EngD THESIS REPORT

Automatic and Optimized Hardware Sequence Generation and Inline Interpretation

The design and implementation of tools for reducing the development and deployment efforts of YieldStar's hardware action-scheduling software

Georgios Evangelou
October 2023
Department of Mathematics & Computer Science

EngD SOFTWARE TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Automatic and Optimized Hardware Sequence Generation and Inline Interpretation

The design and implementation of tools for reducing the development and deployment efforts of YieldStar's hardware action-scheduling software

Georgios Evangelou

October 2023

Eindhoven University of Technology
Stan Ackermans Institute – Software Technology

EngD Report: 2023/068

Confidentiality Status:
Public version

Partners

ASML Netherlands B.V

Eindhoven University of Technology

Steering Group

ir. Tim Kouters (ASML)
dr. Zhifeng Sheng (ASML)
dr. ir. Stef van den Elzen (TU/e)

Date October 2023

Composition of the Thesis Evaluation Committee:

Chair:	prof. dr. Alexander Serebrenik (TU/e)
Members:	dr. Zhifeng Sheng (ASML)
	dr. ir. Stef van den Elzen (TU/e)
	dr. Yuwei Jin (ASML)
	ir. Harold Weffers, EngD (TU/e)

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Contact Address
Eindhoven University of Technology
Department of Mathematics and Computer Science
MF 5.072, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31 402474334

Partnership
This project was supported by Eindhoven University of Technology and ASML.

Published by
Eindhoven University of Technology
Stan Ackermans Institute

EngD-report
2023/068

Preferred reference
G. Evangelou, Automatic and Optimized Hardware Sequence Generation and Inline Interpretation. Eindhoven University of Technology, EngD Report 2023/068, October 2023

Abstract
YieldStar is the main metrology machine produced by ASML. It scans wafers previously exposed by lithography machines in order to calculate important lithographic metrics on their surface and help improve wafer quality and yield. Acquiring images on wafers is a complex process; engineers spend much manual effort (graphically) modelling and implementing (into code) the behavior of the machine.

In this project, we proposed, designed, and implemented two machine-type agnostic tools that automate and optimize this development procedure. The Sequence Design and Optimization Tool (SDOT) uses a requirements-based input to produce and visualize optimal action sequences, utilizing constraint programming algorithms to schedule actions for machine sensors and actuators. The Inline Sequence Interpretation Tool (ISIT) translates the action sequences into hardware commands, which are then queued to the hardware peripherals of YieldStar.

SDOT and ISIT have already been demonstrated for the latest machine type (YS-500) and showed tangible improvements over the existing methods. Furthermore, they will be used for the software implementation of the next product (YS-550) and it is expected that they will help decrease the implementation and deployment time of the scheduling module by 50% and 75% respectively.

Keywords
action scheduling; job-shop scheduling; synchronization; hardware sequence generation; constraint programming; optimization; visualization; ASML; TU Eindhoven; EngD.

Foreword

Within ASML, besides making lithography systems like Twinscan, we also make metrology systems like YieldStar. YieldStar allows the measurement of on-product overlay and focus, using diffraction-based technology, on specific overlay and focus targets. To be able to measure the on-product overlay or focus offset on such targets, YieldStar needs to move a set of machine hardware peripherals (such as motion axes, shutters, cameras) in sequence to acquire images of the specially designed targets on the product wafer, while optimizing for throughput.

At this moment, the sequence of hardware actions are designed and implemented manually. This is both time consuming and error-prone. Even with all this trouble, an optimal sequence is not guaranteed, as the design process mostly relies on human ingenuity. Hence, I proposed an OOTI project to automate the design, implementation and deployment of the hardware sequence. The goal was to save valuable design effort, while guaranteeing the sequence to be optimal.

I am very happy to have had Georgios as the OOTI trainee to work on this project. He worked hard and cleverly to design and implement the tools for automating the hardware sequence generation. I am very impressed by the result he delivered and I am happy to see that this set of tools will be used in the next generation of YieldStar machines.

I am also very happy that Georgios will join ASML soon. I wish him all the best in ASML.

Zhifeng Sheng
September 2023

Preface

This is the public version of the original EngD thesis that is permanently confidential. Therefore, some sections of this document have been edited or removed.

The purpose of this report is to provide a detailed overview of the final project of my Software Technology (ST) Engineering Doctorate (EngD) traineeship, conducted for ASML and the Computer Science and Mathematics Department at the Eindhoven University of Technology. ST EngD is a two-year doctorate-level program provided by the Eindhoven University of Technology under the banner of the 4TU.School for Technological Design, Stan Ackermans Institute.

This project took place in the Software Layout (FC-120) group within ASML, which is concerned with defining and improving the software design of the YieldStar platform and products. YieldStar is a line of ASML products concerned with optical metrology. The goal of the project was to automate and optimize the process of designing, implementing, and deploying a part of the behavior of YieldStar machines concerned with performing optical measurements on wafers, shortening development time, and increasing machine throughput.

In this thesis, we introduce the context and objectives of the project, the system designed and implemented, and, finally, the results and findings throughout this journey. This report was compiled during the project as a live documentation of my progress, serving as an organized artifact with chapters for different readers. More specifically:

- The **Executive Summary** is a high-level overview of the project, aimed at managers and executives, summarizing the business needs behind the project, its results and future.
- **Chapter 1** provides sufficient background to establish the domain of the problem under investigation for readers with no affiliation to ASML, lithography, and metrology.
- The problem itself is analyzed thoroughly in **Chapter 2**, which is more intended for people with pre-existing domain knowledge.
- **Chapter 3** describes the main stakeholders involved in the project and can provide insights into where an EngD trainee can look in order to harness knowledge, guidance, and needs from relevant experts.
- **Chapters 4 and 5** are more focused towards eliciting requirements from stakeholders and formulating the system's architecture. Thus, this chapter is more relevant to software/system architects.
- **Chapters 6 and 0** are concerned with the implementation of the system into code and how it was tested against the requirements and target scenarios.
- **Chapters 8, 9, and 10** are again less technical, with **Chapter 8** focusing on the conclusions, findings, and suggestions for future work, **Chapter 9** describing the processes and organization of the project, and **Chapter 10** documenting some personal conclusions from my efforts.

Georgios Evangelou
September 2023

Acknowledgements

A long and challenging journey comes to an end. Months of discussions, thinking, trials-and-errors, and, of course, excitement and stress bring this project to a high-note conclusion. During this fruitful adventure, which was both inspiring and demanding, I was accompanied by mentors, colleagues, friends, and loved ones. Each of them contributed to the "well-being" of the project or me in their own unique way, motivating me at my high moments and lifting me upwards at my low points.

My ASML supervisor, **Zhifeng Sheng**, was the "man on the frontline" beside me. He was always there to guide, simplify, explain, suggest, and challenge. We spent countless hours together discussing and brainstorming on the project. With his ingenuity, Zhifeng made me steadily confident about the project, since I knew that there were never issues that we could not solve together. He also showed trust and made me feel optimistic about the opportunities ahead.

Stef van den Elzen was a "calm force" as we say in Greek; a composed mentor who was never short of ideas and mature guidance. As the TU/e supervisor, Stef was there to provide fruitful advice, inculcating how I could best handle my focus and what ideas I could materialize. When needed, he pushed me to go a bit further, and he instilled more creativity and elegance in my results.

With his sharp mind and relaxed style, **Tim Kouters** (our ASML group leader) was there when needed, to provide mature and balancing suggestions on how to tackle managerial challenges. Seeing the bigger picture from a group leader's perspective, Tim helped me realize the potential and boundaries of my project: what can be done and what should be done.

During this effort, I was also accompanied and guided by countless other experts that provided feedback and fueled the project with ideas. I would like to give special regards to **Yuwei Jin**, **Joost Verkooijen**, **Melchior Hofman**, **Wim Willems**, **Andrea Pasqualini**, **Omneya Siam**, and **Jolanda Schagen**. Each one contributed in making my efforts more meaningful and successful.

I would also like to thank my EngD **colleagues and friends**, as well as **Yanja Dajsuren**, **Desiree van Oorschot**, **Karin Majoor**, and **the coaches** of the Software Technology EngD. They supported me with their own unique way in my development and well-being, shaping an experience that was both beneficial and enjoyable.

Additionally, I would like to heartwarmingly express my appreciation to **my family and friends**, who accompanied me during this challenging journey, cheering for my successes and "hugging" me when I needed to "catch a breath." They gave me the hope, they gave me the love, and they gave me the company that I needed, to work hard towards my goals and ambitions. I would not feel the same confidence and energy without their presence, affection, and interest.

Last but not least, I would like to convey my appreciation to the Thesis Evaluation Committee members **Alexander Serebrenik** and **Harold Weffers** who, alongside Zhifeng, Stef, and Yuwei, spent considerable time reviewing my thesis and grading my efforts.

Georgios Evangelou
September 2023

Executive Summary

The Acquirement Hardware Sequence Generator (aHSG) is a software component of YieldStar. It translates high-level instructions into commands executed by hardware peripherals (inside YieldStar) responsible for preparing and performing acquirements (i.e., optical measurements) on wafers. The Move Acquire (MA) sequence is a diagram that describes this behavior of the machine and the aHSG is its software implementation.

Designing an MA sequence and implementing it into an aHSG every time a new YieldStar machine type is introduced is a tedious, year-long process that involves a lot of manual effort and relies solely on human reviews and ingenuity. It requires multiple revisions and tests to identify errors and optimize throughput. Furthermore, since this procedure consumes and produces many artifacts (i.e., documents, models, and code), it is almost impossible to ensure absolute consistency between them. ASML aims to improve this procedure, thereby reducing the human effort needed and improving the throughput of the aHSG-related machine behavior.

With this project, we aimed at improving and automating the workflow of introducing new or revised aHSGs in YieldStar machines. We created a requirements-based software platform, which automatically generates sequences of hardware commands, optimized for throughput. The system minimizes human intervention in the process, while providing automatic verification and visualization mechanisms to ensure the correctness of the sequenced commands.

The software platform developed for this project consists of two tools:

- Sequence Design and Optimization Tool (SDOT): By providing this (standalone) tool with a requirements-based input, Functional Designers (FDs) use it to automatically generate and optimize MA sequences. SDOT also shows diagrams of the expected behavior and statistics for each hardware peripheral. Finally, it exports the sequence as a file to configure ISIT.
- Inline Sequence Interpretation Tool (ISIT): ISIT is a software component integrated in YieldStar and it is configured by the sequence generated by SDOT in order to translate acquirement requests into hardware commands. In other words, Software Developers (SDs) shall use this software module as a replacement for traditional aHSG implementations.

The developed tools were configured and demonstrated for the current YieldStar machine, YS-500. We showed that they achieve the following main goals:

- They facilitate consistency between the artifacts of the procedure, binding the requirements with the MA sequence and the MA sequence with the machine behavior.
- They unravel additional throughput improvements for the MA sequence, initially missed by manual design
- They can reduce the workload needed to develop a completely new MA sequence from approximately one year to five weeks (rough approximation, 90% improvement).
- They reduce the workload needed to implement a new MA sequence into a machine type from two days to one day (50% improvement).
- They reduce the workload needed to deploy different MA sequences on a machine, from two hours to thirty minutes (75% improvement).

We recommend that the workflow implemented within this project be introduced to the development of the aHSG of the upcoming YS-550, as well as future machine types. Stakeholders (i.e., FDs, SDs, and other engineers) are already enthusiastic to use the tools to automate their work and potentially increase the expected performance of YieldStar machines. In fact, SDOT is already being used to verify and optimize the design of the MA sequence for YS-500, which is the most recent YieldStar machine type in production. Finally, the two tools developed under this project could also be adapted and used to automate and optimize other scheduling processes as well. One such example is the Wafer-Exchange (WEX) sequence, which is a similar concept to the MA sequence.

Table of Contents

List of Figures

List of Tables

1 Introduction

This chapter aims to familiarize the reader with the business and technology context surrounding this project. The reader is provided with general knowledge on the company, its affiliation with lithography and metrology, and YieldStar, one of its metrology solutions. Finally, an analysis on the Move Acquire (MA) sequence and the Acquirement Hardware Sequence Generator (aHSG) is provided.

1.1 ASML

ASML is a technology goliath that currently drives lithography, nanoelectronics, and our world forward. It is the world leader of lithography machines, also known as scanners, that produce much of the electronic chips (such as processors and memory components) that we use in our computers, phones, cars, and almost every other advanced technological device. Some of the major customers of ASML include TSMC, Intel, and Samsung. The company is based in Veldhoven, The Netherlands, and is spread across various locations around the world.

The manufacturing of chips is a very complex and technologically demanding process. Unimaginable precision is required to ensure that the chips produced by the ASML machines live up to the required standards and that near optimal yield is achieved. Low quality means huge costs for ASML's clients and lengthy delays for the supply line. Thus, ASML puts a lot of effort in enhancing the quality of its lithography machines and processes.

In order to facilitate this need, ASML has been developing tools (machines and software) that complement its lithography machines and ensure that any inaccuracies or erroneous structures in their output are detected and dealt with on time. Such activities are carried out using optical metrology (YieldStar platform), E-beam metrology (HMI platform), and computational lithography (e.g., Brion). These tools scan the wafers produced by the lithography machines, provide insights on the structures etched or developed on top of them, and/or help finetune the lithography machines for the next wafers that they process.

1.2 Metrology using YieldStar

As mentioned earlier, YieldStar is the product platform concerned with the optical metrology within ASML. YieldStar machines are placed inside chip-manufacturing fabs, alongside lithography machines, either as standalone or integrated solutions (Figure 1). A YieldStar machine is periodically fed with etched or developed wafers, it aligns them properly, and it inspects them with optical sensors following diffraction-based methods (Figure 2). The wafers are scanned at specific spots inside or between the printed structures. These spots are called targets.

Figure 1: Standalone and Integrated YieldStar solutions
Sources: [1], [2]

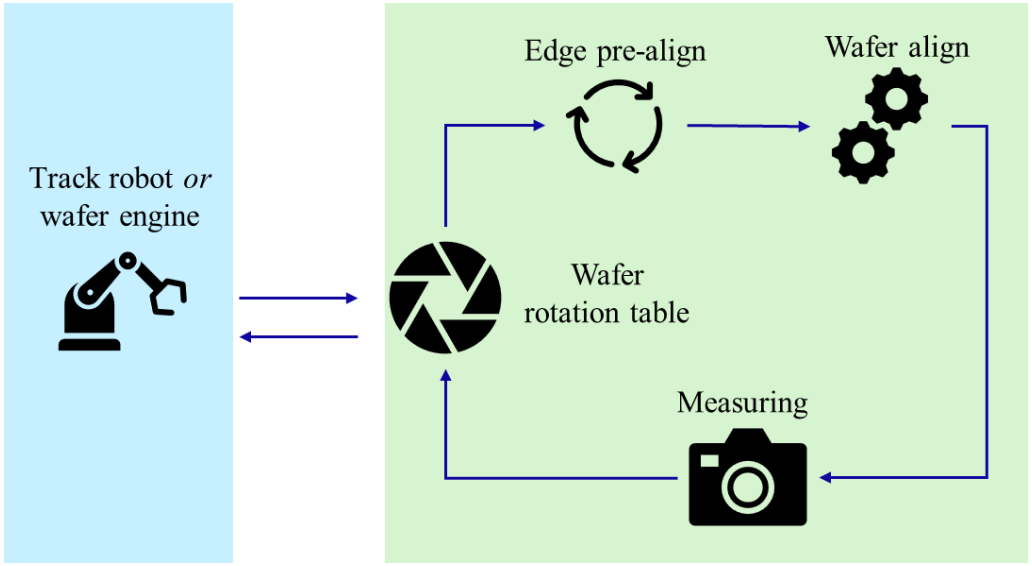

Figure 2: A wafer's path inside YieldStar

By inspecting the targets on the wafer under inspection, the machine estimates lithography-related metrics. Such metrics are associated with the Overlay (OV), the Critical Dimension (CD), the Side Wall Angle (SWA), and the Focus. Maps of these metrics are then composed and facilitate the capability to finetune the configurations of the lithography machines.

To summarize, YieldStar essentially performs diagnostic measurements on wafers so as to monitor and improve the production quality of the lithographic scanners.

1.3 The MA sequence and aHSG

This section was edited for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

Performing metrology measurements on a wafer is a process that consists of various steps and in this section, we look at it from a high abstraction level. After a wafer is inserted in the machine, it needs to be aligned first. Then, hardware resources such as lenses, stages, and illumination sources need to be calibrated to prepare for the acquirement. An acquirement is the final step in the procedure; one or more cameras inside YieldStar capture the light patterns diffracted by illuminated targets on the wafer. Figure 3 illustrates some steps of this process.

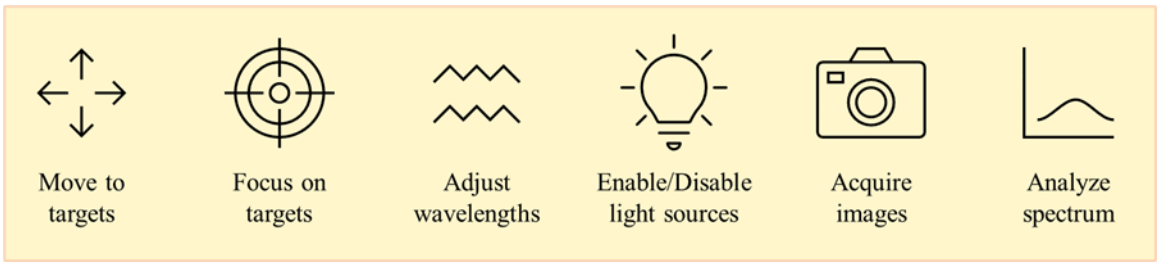

Figure 3: Steps needed to prepare and execute an acquirement

Finally, the captured images are used to compute the OV, CD, SWA, and Focus maps. Multiple hardware peripherals inside the machine need to execute sequences of commands in a coordinated manner in order to perform the above steps and variations of such. Most of these steps are formulated in what is called the Move Acquire (MA) sequence.

The MA sequence is the workflow that describes the actions that entail the steps to prepare for and execute acquirements. It denotes the order, conditions, and precedence constraints of these actions, as well as the sensory and actuator subsystems that are assigned to execute them. Therefore, the MA sequence is closely bound with the specifications, devices, and performance of the machine.

Typically, a new variant of the MA sequence is introduced whenever a new YieldStar machine type is developed, since there may be different requirements, peripherals, and goals, compared to previous iterations of YieldStar. Updates to the sequence of an existing product may also happen, as engineers find ways to optimize it. Both creating and updating a sequence are done manually.

After it has been designed, the MA sequence is implemented into a software module, the Acquirement Hardware Sequence Generator (aHSG). aHSG is responsible for translating high-level instructions into the sequences of hardware commands that will be executed by the machine's peripherals to facilitate the behavior described by the MA sequence and configured by the high-level request. For each machine, the MA sequence and aHSG are closely connected and any changes should be "synchronized" between them.

1.4 Project context

The automation and optimization of the design of the MA sequence and its implementation into the aHSG are the subjects of this project. As later analyzed in Chapter 2, this process poses many challenges: it is done manually, it is complex, and it does not ensure that the resulting system behavior is optimal in terms of throughput. The main drawback is its total development timespan, which is more than one year as seen in Figure 4. Through this project, we designed and implemented a solution that improves this procedure, making it automatic and optimized.

The business goals that drove the project are two:

- Decrease the work hours needed to design and implement the sequence.
- Improve the throughput of the machine as well as the lifecycle of its components.

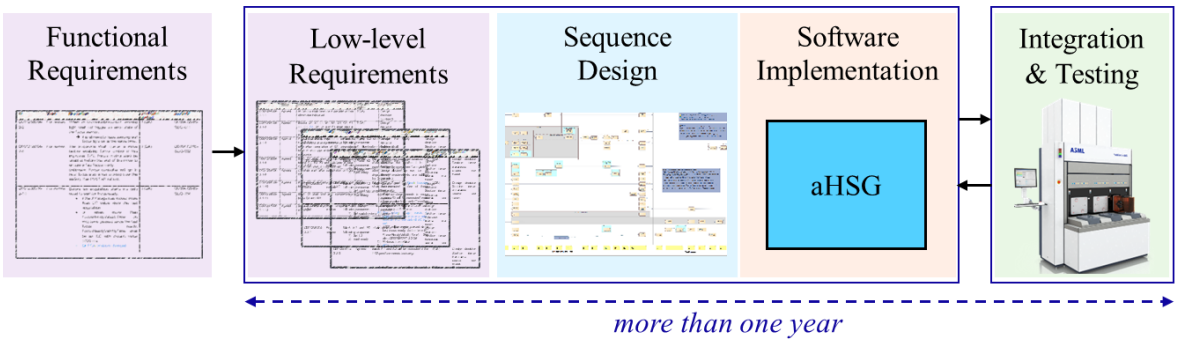

Figure 4: The process from the conception of an MA sequence to its implementation into an aHSG

1.5 Thesis outline

This thesis is organized in a V-shaped abstraction level structure, with the first and final parts being non-technical and generic, while the middle parts dive into technical analyses. The chapters are classified in four categories:

- **Introduction**
  Chapter 1 is a generic introduction to the company, its machines, and the project, while Chapter 2 elaborates on the problem domain and the objectives.
- **Transition**
  Chapter 3 depicts the stakeholders of this project (from ASML and TU/e): the supervisors, the clients, and external consultants. Chapter 4 translates their needs and wishes into requirements and usecases.
- **Design and Implementation**
  The requirements and usecases drove the architecturally significant decisions and the design of the system. The latter are analyzed in Chapter 5. Chapter 6 describes the implementation of the design into code, while Chapter 7 explores how the system was verified and validated.
- **Overview**
  Chapter 8 provides a summary of the outcome, addressing the benefits, limitations, and recommendations. Chapters 9 and 10 conclude this thesis by documenting the processes that were followed throughout the project and the lessons learned.

2 Problem Analysis

This chapter continues from the project context described in the previous chapter and elaborates on the problem itself and the objectives of the project. It also analyses the aHSG module, which is closely related to the project.

2.1 Problem Context

The following subsections dive deeper into the importance of the aHSG and the considerations of which this project addresses.

2.1.1 Creating an MA sequence and the aHSG

Designing a new MA sequence is a long, tedious, and trial-and-error based procedure. Two main groups of stakeholders are involved in this procedure (see Chapter 3): the Functional Designers (FDs; a.k.a. physicists) and the Software Developers (SDs). The FDs receive the high-level requirements for the sequence and design the sequence as a diagram[1], documenting any lower-level requirements that become apparent.

The SDs then use the diagram and requirements in order to implement a new iteration of the aHSG. As discussed in Section 1.3, the aHSG is a software module (different for each YieldStar variant) that is responsible for translating a high-level acquirement-related instruction into lists of low-level commands. These commands are later sent to the hardware peripherals and a synchronizer in order to perform an MA sequence.

After implementation, the aHSG is tested on the machine itself; the engineers analyze the behavior and performance of the machine and compare them to expected results. The conclusions drawn from the verification of the aHSG's functionality in the real system are used in a feedback loop to revisit its implementation, as well as the design of the MA sequence that it embodies.

2.1.2 Considerations and challenges

The engineers review how the machine operates when different instructions are given as stimuli, as several questions need to be answered:

- Did the machine act as intended?
- Were there any errors perceived by the sensors?
- Were the throughput and critical path as expected?
- How can any ordering issues and performance bottlenecks be fixed?

The answers to these questions fuel the feedback loop that FDs and SDs together follow, in view of finalizing the sequence and the aHSG before and during the machine's lifecycle.

2.2 Project objectives

The purpose of this project was to rework the above workflow: the design of the HW sequence and its implementation into an aHSG. This vision was drawn from the following matters:

- The design of an MA sequence needs much manual effort.
- The optimization of the sequence is purely based on ideas of the engineers; it is not a deterministic and exhaustive procedure.
- The translation of the sequence and the low-level requirements into an aHSG is error-prone.
- The sequence is static; it is not optimized for each possible sequence instance possible.

[1] In some cases, the sequence diagram was also maintained by the SDs.

- The whole process takes more than one year to complete[2].
- There is no way to visualize alternative KPIs and critical paths before testing on the machine itself.

From the above, we drew the main needs of our clients:

- Generate optimal sequence designs automatically
- Automatically translate the sequence into code or facilitate a similar capability
- Ensure that the sequence and code agree with the requirements
- Provide visualization and analysis methods
- Make the execution in line (as a secondary need)

The objectives are as shown in Table 1.

Table 1: Project objectives

ID	Description
OB-01	Generate the sequence model automatically
OB-02	Optimize the sequence for different goals, e.g., increase throughput, reduce heating, prolong component lifetime, reduce dynamic strain
OB-03	Assist or replace the manual implementation of the aHSG
OB-04	Visualize the generated sequence and the expected critical path
OB-05	Provide analyses on the estimated KPIs of the generated sequence, e.g., duration, duty cycle of the resources
OB-06	Make the process inline

2.3 The Acquirement Hardware Sequence Generator (aHSG)

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

[2] …as noted by one of the former FDs. Nevertheless, this process can also take less time if there are not a lot of changes between machine versions.

3 Stakeholder Analysis

To fully capture the conditions around which the project was carried out, it is important to list the main stakeholders who were explicitly or implicitly involved. Stakeholders include the ASML engineers who were either supervising the project or expecting to consume its results as users (relevant to the problem analyzed in Chapter 2), the TU/e supervisor and program director, as well as other experts from both organizations who provided insights or suggestions.

3.1 ASML

The ASML colleagues that were involved in the project can be divided into two groups: mentors and clients. The first category contains peers that guided the project, monitoring the progress and offering solutions. The second category contains people whose work the project aimed to simplify and improve.

3.1.1 Mentors

Zhifeng Sheng
Zhifeng was the initiator and company-appointed supervisor of the project. He was the main person responsible for monitoring my progress, reviewing most produced artifacts, and making sure that my efforts were aligned with the goals of the project and the company. He was also engaged in technical details, providing ideas and suggestions.

Tim Kouters
Tim was the project owner and the Group Leader (GL) of our group (FC-120: Software Layout). While his available time was limited, he did attend most recurring meetings of the project, and provided valuable propositions, as well as guidance over stakeholder management.

3.1.2 Clients and Experts

There were numerous clients and experts involved from ASML, such as physicists, software architects, software engineers, and throughput engineers. It is useful however to focus on two main categories: the Functional Designers (FDs) and the Software Developers (SDs).

Functional Designers (FDs)
The FDs were responsible for designing the MA sequence and formulating its low-level requirements. They were closely affiliated with the project, since most of its goals targeted them as the beneficiary. They attended many discussions and demonstrations, steering developments towards features that they will incorporate in their workflow for future machines. They were characterized by high enthusiasm about the progress, and they were very keen to contribute with requirements and feedback.

Software Developers (SDs)
The SDs were responsible for designing and implementing the aHSG, following the models and requirements agreed with the FDs. Their enthusiasm was initially limited, since they have already simplified their workflow for current machines, requiring relatively low effort. However, as the project matured, there were multiple benefits unraveled that provide significant improvements to the aHSG's development and deployment workflow. Also, with the introduction of more complex machines in the future, the automation of the SDs' work becomes even more valuable.

3.2 TU Eindhoven

3.2.1 Mentors

Stef van den Elzen
Stef was the second supervisor of the project. He was responsible for overseeing the project by focusing more on the academic interest. He was knowledgeable in visualization concepts, which was a helpful asset for the project, since it entailed many visualization elements. Thus, he provided a lot of guidance and ideas of what can be implemented in that regard, but also what other features could be supported.

Yanja Dajsuren
Yanja is the program director of the Software Technology EngD in TU/e. She was not involved much in the project, but she provided guidance and instructions on deliverables and deadlines related to the university.

Others
Since this was an EngD project aimed to combine industry and academia, we aspired to request consultation from TU/e experts, apart from the ones from ASML. We contacted professors and PhD students in order to extract valuable insights into matters such as constraint programming, job-shop scheduling, and visualization. They were able to provide me with knowledge, opinions, and research material on relevant subjects and helped us decide on the technologies to use.

4 Usecases and Requirements

This chapter analyzes the requirements that were negotiated with the stakeholders, along with the architectural decisions and usecases that they were materialized into.

4.1 Introduction

The process of gathering and managing requirements was a crucial step in the project. Through this stage, the project evolved from a two-page proposal into tangible objectives that it had to achieve and constraints that it should abide by. The requirements are drawn from the needs of the stakeholders and the characteristics of existing systems and infrastructure that the output of the project will interact with or replace. The wishes of the clients are translated into requirements and usecases that are then materialized into the system design described in Chapter 5.

4.2 Elicitation process

Following the Agile methodology and the "fail fast, decide later" approach, much of the project's direction was drawn by frequent prototyping and review by stakeholders. Four types of reviews were organized to that end:

- Weekly reviews with the ASML supervisor
- Bi-weekly reviews with the TU/e supervisor
- Monthly Project Steering Group (PSG) meetings with the supervisors and project owner
- Sprint demos that included relevant clients (i.e., "consumers" of the outputs of the project) and experts regarding the systems

Apart from prototypes and demonstrations, needs and requirements were also drawn from direct discussions with clients. A significant workload during the first months of the project was characterized by meetings with ASML colleagues. The purpose of these meetings was to establish the domain, help me understand how the existing systems worked, and exchange ideas on what would be beneficial and valuable for the engineers.

4.3 Customer wishes

As mentioned in Chapter 3, the clients are grouped into two main categories: the FDs and the SDs. Depending on their line of work, the groups share different wishes that are addressed in this project. The wishes are analyzed in the following subsections.

4.3.1 Wishes of the FDs

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

4.3.2 Wishes of the SDs

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

4.4 Requirements

Sections 4.4.1 and 0 summarize the functional and non-functional requirements that drove the architectural decisions and design of the project, while a more elaborate list is provided in Appendix A: Detailed list of requirements. Keep in mind that the requirements denote no explicit distinction between the two tools developed under this project (see Section 5.1), but they refer to them in unison as "the system." Instead, Chapter 7 links the requirements with the respective tool and verification or validation method.

4.4.1 Functional Requirements

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

4.4.2 Non Functional Requirements

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

4.5 Usecases

4.5.1 Introduction

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

4.5.2 Usecase Analysis

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5 Architecture

Following the requirements and usecases described in Chapter 4, this chapter discloses the architecturally significant decisions and designs that drove the implementation of the solution. Kruchten's 4+1 [3] model is used to provide a holistic view of the system and its subsystems, guiding the implementation later described in Chapter 6.

5.1 Overview

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.1.1 Sequence Design and Optimization Tool (SDOT)

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.1.2 Inline Sequence Interpretation Tool (ISIT)

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2 Architectural Decisions

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.1 Selection of high-level system decomposition strategy

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.2 Selection of the description format for SDOT's input

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.3 Selection of the description format for SDOT's output (i.e., ISIT's input)

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.4 Selection of schedule optimization technology

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.5 Selection of programming language of SDOT

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.6 Selection of visualization technology for the dependency graph

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.7 Selection of visualization technology for the outputs

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.8 Selection of the functional approach for ISIT

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.2.9 Selection of the composition module strategy

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.3 4+1 Architecture Model

Kruchten proposed the 4+1 view model of architecture which is a model for describing the architecture of software-intensive systems based on multiple views [3]. The 4+1 views are the following:

1. The design or **logical view** defines classes and interfaces to represent the problem.
2. The dynamic or **process view** describes the system's processes and threads.
3. The physical or **deployment view** describes the system's physical architecture.
4. The development or **implementation view** describes the organization of the software modules in the system.
5. The scenarios or **usecase view** describes the system from the viewpoint of its users. This part is documented in Section 0.

The first four views constitute a description of the architecture of the software system designed and the architecturally significant decisions made. The fifth view are the usecases or scenarios that serve as example uses of the system (see Section 0). Figure 5 shows a summary of these views.

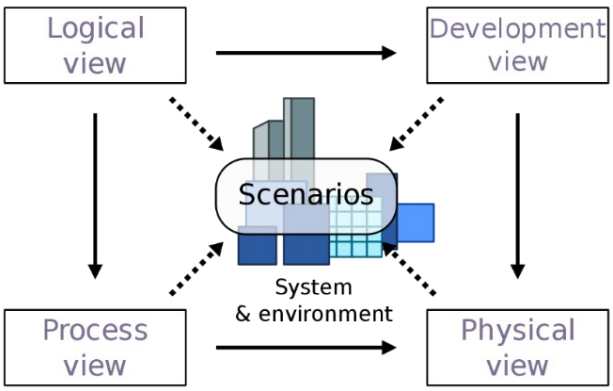

Figure 5: 4+1 view
Source: [4]

5.3.1 Logical view

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.3.2 Process view

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.3.3 Development view

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

5.3.4 Deployment view

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6 Implementation

This chapter analyzes how the project was implemented into SDOT and ISIT. It explains the workflow of how the tools are used and demonstrates snapshots of the individual steps and features.

6.1 SDOT

SDOT is a user-driven tool. It is started and configured directly by a user (typically an FD), who provides it with the necessary data and configures its execution. The use of SDOT can be divided in five steps, described from Subsections 6.1.1 through 0. More detailed information and instructions can be found in SDOT's User Manual [5].

6.1.1 Input modelling

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6.1.2 Configuration of execution

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6.1.3 Sequence generation and visualization

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6.1.4 Export of instructions

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6.1.5 Export of configuration file

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6.2 ISIT

Unlike SDOT, ISIT is not a user-driven tool. It's an internal software module that automatically responds to translation requests. Human intervention is only needed for it to be enabled and configured. Subsections 6.2.1 to 6.2.3 provide a summary of how ISIT is configured and used, while more specific information is provided in its instructions manual [6].

6.2.1 Integration within YieldStar's software

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6.2.2 Configuration using the input from SDOT

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

6.2.3 Translation of the segregated sequence into hardware commands

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

7 Verification and Validation

This chapter elaborates on the methods designed and followed to ensure that the tools developed abided by the functionalities and quality-related aspects as per the requirements and usecases analyzed in Chapter 4.

7.1 SDOT

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

7.1.1 Internal verifiers

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

7.1.2 Automated testing

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

7.1.3 Manual use

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

7.2 ISIT

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

7.2.1 Automated testing

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

7.2.2 Manual testing

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8 Conclusions and Future Work

This chapter serves as a wrap-up of the results achieved, insights gained, and recommendations compiled during the project. It provides a synopsis of what the company gained and how further benefits can be harnessed by utilizing the findings and developed tools.

8.1 Benefits

Chapter 7 provided tangible evidence on the effectiveness of the tools and the correctness of the output. It is useful, though, to provide an outlook on how the tools are going to be used by the relevant stakeholders and what added value they provide to the processes they were developed to optimize.

8.1.1 SDOT

SDOT provides the FDs with a way to automatically construct and optimize the machine behavior, reducing their efforts to develop the MA sequence. Designing the sequence is a considerable portion of the year-long process of the MA and aHSG related developments. Thus, with the introduction of SDOT, we expect a substantial decrease in the workflow's duration. Also, consistency is another key aspect, as SDOT further couples the documented requirements with the sequence produced. As presented in Appendix C: Consistency and optimization, SDOT was successful in identifying not only inconsistencies between the MA sequence and the aHSG implemented, but also optimization opportunities that were initially overlooked by the engineers that manually designed and revisited the sequence.

The FDs have already regarded SDOT as valuable to their present and future work, stating that it is also convenient to integrate into their workflow. In fact, the FD currently responsible for maintaining the MA sequence for YS-500 is using SDOT to establish a consistent model of the machine and its behavior, and possibly to find leftover points for improving the machine's behavior. After getting familiar with the tool for this usecase, he will be using it in the coming months to create the MA sequence of the upcoming YS-550 machine type (developments start in 2024; first machine to be introduced in 2025).

Indeed, we conclude that SDOT and the workflow it facilitates should be utilized to design and optimize the MA sequence of YS-550 and future machines. It will facilitate the sequence being created faster, with massively reduced human effort, and with fewer fixes and reviews needed. An SDOT-generated sequence will possibly result in increased throughput and improved utilization of resources. Since SDOT is usecase agnostic, it can also be used for other scenarios within ASML as well, including the WEX sequence (see the recommendation described in Subsection 8.3.1).

8.1.2 ISIT

Although the SDs were initially hesitant for ISIT to be introduced as a replacement for the original aHSGs, they later accepted and promoted its integration to production code. The definitive evidence extracted proved its value for automating and improving the workflow under consideration. Extensive testing demonstrated that ISIT is able to rapidly and successfully facilitate an MA sequence, without the need to rebuild the machine's software, as needed when the original aHSG was updated. This also considerably decreases the time and effort needed to experiment with different MA sequences, gather throughput statistics, and conclude on the best MA sequence in practice. In fact, the time to implement the MA sequence into code was reduced from approximately two days to less than a day[3] (more than 50% decrease), while deployment time was shortened from approximately two hours to thirty minutes (around 75% decrease)[4].

[3] In contrast to original aHSGs, for ISIT the developers do not need to implement the sequence itself, but only any new blocks or conditions.

[4] For ISIT, only a machine restart is needed (around thirty minutes). For traditional aHSGs the code needs to be rebuilt, repackaged, and reinstalled on the machine (around two hours). Additionally, with only minor adjustments, reconfiguring ISIT will not require a machine restart, reducing its reconfiguration time to only five minutes.

While it is not yet confirmed if ISIT will be used for YS-550 and future machines, the feedback received and the fact that its implementation for YS-500 has already been integrated provide a positive outlook. Even if it is not used as the main aHSG, it can still be used by throughput engineers to easily experiment with alternative sequences for certain scenarios, harnessing more machine productivity.

8.2 Limitations

Despite satisfying all core requirements with great success, the tools demonstrate some restrictions that need to be taken into account. To this end, this section analyses some limitations of the delivered solutions.

8.2.1 Static duration of actions

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8.2.2 Lack of support for floating-point-number durations

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8.2.3 Alternative actions for resource

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8.2.4 Machine-dependent action and condition translation

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8.2.5 Manual transcription of the requirements into a problem definition

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8.3 Recommendations for future work

This section lists suggestions on additional features and improvements that can be introduced in the tools produced. These items were either outside the context of the project (e.g., see Subsection 8.3.1), considered as low priority and ultimately not completed (e.g., see Subsection 8.3.2), or resolutions of the limitations documented (e.g., see Subsection 8.3.3).

8.3.1 Reuse for other scenarios

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8.3.2 Facilitate visual editing

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

8.3.3 Dynamic action durations

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

9 Project Management

This chapter constitutes a view of the activities of the project using a Work-Breakdown Structure and how they were planned throughout the ten months. The risks are analyzed as well, providing contingency and mitigation strategies.

9.1 Work-Breakdown Structure (WBS)

The project workload was divided into five categories of activities:

- **Planning and Management**: The actions including (but not limited to) the elicitation of remaining tasks, their assignment into the available work time (in view of accomplishing the goals and milestones of the project), and the management of the stakeholders and their expectations.
- **Research**: The activities related to the investigation of the problem domain and state-of-the-art, the experimentation with available tools and alternatives, and the exploration of relevant systems.
- **Design and Implementation**: The workload that was concerned with devising the architecture of the system, identifying alternatives and deciding on choices, and the implementation of the software tools SDOT and ISIT.
- **Verification and Validation**: The processes related to ensuring that the tools produced "facilitate the correct features" (validation) and that "these features work correctly" (verification).
- **Reporting**: The compilation of the thesis and presentations for ASML and the EngD defense committee.

The WBS diagram depicting the above is shown in Figure 6.

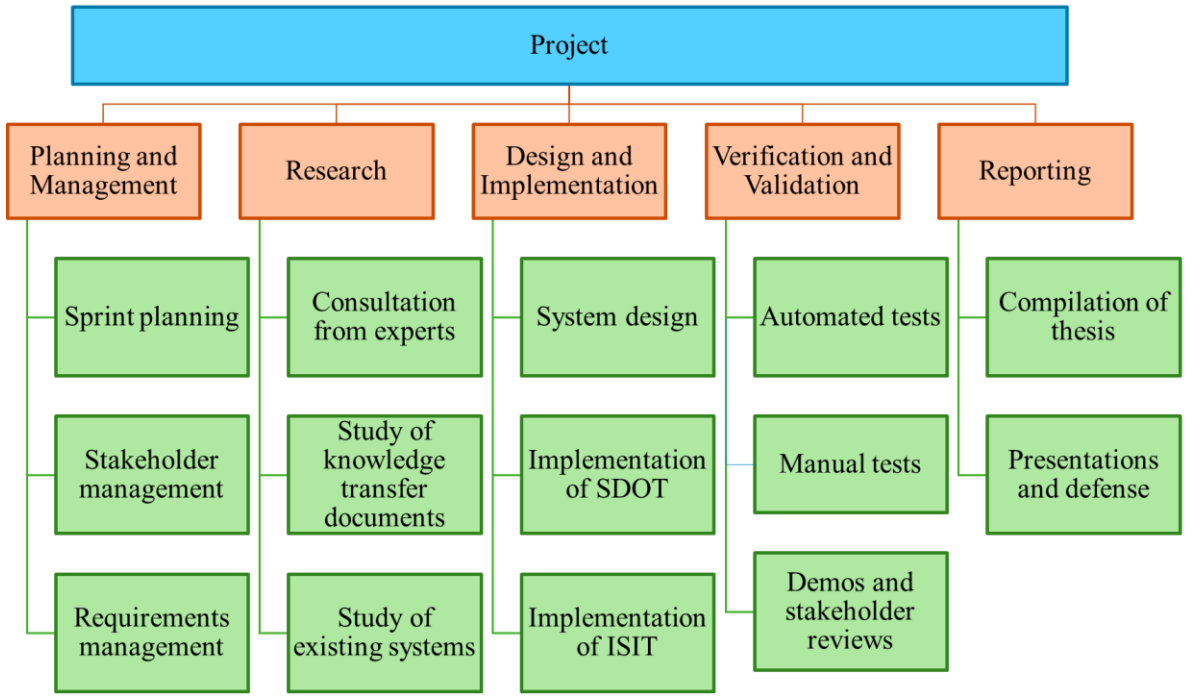

Figure 6: The WBS diagram

9.2 Planning

During the ten months of the project, it was crucial to plan, execute, and monitor the activities mentioned in Section 9.1. To this end, the project followed an iterative Agile-based methodology and the "fail fast, decide later" approach. Thus, much of the direction of the project was drawn by frequent prototyping and review by stakeholders. Nevertheless, this approach was guided by the rough project plan shown in Figure 7.

Initially, most of the effort was concentrated on creating strong foundations on the domain and the context, while identifying the stakeholders and their requirements. Prototypes were used to experiment with ideas and technologies, demonstrating how the project can progress. When the alternatives were well-understood and analyzed, we started formalizing the architectural designs and deciding on the technologies to use. Since the project entailed interesting innovative elements, we also filed an Invention Disclosure Form (IDF), in view of patenting our design and implementation. The transition between prototyping and formally developing the two tools was accompanied by broader demos to stakeholders and it was followed by the integration of ISIT to ASML's QBL and the usage of SDOT for YS-500 (and preparation for YS-550). This thesis was compiled as a "live" document throughout the project and as long as most results were established, broader presentations of the solution took place.

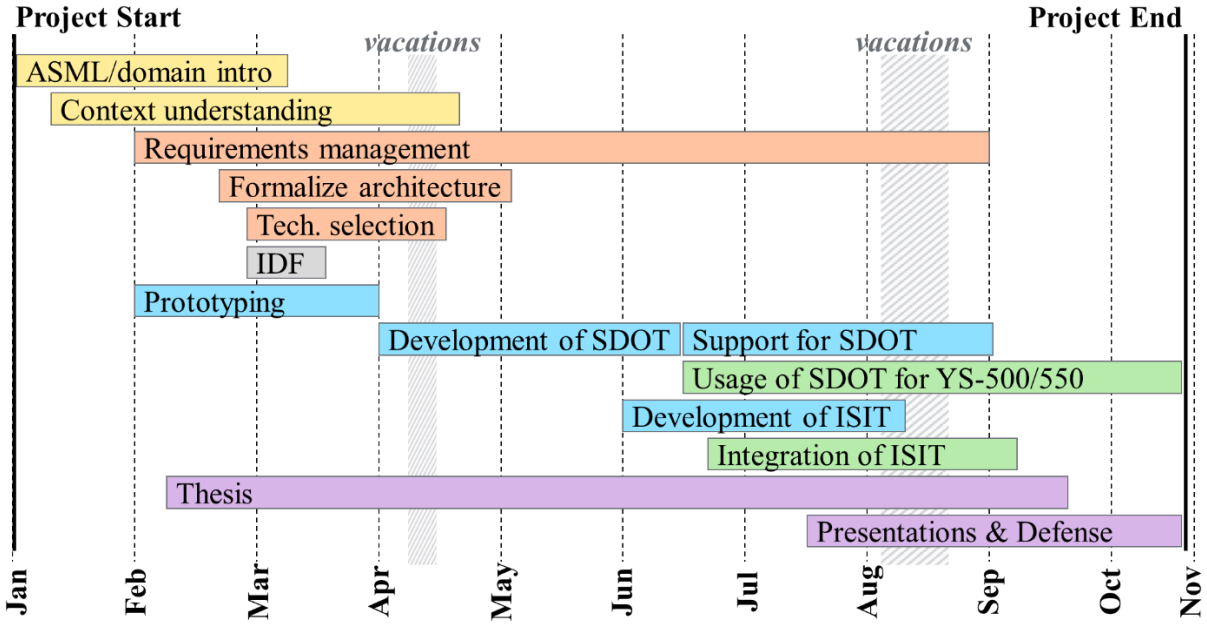

Figure 7: Coarse project plan

This plan was respected throughout the project and revised as soon as new information or feedback was available. The goals of the three-week sprints shown in Table 2 were determined by the rough project plan, and aimed towards allocating enough time for each stage in the project and implicitly time-framing the actions and preventing compounding of delays.

Table 2: The main goals per sprint

Sprint	Goals
Setup weeks	Familiarize with the company and domain, setup IT systems, do introductory trainings
Sprint 1	Gather information on the HSG, talk to stakeholders, draft ideas
Sprint 2	Prototype SDOT, formalize requirements, and negotiate mockups
Sprint 3	Prototype SDOT, solidify priorities of functionalities, and decide on technologies
Sprint 4	Design the system and start production-level implementation of SDOT
Sprint 5	Start implementing ISIT and consolidate first thesis draft
Sprint 6	Solidify ISIT's development and devise integration strategy
Sprint 7	Adapt SDOT, ISIT, and the thesis based on reviews and results
Sprint 8	Experimentally integrate ISIT to QBL, demonstrate and improve SDOT
Sprint 9	Extensively document and verify ISIT, integrate it formally to QBL, update MA sequence scenarios, and compile TU/e related reports
Sprint 10	Prepare thesis for second round of reviews, address updated scenarios, and present the progress

Sprint 11	Finalize SDOT and deliver thesis for second round of reviews
Sprint 12	Finalize ISIT, deliver thesis for third round of reviews, and present to company
Sprint 13	Finalize thesis, present to company and evaluation committee

In order to facilitate a frequent cycle of holistic reviews of the project status, Project Steering Group (PSG) meetings were organized. In the PSG meetings, we presented the current status, challenges, impediments, and next actions to the main stakeholders: the ASML supervisor, the ASML group leader, and the TU/e supervisor. Furthermore, the prototypes developed under this project were demonstrated to a larger audience almost monthly. This audience consisted of clients/consumers of the tools developed, domain experts from the company, as well as other engineers who were interested in the project. The attendants provided feedback on how the tools could be improved and how the direction of the project could be finetuned to benefit the company and the users most.

9.3 Risk analysis

In order to successfully handle unprecedented circumstances, we compiled the register shown in Table 3. It demonstrates the risks that were identified to potentially hinder the progress of the project, as well as their importance and the strategies devised to prevent (mitigation) or tackle them (contingency).

Table 3: The risk register

ID	Description	P:C:S[5]	Mitigation plan	Contingency plan
R_01	The trainee gets sick	2:3:6	Self-care	Replan actions
				Reduce the scope
R_02	Supervisor(s) gets sick	2:2:4	N/A	Ask for alternative person of reference
R_03	The selected scheduling method used does not produce optimal results	1:2:2	Read existing documentation	Switch to alternatives (numerous exist)
			Investigate discussions within the community	
			Prototype early	
R_04	Workload exceeds expectations	2:2:4	Manage expectations	Replan actions
			Plan and develop iteratively	Reduce the scope
R_05	The selected scheduling method is not time efficient	2:1:2	Read existing documentation	Switch to alternatives (numerous exist)
			Investigate discussions within the community	
			Prototype early	
R_06	Software developers are not satisfied with the value of the project	3:2:6	Carefully listen and document their needs and insights	Put more emphasis on SDOT than on ISIT
			Find ideas that provide considerable improvements over existing methods	Emphasize more on the coding instructions and not on ISIT
R_07	Testbench not available for demo	2:2:4	Investigate availability	Perform thorough testing via a Devbench and provide convincing proof that it
			Book a timeslot early	

[5] Probability:Consequence:Severity; where Probability x Consequence = Severity

				would work correctly on a Testbench as well
R_08	Machine not available for demo	3:1:3	Investigate availability	Perform thorough testing via a Testbench and provide convincing proof that it would work correctly on a real machine as well
			Book a timeslot early	
R_09	Absence of HSG module owner	2:1:2	Ask about their personal leave days	Approach replacement module owner
			Schedule meetings early	

10 Epilogue

This chapter provides a personal conclusion to the project, demonstrating some useful learnings and a retrospective view on the achievements and challenges.

10.1 Lessons Learned

The purpose of this project was not only for the company to benefit from the technical developments and findings, but also a personal means for me to gain more experiences and shape insights. Apart from the knowledge on technologies and tools that I absorbed and used, I constructed a number of generic methods to help me progress more steadily and efficiently. I aim to closely consult them in my future endeavors.

10.1.1 Rapid prototyping

Rapid prototyping can be an efficient method to estimate the capabilities and challenges of available technologies and ideas. It is probably the most effective way to receive feedback and suggestions from stakeholders as well, since they can see a tangible representation of the proposals and ideas, reducing confusion and miscommunication. The responses and propositions of the stakeholders are also easier to grasp and facilitate.

Throughout the project, the above mindset was followed quite closely; almost all weeks of the project resulted in a new or revised prototype of the system or its parts. Monthly demonstrations to the supervisors, clients, and software architects became a core ingredient of the project's direction and progression. The stakeholders were iteratively becoming more aware of the potential of the concept and were more convinced of the added value offered by the two tools.

10.1.2 Compare predicted and actual workload

Keeping track of predicted versus actual time spent in each sprint (for stories and tasks) can significantly help with planning and organizing activities. For instance, I observed that in this project, the actual time spent on tasks was on average 50% higher than the predicted time. While this seems like a considerable error, it was in fact very valuable. Why? Because it was consistent.

More specifically, it was demonstrated that for three-week sprints, an accumulation of two weeks of predicted workload at the start of the sprint was accompanied by an additional week's worth of unplanned activities. That was a fruitful discovery that enabled my sprint planning to be quite accurate, keeping productivity high and preventing many stories or tasks "leaking" to the next sprints. This fact remained quite consistent almost throughout the whole project and helped me keep a steady cadence.

10.1.3 High-level planning maintains focus

Uncertainty at the beginning of a project is always prominent. There are many paths and challenges that are vague or hidden altogether. It is vital, however, to devise a high-level plan that can dictate a general direction, time-framing the phases of the project. For instance, if a development takes too much time, due to, for instance, increasing demands from clients, it is crucial to acknowledge this fact and confine expectations. Otherwise, delays in the current phase will start "leaking" into follow-up phases and lead to the postponement of milestones.

While I was initially not very focused on planning further into the future, my supervisors wisely indicated that determining a vision and milestones for the months to come would contribute to the smoother administration of the project's activities. Indeed, by following that approach, I was better prepared to gather the resources I needed and schedule the required meetings for the activities that would follow. This led to better parallelization of tasks and fewer occurrences of impediments and stalling.

10.2 Project Retrospective

From my viewpoint, the project can definitely be described as a rollercoaster. There was a plethora of emotions, sometimes manifested in parallel, driven by the diverse stakeholders that I had to satisfy and the difference in nature of the two tools I developed.

The first months felt like a deep dive into the unknown, as I was starting to explore the vast domain space of ASML and talk with more than a dozen engineers and experts from the company and the university. My private life was also bolstering the same feeling, as I was just entering a new phase in my personal relationships and life balance, with extensive changes. Juggling through my new responsibilities, the status quo, and arising challenges, I certainly did not find it easy to put things in order, but I certainly did manage to do so. Regarding the project, I strategically organized many meetings with possible stakeholders so that we get acquainted and I became familiar with the domain and the technical challenges. In the meantime, I had already started investigating and implementing ideas on a prototype that later came to be SDOT.

In the second phase of the project, I learnt how to work with the stakeholders to establish a baseline of the vision and I focused on iteratively evolving and presenting the system under development. While the developments on SDOT were progressing quite smoothly, I had to manage conflicting opinions on the future vision. The proposals that later formed ISIT were initially not well-received by the software engineers responsible for aHSG and the modules around it. They were concerned that none of the alternatives would demonstrate enough added value to justify the effort to develop and integrate such a tool. In general, this phase was both satisfying and troubling; the project was becoming more tangible and interesting, but also required providing strong proof of added value.

As summer came closer, the project reached its peak intensity. SDOT received wide approval and there were already declarations that it would be used by the FDs for the current and future YieldStar machines. Implementing and demonstrating ISIT also made it appealing towards the relevant stakeholders, who became much less skeptical and much more enthusiastic. At that point, I had already broadened my technical skills and I was more able and specific to promote the ideas and address remaining concerns.

The period that followed was a defining factor in my development. Having to integrate the tools, present the findings, and demonstrate the solutions fueled a lot of feedback by highly technical experts, who helped me identify points of improvement, implicitly teaching me better practices of software engineering and critical thinking. Unfortunately, I was not able to demonstrate ISIT on a full YS-500 machine, because of matters unrelated to my project (e.g., limited availability of the demo machine, urgent requests from ASML's customers). In spite of that, I succeeded in linking the tools and showcasing the significant advantages of the new workflow. Hopefully, the thesis is a representative conclusion to the achievements of the project!

Glossary

Term	Description
a.k.a.	also known as
acquirement	the process of acquiring optical images of wafers
BDD	Behavior Driver Development
CLI	Command Line Interface
CP	Constraint Programming
DSL	Domain Specific Language
EngD	Engineering Doctorate
FD	Functional Designer
ISIT	Inline Sequence Interpretation Tool
KPI	Key Performance Indicator (metric)
MA	Move Acquire (sequence)
PDDL	Planning Domain Definition Language
QBL	Qualified BaseLine
scanner(s)	lithography machine(s)
SD	Software Developer
SDOT	Sequence Design and Optimization Tool
ST	Software Technology
SW	software
TU/e	Eindhoven University of Technology
UID	Unique Identifier
WBS	Work-Breakdown Structure
WEX	Wafer Exchange (sequence)
YS	YieldStar (machine)

References

This section was edited for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

[1] ASML, "YieldStar 380G," [Online]. Available: https://www.asml.com/en/products/metrology-and-inspection-systems/yieldstar-380g.

[2] ASML, "YS-T375F," [Online]. Available: https://my.asml.com/products/applications/PublishingImages/Pages/YieldStar/YS/YS-T375F.png.

[3] P. Kruchten, "Architectural Blueprints—The “4+1” View," *IEEE software,* pp. 42-50, November 1995.

[4] Wikipedia, "Architecture view model," [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e6/4%2B1_Architectural_View_Model.svg/531px-4%2B1_Architectural_View_Model.svg.png. [Accessed May 2023].

[5] G. Evangelou, "SDOT's User Manual," September 2023. [Online]. Available: https://apps-bbdc-prd.asml.com/projects/YSSEN/repos/sdot/browse/User Manual.docx.

[6] G. Evangelou, "How to use and update the Auto-HSG (aka ISIT)," September 2023. [Online]. Available: https://wiki.asml.com/wiki/confluence/pages/viewpage.action?pageId=327593210.

Appendix A: Detailed list of requirements

This section was edited for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

Section 4.4 provided a summarized view of the requirements of the project. For completeness, this appendix provides the full register.

Appendix B: The MA sequence of YS-500

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

Appendix C: Consistency and optimization

SDOT and ISIT helped reveal inconsistencies between the requirements, diagram, and implementation of the MA sequence, as well as demonstrate further opportunities for throughput improvement. This appendix provides examples for both scenarios.

C.1 Identification of model inconsistencies

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

C.2 Identification of optimization opportunities

This section was removed for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

Appendix D: Higher-resolution snapshots

This section was edited for confidentiality purposes. To view the original information of this section, please request the confidential version of the report from ASML.

This Appendix provides higher-resolution images for the sequence-related diagrams of Chapter 6.

About the author

Georgios Evangelou received his BSc and MSc degree in Electrical and Computer Engineering from the University of Patras in 2020, where he also served as a teaching assistant at the Digital Signal and Image Processing lab. He also participated in various projects, such as the 2018 APS/URSI design contest (Boston, USA), where his team achieved 4th position globally. From 2019, he became an R&D Engineer at LMS, where he worked on EU-funded industry projects, developing Human-Robot Collaboration systems. He was also a work-package leader and hosted discussions between R&D and industry partners. Afterwards, in his EngD at the Technical University of Eindhoven, he worked on projects for Philips Research, Airbus, and CERN for various technical and non-technical roles. He performed his graduation project for ASML, with this document being the project's thesis. His current interests include software design and implementation, parallel computing, and group leadership.

PO Box 513
5600 MB Eindhoven
The Netherlands
tue.nl

EngD SOFTWARE TECHNOLOGY

EINDHOVEN UNIVERSITY OF TECHNOLOGY