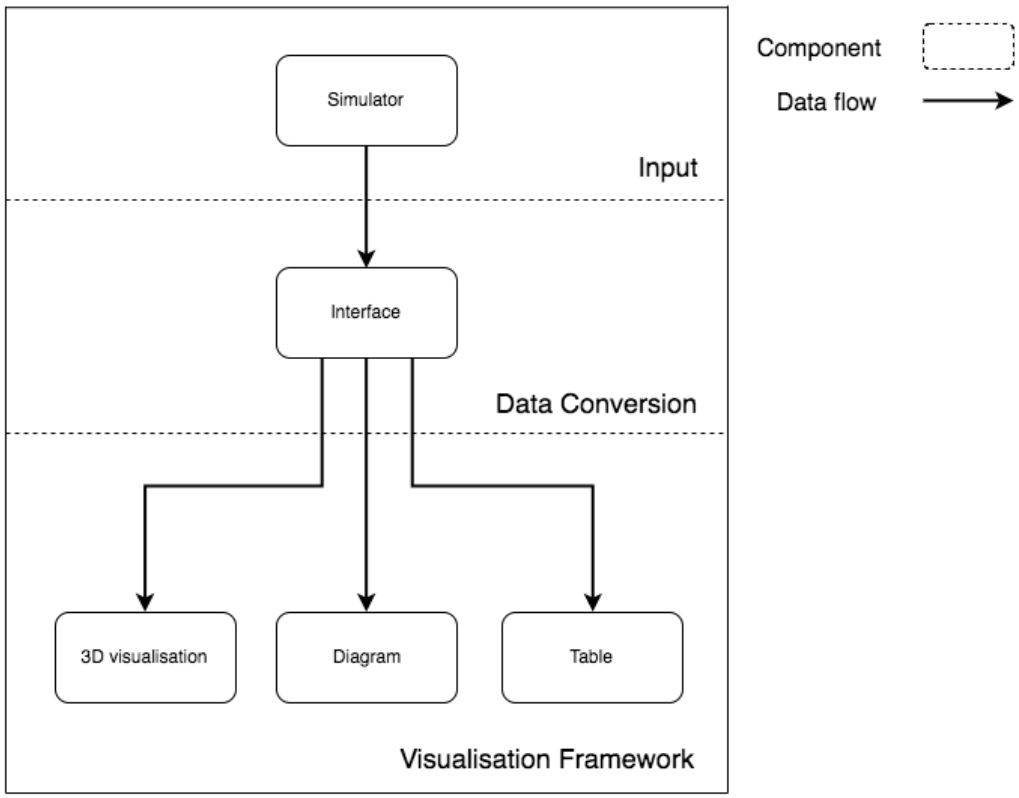

Figure 5.7: A schematic overview of the component-based system architecture with a single simulator connecting to the visualisation framework. Blocks represent components in the system and arrows represent a data-flow.

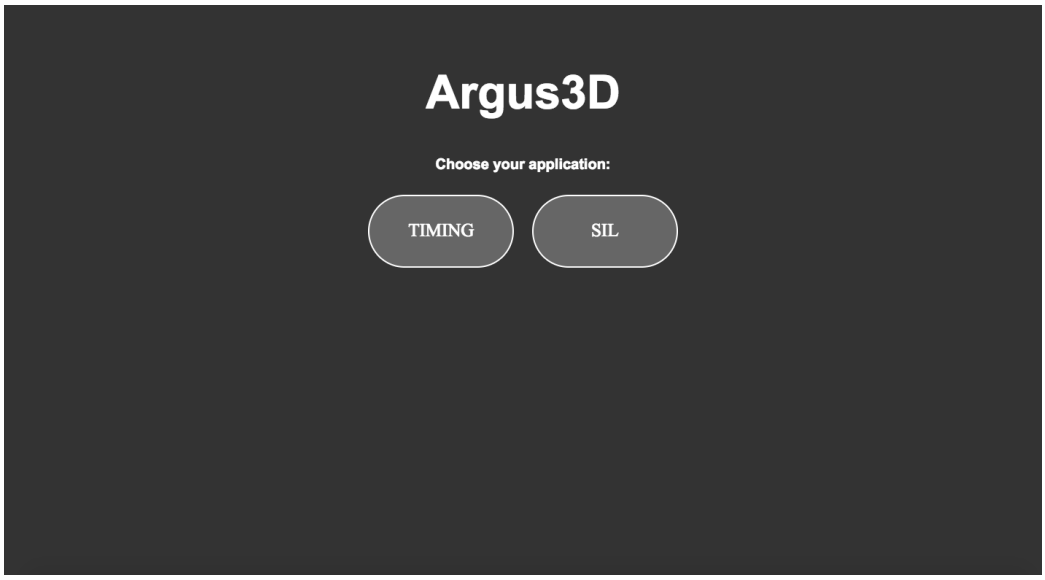

Figure 5.8: A GUI that presents options for choosing a simulator.

Mentioned above, the simulators have their own interfaces within the visualisation framework due to the differences in what format the data is provided. In the requirements we stated that the timing design and SIL simulators had to be implemented in the visualisation framework, therefore we discuss how these simulators are connected to the visualisation framework. Both simulators provide information about the state of objects (e.g. sheets or print carriages), but are unaware of the print system itself in terms of paper path layout or relative position of the objects to one another. The information about the paper path and the relative positions of the objects is present in predefined files. These files come in the form of XML and JSON files, and require to be loaded by the system (*F5*). In figure 5.9 we show how these files are loaded into the visualisation framework. These files are then parsed and presented in forms of 3D visualized objects such as the paper path (in accordance to requirement *F6*) as shown in figure 5.11. In addition for the SIL simulator, information about the components of the physical printer (print carriage, input models, etc.) is provided in a JSON file containing 3D models with relative positions to each other. Thus loading the 3D models adheres to requirement *F12*. In figure 5.10 imported 3D objects into the 3D visualisation area of a print system are shown.

Figure 5.9: A GUI that allows the user to define the files containing static information about the layout of the paper path and other print system components.

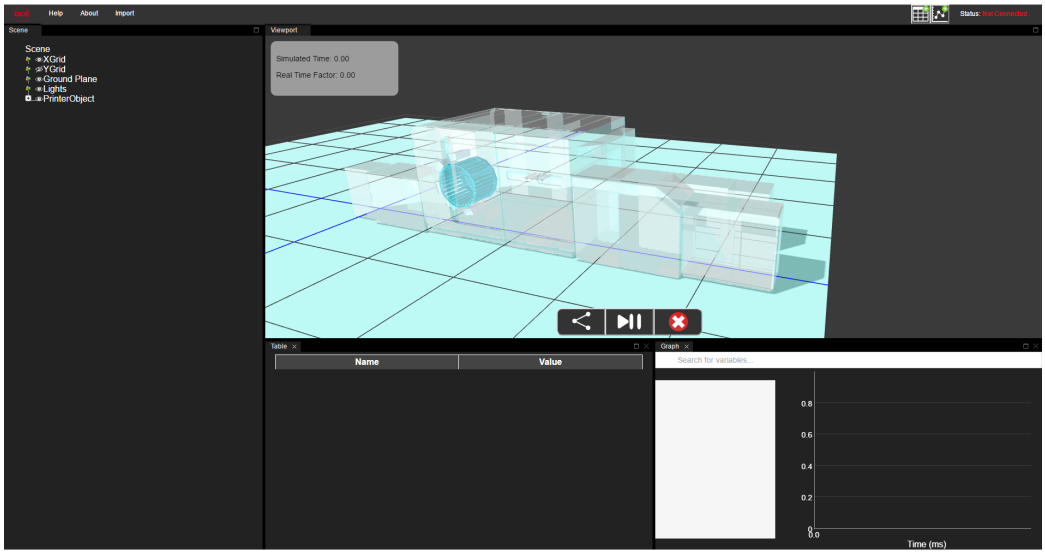

Figure 5.10: Static components with information about relative positions are loaded into the 3D visualisation area.

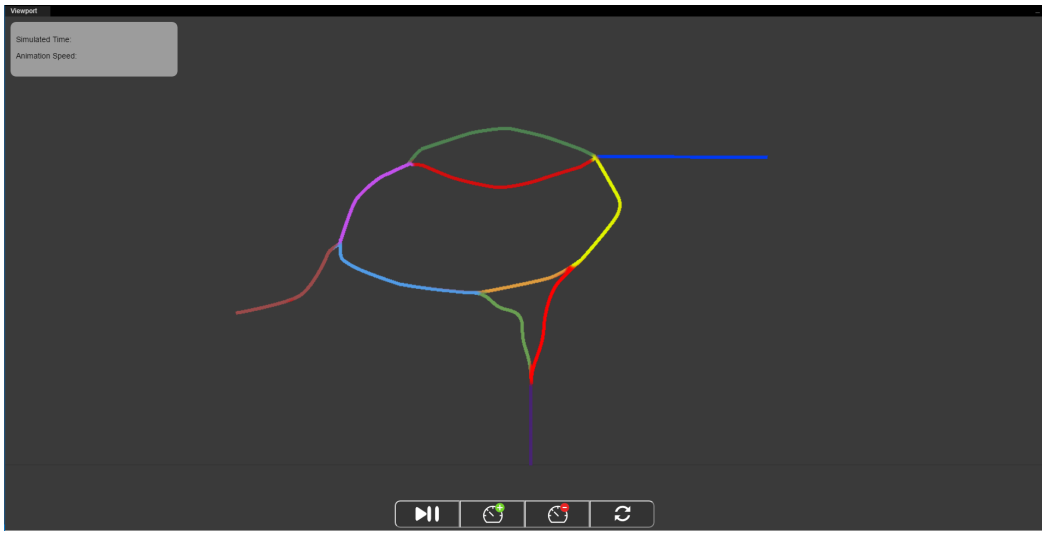

Figure 5.11: A paper path loaded in the 3D visualisation area.

Once the static data is loaded, the dynamic data can be visualized. Recall that for each simulator an interface was developed in order to obtain the data and visualize it within the visualisation framework. The timing design simulator provides information about the sheet behaviour on an interval basis and thus requires linear interpolation between the presented data points. The interpolation is required otherwise the visualisation would only present data that is already known from the input file. In figure 5.13 we present a schematic overview on how the data is presented and translated through linear interpolation to a usable visualisation. In the left area of figure 5.13 the data provided by the timing design data is presented as the red points and the linear interpolation done by the visualisation area as the dotted blue line between the red points. The red points correspond to sheet locations in the paper path, whereas the blue line in the paper path are the interpolated positions of the sheet in the paper path. Without the linear interpolation only limited information would be present in the visualisation framework. The interpolated areas provided valuable insights to the Océ engineers.

Data in the SIL simulator is calculated real-time. In order to access the data from the SIL simulator, a transfer protocol is required. This transfer protocol allows for sending and requesting commands that contain additional information. In figure 5.12 we present a schematic overview of the communication between the visualisation framework and the SIL simulator. In addition, the information from the SIL simulator needs to be requested as often as possible to provide an accurate representation of the simulate data in the visualisation framework. Therefore, on every refresh cycle in the visualisation framework a request is sent to the SIL simulator to request information about variables. In requirement *NF2* we stated that only information relevant to the 2D and 3D visualisations should be requested to maintain an acceptable level of performance. Therefore, internally in the visualisation framework the variables that

are required within the 2D and 3D visualisation are stored and only those variables are requested from the SIL simulator.

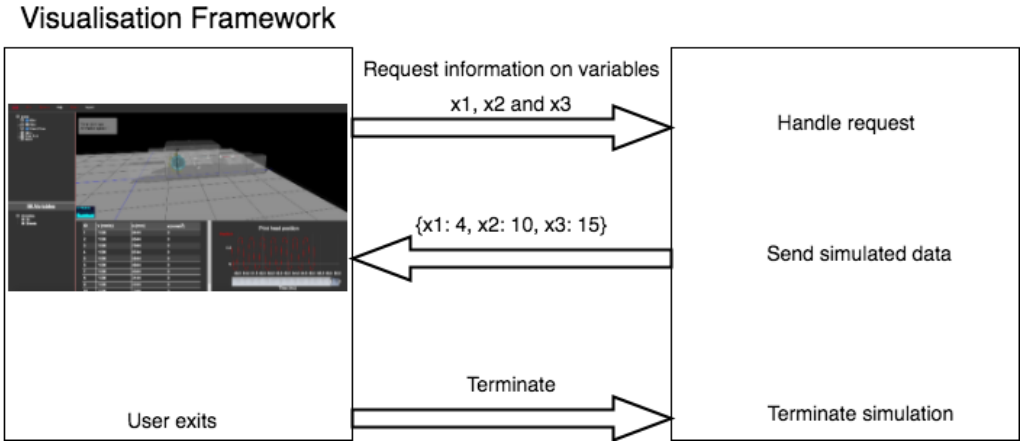

Figure 5.12: A schematic overview of a potential communication between the visualisation framework and the Software-In-the-Loop simulator.

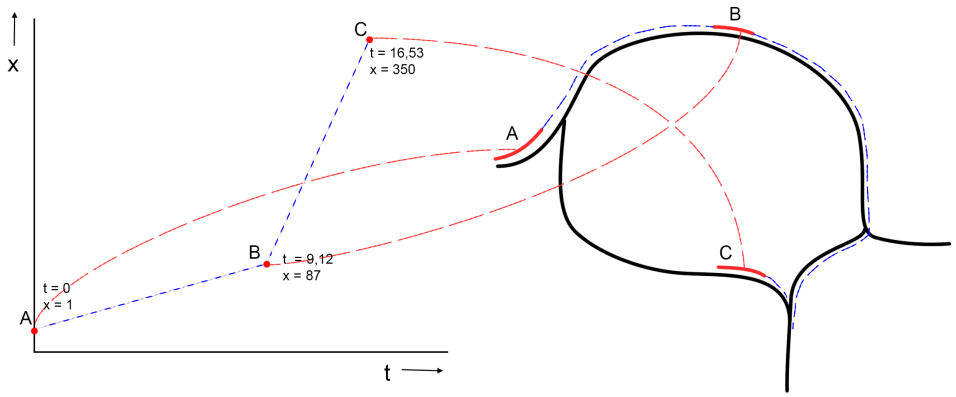

Figure 5.13: A schematic overview of the interpolation of the data provided by the timing design simulator. Point A in the diagram corresponds to the sheet location in the paper path. Similarly, point B in the diagram corresponds to sheet location B in the paper path. The dotted line presents the interpolated locations between points A and B in both the diagram and paper path.

5.6 Discussion of Requirements

Although most requirements have been implemented one way or the other, a few requirements are only partially met or not met at all. The only functional requirement that is not implemented in the current version of the visualisation framework is *F18*.

Due to the prioritization of the requirements in accordance to the MoSCoW method, this requirement was deemed a nice additional feature but not important to the system design. Although, adding a layer of virtual reality to the 3D visualisation area might provide insight into sheet behaviour that is otherwise harder to understand, it will also cause to lose out on additional information that is now presented through tables and diagrams.

Among the nonfunctional requirements *NF2* and *NF3* are conditionally met. This means that they are met under certain conditions. Whether *NF3* is met is based on the success of *NF2*. If the system deems a lot of information relevant to the visualisation (either through input files or user choices), the processing power required for requesting data from the simulator to the visualisation framework increases. An increase in processing load in the visualisation framework leads to stuttering in the 2D and 3D visualisation and thus failing to meet requirement *NF3*. However, when only little data is required, visualisations run smooth and *NF3* is met. Finally, nonfunctional requirement *NF6* is currently only met for the implemented timing design simulator. The visualisation framework is implemented through a WebGL system and thus in core a website that is not limited to operating systems. The Software-In-the-Loop (SIL) simulator only operates on a Microsoft Windows operating system and thus the SIL visualisation will not work. The timing design simulator generates files that are operating system independent and thus available cross-platform.

6 | Evaluation and Results

In this chapter the evaluation of the visualisation framework is discussed. Through an evaluation consisting of a user study we aim to answer the following research question:

- *To what extent is the developed visualisation framework able to support the experts in configuring and observing simulators in print system design?*

In the research question we define experts as the user of the developed visualisation framework. The experts are specialists to their domains in specific stages of the project. The number of experts present at any given time is rather small and thus limit the number of participants to the user evaluation. As their tasks are specific to their domain, it is near impossible to address potential scenarios for each participant in a controlled setting. In [15], we were testing for specific interactions with the graphical user interface (GUI) rather than testing the effectiveness of supporting the Océ engineers in their workflow to design print systems. Therefore, we need to apply a different strategy in evaluating the effectiveness of the visualisation framework in supporting the Océ engineers in print system design.

6.1 Method

In the contemporary era of software development, it becomes more common to test the application in the natural setting with either little or no control imposed on the participants' activities with the application. This form of evaluation is referred to as a field study [28]. By applying a field study, the evaluation will more closely resemble a real-world setting in which the interactions of the users can be interrupted or overlapped by other actions. This will provide a setting in which the application would be ultimately used in and thus provides a better sense of how successful the application will be in everyday use. However, it is harder to test for specific hypotheses or provide the same degree of certainty on the interactions of the users with the application compared to a controlled setting.

As the visualisation framework is intended to be used within the workflow of the Océ engineers, it should also be tested in a similar environment. This can be achieved through a field study. The field study is implemented over a time period of a week and aims to verify the success of the visualisation framework. As not every task of

the engineers requires the use of the visualisation framework, providing a time frame of a week will more likely induce the use of the visualisation framework in a natural setting, rather than a shorter period of time (e.g. a day or less).

The visualisation framework is completely novel and shares little or no similarities with the previously available visualisation frameworks, the engineers are provided with an explanation of the visualisation framework beforehand. In addition, a help-page is available to the engineers which explains most features in detail. However, if the engineers still struggle throughout their experience, they are encouraged to contact the researcher for additional explanations regarding their problems. Before the engineers are encouraged to use the visualisation framework within their daily workflow, they are asked to log any issues or feedback they encounter during the field study. If only little feedback is received midway through the field study, the engineers are reminded to log their feedback. If, and only if, the users were not able to use the visualisation framework throughout the time period of a week, the users are presented with a set of tasks tailored to either timing design or Software-In-the-Loop users (these tasks can be found in Appendix B. By the end of the time period, we will visit the engineers in their natural setting and provide them with a questionnaire combined with a short interview regarding their user experience with the visualisation framework. The questionnaire is composed of the System Usability Scale (SUS) in its full length [11]. Alternatives to the SUS are the System Usability Measurement Inventory (SUMI) [19] and the Web Analysis and MeasureMent Inventory (WAMMI) []. In contrast to SUS, the SUMI and WAMMI require a participant count of over 20 in order to provide valuable insights into the working of a system. The SUS proves to be a rigid method for testing the user experience of a system even in a small sample [30]. In table 6.1 the thresholds for each user experience level are provided as determined by Bangor et al. [10]. The SUS is focused on determining the overall usability of the visualisation framework. Therefore, following the questionnaire, a semi-structured interview is performed to acquire additional information about the frequency of use and feedback on functionalities of the visualisation framework. Through the semi-structured interview we can determine how the SUS score compares in frequency of use and the ease of use in terms of completion of tasks and encountered errors. In Appendix A, the questionnaire and semi-structured interview can be found. Results from the user evaluation can be found in Section 6.5.

Table 6.1: Comparison of original SUS threshold and new SUS thresholds

SUS Threshold	User Experience Level
0-50	Not acceptable
50-70	Marginal
70-100	Acceptable

6.2 Participants

The participants are divided into two groups: timing design and Software-In-the-Loop users. Each of the groups is respectively user of the simulator according to their domain. Timing design users mainly concern with the sheet behaviour in print system design. Software-In-the-Loop users concern with the behaviour of the entire print system. The sample size of this evaluation is 6 and consist of 2 timing design users and 4 Software-In-the-Loop (SIL) users. In evaluation, we do not distinct between the two user groups, in reviewing the results we distinct between the two user groups if we see fit. This questionnaire is solely focused on rating the overall usability of the visualisation framework and its total functionality. In addition, the questions from the semi-structured interview are also applicable to both user groups as the discussed functionalities are shared between the different systems.

6.3 System

The evaluation of the visualisation framework consists of testing the proof-of-concept developed throughout this report. In figure 6.1 both domains of the visualisation framework are presented. The visualisation framework uses the Three.js engine to visualize 3D objects and requires a NodeJs native module to connect to the simulators on local machines and is implemented in an Electron shell. The visualisation framework runs on Microsoft Windows machines.

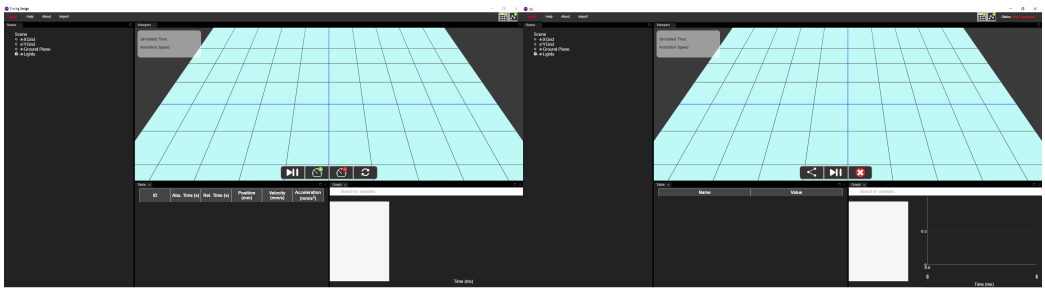

Figure 6.1: The two instances of the visualisation framework presented next to each other. On the left the timing design environment and on the right the Software-In-the-Loop environment. The two user interfaces are mostly similar but differ with regards to the control buttons in the 3D visualisation area and informative text in the menu bar indicating a connection state with the Software-In-the-Loop environment.

6.4 Apparatus

The evaluation is performed on the local desktop computers of the participants operating on Microsoft Windows operating system. User participants use mouse and keyboard throughout the field study in order to interact with the visualisation framework.

6.5 Results

6.5.1 Results from System Usability Scale

We present the results on a per question basis and the overall mean score of the visualisation framework. Figure 6.2 shows a box plot displaying the median scores for each of the questions as well as the standard deviation. On each box, the central mark indicates the median (red), and the bottom and top edges of the box indicate the 25th and 75th percentiles (blue), respectively. The whiskers (dotted vertical line) extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the '+' symbol. The questions have not been normalized, as they have to be in order to calculate the overall SUS score, and represent the values as obtained from the questionnaire. Higher scores in the odd questions and lower scores in the even questions indicate that the visualisation framework was friendly in use. In figure 6.3 the overall scores the users rated the visualisation framework are shown. The box plot shows scores ranging from 70 to 88 with a mean of 78.

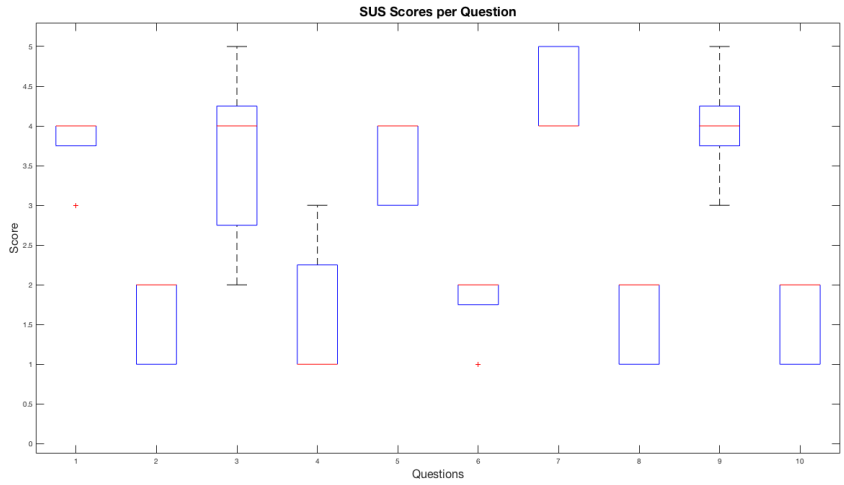

Figure 6.2: A box plot containing the results on a per question basis. The results are not normalized and thus directly correspond with the results obtained from the questionnaire.

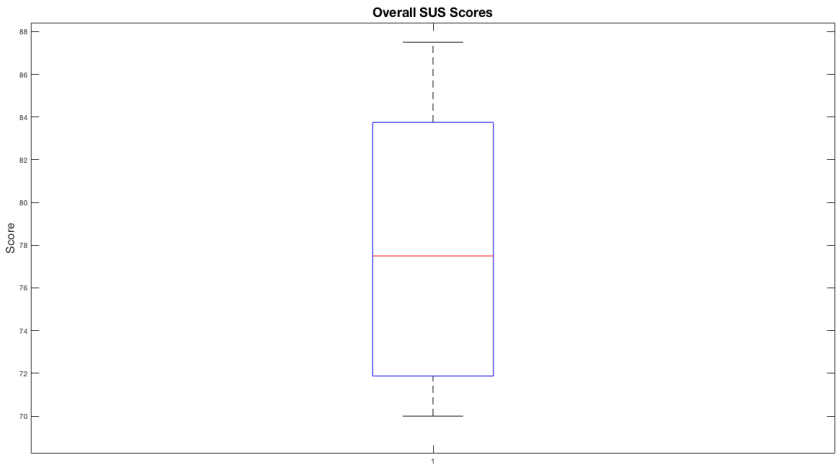

Figure 6.3: A box plot displaying the normalized overall scores as rated by the users of the visualisation framework.

6.5.2 Results from Semi-Structured Interview

Throughout the field study the participants used the visualisation framework on average 3 times (low: 1, high: 15). They noted that the visualisations provide useful insights, but are simply not necessary for every occasion. The users also mentioned that their frequency of use would depend on the stage of the project they are working on.

Mainly Software-In-the-Loop users noted that they were not able to complete their task in all scenario's. When the simulation ended, so would the visualisation framework and thus losing the valuable insights. The timing design users encountered less issues, however one user noted that in some cases there were issues with the input files.

In addition, the Software-In-the-Loop users noticed that the overall interaction with the visualisation framework was smooth, however when they tried to add additional diagrams and tables, the performance would drop and seemed to stutter.

All of the users were satisfied with the implementation of the visualisation of 3D objects in the visualisation framework. It provided useful insights that would otherwise be hard to grasp, such as displaying sheet numbers and the front and rear side of the sheet. Another example given was the insight into the print head movement that was not fully completing its planned trajectory. In addition, the implementation of an adjustable graphical user interface appealed to the users and allowed them to focus on the task at hand by organizing the windows accordingly.

As the disciplines in which the users are active differ, all user provided different suggestions for missing features. The Software-In-the-Loop would like to see a direct connection between the different components of the visualisation. In addition, a direct

transformation from CAD model to importable file format to the 3D visualisation would be a welcomed feature. The timing design users were mainly concerned with displaying additional information with regards to sheet behaviour. They proposed a feature for displaying information about inter-sheet relations and extension of the diagramming facility to subtract values.

Among the participants, half of them would actively contribute to the development of the visualisation framework to improve its capabilities to support the Océ engineers in configuring and observing various simulators.

Finally, the participants were asked to rate the visualisation framework. The visualisation framework scores an average of 7.3 out of 10 with the main comment that it satisfies their direct needs but requires overall improvement.

7 | Discussion

In this chapter we discuss the results from the user evaluation in Section 7.1. In Section 7.2 we reflect on the entire project.

7.1 Discussion of Results

The users evaluated the visualisation framework positively with an average SUS score of 78 and average grade of 7,3 out of 10. If we look at the scores for the individual questions in figure 6.2 we can see that the visualisation framework scored positively throughout the various questions. Question 1 (min: 3, max: 4, median: 4), *I would like to use the system frequently*, scored relatively high with 4 out of 5. However, if we compare this to the results from the field study, we can see that throughout the time frame of the field study they have rarely used the visualisation framework. The participant also mentioned that the frequency of use is highly dependent on the stage of the project and thus they might foresee more frequent use in the future. Question 3 (min: 2, max: 5, median: 4) *I thought the system was easy to use*, shows some disagreement, however not caused by the complexity of the system (question 2; min: 1, max: 2, median: 2) nor the fact that the users would need assistance from a technical person (question 4; min: 1, max: 3, median: 1). In addition, the users indicated that they did not need to learn a lot before they could use the system (question 10; min: 1, max: 2, median: 2) nor that they would expect most people to have troubles learning the system (question 7; min: 4, max: 5, median: 4). Finally, all participants indicated that they felt confident in using the system (question 9; min: 3, max: 5, median 4). Scores from the previously mentioned questions are not in line with the scores in question 3. Only one user rated one questions (question 3) below the average score (a score of 3 marks the average) of the SUS. We expect that this user in particular encountered an issue specific to his domain, looking at the result from the semi-structured interview it is most likely the user missed a direct connection between CAD models and the visualisation framework.

The remainder of the questions (5, 6 and 8) are with regards to integration of functionalities and graphical appearance. In question 5 (min: 3, max 4, median: 4), *I found the various functions of the system were well integrated*, the participants indicated that the features were well integrated. In the semi-structured interview

they indicated similarly, but saw a set of features missing from their required toolset. The graphical appearance, as topic of the research to this report [15], was rated to be consistent (question 6; min: 1.75, max: 2, median: 2) and did not appear cumbersome to the participants (question 8; min: 1, max: 2, median: 2).

Overall, the visualisation framework scored positively in both the SUS and the semi-structured interview. As the average score of the SUS (min: 70, max: 88; median: 78) indicates, it is well within the range of acceptable (70-100), but can still improve in order to achieve a higher level of user experience. Similar results were obtained from the semi-structured interview, where the participants indicated to be overall pleased with the visualisation framework but were missing certain features that would complete their toolset.

7.2 Project Discussion

The results indicated that the users were satisfied with the visualisation framework as an addition to their toolset for print system design. However, in order to fully support the Océ engineer's improvements and extensions of specific functionalities are needed. These weaknesses only came to light when a larger user group was involved in testing the features. During the development of the visualisation framework, focus to the improvement and extension of these features could have been resolved by including a larger user group during the iterative development of the software. In the current situation (mentioned in Section 5.1), a single user presented his personal needs for improvement of the visualisation rather than a larger pool that would provide insights on structural changes required. When extending the user pool for intermediate releases, more time has to be dedicated to retrieving and analysing feedback rather than developing the tool.

As already mentioned, the users that participated in the user study showed interest in using the visualisation framework. But, a relatively small user group participated in the user study which causes the results to be inaccurate. A larger user group would provide a better indication in the overall adoption of the visualisation framework throughout the disciplines. An issue with increasing the user group, however, would be that users other than specialists would also be included which in turn might provide results that do not reflect on the main goal of the current version of the visualisation framework. Extending the visualisation with additional features and functionalities (possibilities for these features and functionalities are presented in Chapter 9), to not only adhere to specialists, would increase the effective user group.

8 | Conclusion

In this report a visualisation framework for print system development at Océ technologies was developed and evaluated, resulting in adoption of the visualisation framework in the workflow of the Océ engineers.

In the research prior to this report [15], we developed an understanding of what visualisations and graphical user interface (GUI) aspects are needed by the Océ engineers. In this report we focused on developing the visualisation and the features accompanying the GUI developed in the research prior to this report.

During the development of the visualisation framework we adopted an iterative development strategy. The benefits of applying an iterative development strategy are: the detection of design breakage early on, applying a modular architecture; and early and frequent involvement of the users in the development process. Because the users were involved early and frequently in the early stages when determining the initial requirements focussing on the overall user experience, and a single user throughout the development of the visualisation framework, requirements were removed, extended or added while the development of the visualisation framework was ongoing. This proved extremely powerful as many of the requirements were not known before the project started. In the design of the visualisation framework a modular architecture was adopted to increase maintainability and extensibility. Through an interface simulators can connect to the visualisation framework which is componentized into components each with their dedicated task. Components such as a 3D visualisation area, diagram and table were developed to aid the Océ engineers.

To verify that the developed visualisation framework was in accordance with the expectations of the Océ engineers and indeed supports the engineers in configuring the simulators, we applied a field study. In the field study we provided the engineers with the visualisation framework and encouraged them to apply it in their daily workflow. Throughout the field study we received feedback with regards to certain features or design errors that could require fixing. We present possible solutions to these weaknesses in Chapter 9.

Results from the evaluation showed that the engineers were generally pleased with the visualisation framework and proved to be a rigid solution to implement in their workflow. Therefore, we can conclude that the visualisations and user interface present in the visualisation framework proves sufficient to support the engineers in configuring and observing simulators throughout print system design.

9 | Future Work

The visualisation framework was generally well received, but software development is never finished. Therefore, we present recommendations for further research and recommendations for further development of the visualisation framework.

9.1 Recommendations for Future Research

As we noted in Chapter 7, there are improvements possible regarding the inclusion of the users throughout the project. For future work, we therefore recommend increasing the user pool used throughout the development of the system. A larger pool results in more variety in feedback and provides insight in structural improvements within the system.

Similarly, we discussed that the results may or may not be accurate for a larger user group. Therefore, we suggest increasing the scope of the visualisation framework and thus increasing the effective user group that can be used to evaluate the effectiveness of the visualisation framework in supporting the Océ engineers in configuring and observing simulator for print system development.

We now implemented the field study over a time period of a week. This resulted in only a limited use of the visualisation framework by many of the participants of the field study. By applying the field study to a stage of development in which the visualisations add to the productivity of the Océ engineers and increasing the time frame in which the field study takes place should also increase the accuracy of the results.

Finally, other domains should be research to which a visualisation framework could provide support. Potential areas within Océ which could benefit from an extension of the current visualisation framework are maintenance and training and analysing data obtained from customers. In addition, a virtual reality implementation to the previously mentioned extensions could improve the effectiveness with which the data can be analysed.

9.2 Recommendations for Further Development

In the current state of the visualisation framework, there is support for two types of simulators: timing design and Software-In-the-Loop (SIL). However, within Océ many more simulators and models are used for print system development. Therefore, additional interfaces could be implemented to support more simulators and thus extend the accessibility of the visualisation framework to more disciplines.

Similarly, with the addition of more simulators, more or extended visualisation components or are required to fulfil the needs of the Océ engineers. Therefore, we propose that particularly the present 2D visualisation components are extended to support a wider variety of data representations. In addition, specific features requested by the Océ engineers include additional information on inter-sheet behaviour and a work flow to import CAD files directly to the visualisation framework.

A downside of the modular architecture is that the components are currently independent of the other components. However, users noted that interaction between the different components would highly increase the effectiveness of the visualisation framework. An example would be that users can select an object in the 3D visualisation area, resulting in selecting the accompanying variable and displaying this in a diagram.

Finally, we already mentioned this in section 5.6 that under certain conditions the performance of the visualisation framework decreases. Therefore, in order to maintain a smooth visualisation throughout, the performance regarding the requesting of variables from the SIL simulator could use a boost.

Bibliography

[1] Node.js. https://nodejs.org/en/.

[2] Webgl. https://www.khronos.org/webgl.

[3] Blender. https://www.blender.org/.

[4] Electron. https://electron.atom.io/.

[5] Goldenlayout. https://golden-layout.com/.

[6] Océ. http://oce.com/.

[7] React. https://facebook.github.io/react/.

[8] Three.js. https://threejs.org/.

[9] Unity. https://unity3d.com/.

[10] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[11] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[12] Lan Cao and Balasubramaniam Ramesh. Agile requirements engineering practices: An empirical study. *IEEE software*, 25(1), 2008.

[13] Dai Clegg and Richard Barker. *Case method fast-track: a RAD approach*. Addison-Wesley Longman Publishing Co., Inc., 1994.

[14] Ujjal Dey, Pabitra K Jana, and CS Kumar. Modeling and kinematic analysis of industrial robots in webgl interface. In *Technology for Education (T4E), 2016 IEEE Eighth International Conference on*, pages 256–257. IEEE, 2016.

[15] Yannick Donners. 3d interactive print visualisation. Internal Report Research Topics, University of Twente, 2017.

[16] Tracy Fullerton, Chris Swain, and Steven Hoffman. *Game design workshop: Designing, prototyping, & playtesting games*. CRC Press, 2004.

[17] L. Soumers H. Hunnekens, E. Schindler. Multidisciplinair modelleren dagelijkse praktijk. `https://www.bits-chips.nl/artikel/multidisciplinair-modelleren-dagelijkse-praktijk-46100.html`.

[18] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 51:915–929, 2015.

[19] Jurek Kirakowski and Mary Corbett. Sumi: The software usability measurement inventory. *British journal of educational technology*, 24(3):210–212, 1993.

[20] J Rodríguez Lloret, Nancy Omtzigt, Eric Koomen, and FS De Blois. 3d visualisations in simulations of future land use: exploring the possibilities of new, standard visualisation tools. *International Journal of Digital Earth*, 1(1):148–154, 2008.

[21] Sus Lundgren. Cover story: Designing games: Why and how. *interactions*, 15 (6):6–12, November 2008. ISSN 1072-5520. doi: 10.1145/1409040.1409042. URL `http://doi.acm.org/10.1145/1409040.1409042`.

[22] Jakob Nielsen. Heuristic evaluation. *Usability inspection methods*, 17(1):25–62, 1994.

[23] Jean-Baptiste Pettit and John C Marioni. bioweb3d: an online webgl 3d data visualisation tool. *BMC bioinformatics*, 14(1):185, 2013.

[24] Matti Pouke, Timo Koskela, Şan Güneş, Matti Matero, Karri Ojala, Jukka Pajukangas, Niko Pietikäinen, and Timo Ojala. 3d visualization of a public transportation system. In *Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia*, pages 377–379. ACM, 2016.

[25] Nicholas Rego and David Koes. 3dmol. js: molecular visualization with webgl. *Bioinformatics*, 31(8):1322–1324, 2015.

[26] N. Roos. Multidisciplinair software ontwikkelen op een virtuele printer. `https://www.bits-chips.nl/artikel/multidisciplinair-software-ontwikkelen-op-een-virtuele-printer-45358.html`.

[27] Walker Royce. Improving software economics. *Application development trends*, 2009.

[28] Helen Sharp, Yvonne Rogers, and Jenny Preece. Interaction design: beyond human-computer interaction, 4th edition. 2015.

[29] Ben Shneiderman and Catherine Plaisant. Designing the user interface: Strategies for effective human-computer interaction. *ACM SIGBIO Newsletter*, 9(1):6, 1987.

[30] Thomas S Tullis and Jacqueline N Stetson. A comparison of questionnaires for assessing website usability. In *Usability Professional Association Conference*, pages 1–12. Citeseer, 2004.

[31] Yin Zhang and Sonali Kudva. E-books versus print books: Readers' choices and preferences across contexts. *Journal of the Association for Information Science and Technology*, 65(8):1695–1706, 2014. ISSN 2330-1643. doi: 10.1002/asi.23076. URL `http://dx.doi.org/10.1002/asi.23076`.

A | Questionnaire and Semi-Structured Interview

System Usability Scale (SUS)

Please rate the following questions on:
Strongly Disagree – Disagree – Neutral – Agree – Strongly Agree

I think I would like to use this system frequently.	☐	☐	☐	☐	☐
I found the system unnecessarily complex.	☐	☐	☐	☐	☐
I though the system was easy to use.	☐	☐	☐	☐	☐
I think that I would need the support of a technical person to be able to interact with this system.	☐	☐	☐	☐	☐
I found the various functions of the system were well integrated.	☐	☐	☐	☐	☐
I thought there was too much inconsistency in this system.	☐	☐	☐	☐	☐
I would imagine that most people would learn to use this system very quickly.	☐	☐	☐	☐	☐
I found the system very cumbersome to look at.	☐	☐	☐	☐	☐
I felt very confident using the system.	☐	☐	☐	☐	☐
I needed to learn a lot of things before I could get going with this system.	☐	☐	☐	☐	☐

Semi-Structured Interview

How often have you used the visualization framework?

How many hours have you approximately spent with the visualization framework?

Were there occasions in which you were not able to complete your task?

Have you encountered any errors?

Which functionality/functionalities stood out to you?

Which functionality/functionalities were missing?

Would you extend the visualization framework with additional functionalities?

How would you rate the visualization framework?

B | User Tasks

HappyFlow

1. Load the [redacted], [redacted] and [redacted].
2. Disable the visibility of the PointsOfInterest.
3. Start the visualization.
4. At timestamp 19 seconds, stop the animation.
5. Inspect the relation in time-distance between sheet 4 and 12 in the diagram.
6. Add a diagram to the graphical user interface.
7. Select the velocity variable from sheet 31 in the newly added diagram.
8. Step through the animation until timestamp 21.
9. Find the position of sheet 45 in the table.
10. Add a table to the graphical user interface.
11. Define the distance between sheet 38 and 50.
12. Resume the visualization.
13. When the visualization has ended, reset the visualization.
14. Start the visualization at animation speed 4.

Software-In-the-Loop

1. Load the [redacted] 3D objects and the corresponding VariableMapping.
2. Disable the visibility of the ground plane.
3. Start the SIL simulation (background process).
4. Connect the visualization to the SIL simulation.
5. In the diagram select the variable [redacted].
6. Add a diagram to the graphical user interface.
7. Select the variable [redacted] in the newly added diagram.
8. Terminate the SIL simulation.

C | Requirements from Prior Research

Below the requirements are presented as found prior research ([15]). These requires have been subject to change throughout the development of the novel visualisation framework.

C.1 Functional Requirements

Functional Requirements	
Integration	
F.REQ_001	Integration with multiple simulators.
Rationale	*Multiple simulators must be able to connect to the visualisation. Within the embedded software department, multiple simulators are used to understand the designed printing behaviour. In order to reduce the learning curve and maintainability of the visualisation tool, multiple simulators are to connect to one tool with a single codebase.*
F.REQ_002	Flexibility in using multiple printing systems.
Rationale	*Throughout Océ multiple printing systems are designed and require thorough testing. Therefore, the visualisation tool must be able to use the paperpath from different printing systems.*
F.REQ_003	Load/Save Configuration file
Rationale	*It must be possible to save the configuration state of the visualisation tool to a configuration file which can be loaded into the visualisation tool again on startup.*
2D Visualisation	
F.REQ_004	Add multiple attributes to a plot.
Rationale	*It must be possible to display multiple attributes in a single plot. The number of attributes which can be added to a plot should not be limited.*
F.REQ_005	Support for multiple scales on a plot.

Rationale	*Add the possibility to apply multiple scales on a plot widget. When two attributes with huge varying scales are displayed in single plot it is impossible to see them both. Therefore multiple scales to a plot could be added.*
F.REQ_006	Assign multiple plots to a visualisation area.
Rationale	*A visualisation area is a base where plots are to be displayed. It should be possible to allowing multiple plots to be shown simultaneously in a single visualisation area. This allows for easy comparison of multiple attributes with different scales.*
F.REQ_007	Crosshair support
Rationale	*Within a plot the cursor should be changed to a cross-hair allowing for exploration of attributes. The crosshair should also provide the option to zoom in on a selection and resume it's normal behaviour on a selected area.*
F.REQ_008	Hide/Show attributes in a plot.
Rationale	*It must be possible to hide/show attributes in a plot.*
F.REQ_009	Auto/Manual scaling in plot widget.
Rationale	*A toggle should be present to alter between automatic and manual scaling of plot.*
F.REQ_010	Make data on visualisation area persistent.
Rationale	*Present plots in a visualisation area could be stored persistent. When the visualisation tool is closed and started again, visualisation area should be in the same exact state as on closing the visualisation tool.*
3D Visualisation	
F.REQ_011	Keyboard and mouse navigation
Rationale	*The 3D visualisation must be able to be controlled with a keyboard and mouse. The required navigations contains zooming, rotating and panning the 3D visualisation.*
F.REQ_012	Loading objects
Rationale	*On the basis of a set of a pre-generated files, 3D objects must either be created or loaded into the 3D visualisation. E.g. CAD files must be able to be loaded into 3D visualisation.*
F.REQ_013	Hide/Show objects
Rationale	*In the 3D visualisation objects must have a toggle to either hide or show them. Allowing to do so will allow to remove potential clutter for certain tasks.*
F.REQ_014	Select objects through the 3D visualisation.
Rationale	*One might want to select an object in 3D visualisation by simply clicking on the object. This could then lead to allow for other actions to be performed (e.g. Hiding/Showing of the object).*
F.REQ_015	Direct link between 3D visualisation and 2D visualisation.

Rationale	*When an object is selected in the 3D visualisation, it could directly translate to displaying the attributes of that object in the 2D visualisation area. E.g. on selection of a pinch (rubber roller for transportation of a sheet), the linked attributes are displayed in the plot.*
F.REQ_016	Altering the transparency of objects.
Rationale	*To create a better overview of the 3D visualisation, one might prefer to alter the transparency of certain objects to see-through and explore objects behind or within.*

C.2 User Requirements

User Requirements	
Interaction	
U.REQ_001	Fluid Interaction
Rationale	*Users must experience a fluid interaction with direct feedback to their actions. E.g. when they navigate through the 3D visualisation, there must be an immediate response to their action.*
U.REQ_002	Shortcuts
Rationale	*To increase efficiency with which frequent users can navigate through the GUI a set of shortcuts must be defined.*
U.REQ_003	Clear link between actions in different windows
Rationale	*When a users performs an actions in a window, this action should also reflect in other windows. E.g. an user toggles a visibility parameter in the list of objects in a certain which should cause an object in the 3D visualisation to either hide or show.*
Appearance	
U.REQ_004	Consistency
Rationale	*Throughout the Graphical User Interface (GUI) consistency must to present in colors, icons and actionable elements. Consistency will decrease the confusion users might experience.*
U.REQ_005	Adjustable layout
Rationale	*The different interface elements could be resizable, draggable and preferably dockable. This allows the user to choose their own preferred layout for their specific task.*
U.REQ_006	Adjustable GUI
Rationale	*Allowing the user to choose a color scheme to their liking. E.g. a dark color scheme or a light.*