Design complexity reduction using object-oriented pagadigm

Technische Universiteit
Eindhoven
University of Technology

/ Department of
Mathematics and
Computer Science
/ PDEng Software
Technology

Design Complexity Reduction using Object-Oriented Paradigm

Exemplified by Redesigning the Scanning Application of ASML's Wafer Positioning Module

Wahyu Utomo

Design Complexity Reduction using Object-Oriented Paradigm

Exemplified by Redesigning the Scanning Application of ASML's Wafer Positioning Module

Eindhoven University of Technology
Stan Ackermans Institute / Software Technology

Partners

ASML Netherlands B.V.

Eindhoven University of Technology

Steering Group

Ir. H. T. G. Weffers PDEng
Ir. M. Hendriks
Ir. B. Ok
Ir. C. F. van Antwerpen

Date **October 2019**

Document Status Public

SAI report no. **2019/096**

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

Contact Address
Eindhoven University of Technology
Department of Mathematics and Computer Science
MF 5.080A, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31402474334

Published by
Eindhoven University of Technology

Printed by
Eindhoven University of Technology
UniversiteitsDrukkerij

SAI report no.
2019/096

Abstract
The Scanning Application is one of the software components inside ASML's Wafer Positioning module that has grown complex over the last few years. As the complexity grows, maintainability decreases. This project was organized to reduce the complexity of the software component by proposing a new Object-Oriented design.

Keywords
Object-Oriented, Software Redesign, ASML, Wafer Positioning, Complexity Reduction, PDEng Software Technology

Preferred reference
Design Complexity Reduction using Object-Oriented Paradigm: Exemplified by Redesigning the Scanning Application of ASML's Wafer Positioning Module. Eindhoven University of Technology, SAI Technical Report, October 2019. (2019/096)

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

This PDEng thesis is approved by the supervisors and the Thesis Evaluation Committee is composed of the following members:

Chair:	Ir. H. T. G. Weffers PDEng
Supervisors:	Ir. H. T. G. Weffers PDEng Ir. M. Hendriks
Members:	Ir. B. Ok Ir. C. F. van Antwerpen Dr. Ir. T. Verhoeff

Foreword

The wafer stage is one of the most characteristic and critical parts of an ASML scanner. The correct and timely displacements of the chucks carrying the wafers is essential to the correct operation of the machine as whole. During the past 15 years ASML has developed proprietary software to control these chuck movements in harmony with other subsystems, improving, extending and optimizing the behavior with every new generation of scanners. We have now reached a point where further development of the software without a major re-design is undesirable.

In this project Wahyu was tasked with the challenge of re-designing this so-called wafer positioning and scanning application in accordance with modern design principles. As a result, he has created a complete overview of all the responsibilities of the existing component. This analysis has led to a completely new out-of-the-box design proposal, following object-oriented principles. It not only meets all the functional requirements, but more importantly it also means a major step forward in meeting non-functional requirements like maintainability and testability. Wahyu has even showed with a proof of concept implementation, tested on actual TwinScan hardware, that the design also meets critical performance requirements.

Thanks to the fresh look, dedication and hard work from Wahyu we now have a truly new design for this core component which meets all our requirements. Without his efforts we certainly would not have reached this point. The outcome of his project allows us to develop future generations of wafer stages for our scanners meeting the high-quality standards our customers expect from us. It even has the potential to become the reference design for other scanning subsystems which have the same needs.

Maikel Hendriks, MSc.
Software Architect, ASML
Veldhoven, 26 September 2019

Preface

This design report documents the results of the graduation project for the Software Technology (also known as Ontwerpersopleiding Technische Informatica (OOTI), Dutch) post-master program at Eindhoven University of Technology.

The project titled "Design Complexity Reduction using Object-Oriented Paradigm: Exemplified by Redesigning the Scanning Application of ASML's Wafer Positioning Module" was executed under the supervision of ASML Netherlands B.V. in Veldhoven. The project was implemented in ten months and was conducted in the Embedded Software Department within the NXT Wafer Positioning team. The goal of the project was to reduce the complexity of the Scanning Application by redesigning the application using Object-Oriented principle to achieve locality of change, unit testability, and better analyzability while maintaining the functionalities and performance.

The report is intended for readers with technical background in software engineering. Readers who seek inspiration on Object-Oriented analysis and design should refer to Chapter 2,3, 5, and 6 to understand the context, problem description, system requirements, and analysis & design. Readers with non-technical background, such as managers, could obtain a general information about the project in Chapter 1.

This report is a public version. A separate private version is available at ASML.

October 2019

Wahyu Utomo

Acknowledgements

I would like to express my very great appreciation to my company supervisors, Maikel Hendriks, Cornee van Antwerpen, and Beril Ok for their continuous support, guidance and feedback throughout the project. Their experience helped me got through technical difficulties in ASML and their encouragement kept me highly motivated to finish the project. I would also like to thank the NXT Wafer Positioning team for helping me during the project; and especially Leon Erps, the team lead, who made me felt welcomed in the team. Thanks for Berihun Yimam, Umut Uyumaz, Suresh Tatavarthy, and Omar Elshal for their technical input and feedback.

I would like to extend my thanks to my university supervisor, Harold Weffers, for his constructive feedback and encouragement throughout the project. His guidance in managing the project will be a valuable knowledge for me to face new projects and challenges in the future. I would also like to thank the program director of PDEng Software Technology, Yanja Dajsuren, and the program secretary, Desiree van Oorschot for their support and management of the entire curriculum of the Software Technology program. Also, I would like to thank all the coaches for their instructive lectures, which helped me improve myself during last two years of studies.

I would like to thank all my fellow PDEng Software Technology 2017 trainees for their feedback and support throughout the program. It was a pleasure to work with these wonderful people from around the world.

Finally, I would like to offer my special thanks to my family, especially to Arinda P. Rachman, my parents, my brothers, and my friends for their love, support, and encouragement.

October 2019

Wahyu Utomo

Executive Summary

ASML is the world's leading provider of complex lithography systems for the semiconductor industry. Inside the ASML machines, there are two Wafer Positioning modules, which transports silicon wafers to predefined locations (e.g., measure and expose position). This module is controlled by a set of software components with the Scanning Application being one of them. Over the years, the Scanning Application has grown significantly bigger as a result of incorporating new requirements. As the software component gets bigger, it accumulates complexity making the component hard to analyze, to test, and to modify. As a consequence, maintaining the Scanning Application becomes expensive.

The project aimed to reduce the complexity of the Scanning Application by redesigning the application using object-oriented principle. The main goals are to achieve locality of change, enhance analyzability, and improve testability. Besides, the new design should have the same functionalities and performance quality.

A new design of the Scanning Application is proposed. This design is a modular design, which distributes the responsibilities of the Scanning Application into smaller classes. This arrangement helps developers understand the component better. The new design prevents modification of a class to ripple to other classes as the coupling between classes is kept low. Also, dividing the component into small chunks makes the unit test applicable to this component.

For a proof of concept, a prototype implementation of this design in the C++ language was provided. This implementation was tested on the testing environment that involves real hardware (Testbench). The result shows that the new design fulfils the functional requirements of the Scanning Application. Furthermore, performance measurements of this implementation show that the new design has an acceptable performance difference. In some settings, the new design even improves the performance of the Scanning Application.

In conclusion, the design and implementation show that the proposed design of the Scanning Application manages to reduce the complexity of the component while maintaining the functionalities and performance. As a result, the cost of maintaining this component becomes cheaper. It is recommended for ASML to investigate the possibilities of applying the same approach to the Scanning Application of other subsystems other than the Wafer Positioning module.

Table of Contents

List of Figures

List of Tables

1. Introduction

In the last two decades, technologies have developed rapidly thanks to the emergence of small, cheap, yet powerful integrated circuits (ICs or chips). ICs are inherently three-dimensional structures of semiconductor materials, designed such way that the structure forms millions of transistors. These transistors work together forming logics and memories that are used in most computers. The structure is built by stacking layers of IC design on top of silicon wafers. To put IC design patterns into silicon wafers, photolithography systems are widely used.

A photolithography system is an integral part of IC manufacturing process. In principle, this system main purpose is to expose silicon wafers to light, which is masked with design patterns. This concept sounds simple; however, in the semiconductor domain, the system becomes extremely complex as the pattern needs to be engraved in nanometer scale. To achieve nanometer scale, the system needs to produce light with a very short wavelength and control it. The system also needs to have advanced lenses system that could retain the resolution of the projected pattern. The system needs to have wafer positioning systems that could accurately position wafer with nanometer precision. Furthermore, before being exposed, the wafers need to be measured to get physical profiles so that the layers can be perfectly aligned.

ASML is a global supplier in the chip manufacture industry in which the company produces photolithography system called TWINSCAN. Inside the TWINSCAN machines, the wafer positioning modules (or usually called chucks) are responsible for safely transporting silicon wafers throughout the lifecycle of the wafers inside the machines. This module is essential since many subsystems need to position wafers at a certain location to meet their business requirements. For instance, the wafer handler subsystem needs this module to bring wafers into load or unload positions. To fulfill its responsibilities, the positioning module is equipped with advanced hardware controlled via software stack on top of it.

One of the core software components that controls the wafer positioning module is the Wafer Positioning and Scanning Application. This software component provides abstraction for the clients of the wafer positioning module to use the positioning functionality. This software component is responsible for handling absolute positioning, synchronized positioning, and state information requests.

Over the last decades, the Wafer Positioning and Scanning Application has grown considerably big, especially the Scanning Application (i.e., this class has 77 methods and ten thousand lines of code) as new requirements keep coming. This growth is inevitable as ASML continuously improves the TWINSCAN system by shrinking the line widths and increasing throughput. As a result, new types of movements, settings, and constraints are introduced. With respect to this improvement, the Wafer Positioning and Scanning Application needs to accommodate those new requirements into its component.

As the changes are made, the Wafer Positioning and Scanning Application accumulates complexity. This component was designed using object-oriented (OO)-like concepts and implemented using the C programming language. The design decomposes the component into several inner-class-like based on their functionality. However, the responsibility distribution is uneven (one class has many responsibilities meanwhile another class only has one responsibility) and there is no further division for a class that holds too many responsibilities. As new requirements keep coming, the class becomes hard to understand and inflexible to change. Furthermore, the modification of this component is not well documented. As a consequence, the time needed for development and the chance of introducing bugs during development increase.

This project offers a solution to redesign the Wafer Positioning and Scanning Application using object-oriented modelling techniques to reduce the complexity of the software component and improve the component's quality towards modification. With quality we mean how easy it is to understand the component and how many classes will be affected and need to be tested when a new requirement comes. Whereas improving the software quality is the main goal, we need to keep in mind that the functionality and performance of the new design should be similar to and preferably better than the currently used design.

In this project, we limited the scope to the Wafer Positioning and Scanning Application and the software elements inside it. This limitation means that we are not going to redesign the component's clients and the peripherals (software components used by the Wafer Positioning and Scanning Application). This limitation also means that we must keep the interfaces between the Wafer Positioning and Scanning Application the same as before so that those components do not need to change anything during integration.

To verify our solution, we organized design review sessions with the experts of the company regularly. To validate our design, we made an implementation prototype of the Wafer Positioning and Scanning Application and tested it on a testing environment that involves real hardware. Using this prototype, we show how the new design incorporates new requirements into the design better than the current design. We also show that the design has the same functionalities with a slight performance difference (i.e., suffers penalty in one place and gains in another place). This difference is expected as we introduced new objects in between calls and implemented the prototype in a specific variant of the C++ programming language.

Outline

The rest of the report is structured as follows. Chapter 2 introduces the domain in which the project was conducted to help the reader understand the context of the project. The introduction includes a brief explanation about the concept of photolithography, the ASML photolithography machine, the Wafer Positioning subsystem inside ASML machine, and the Wafer Positioning and Scanning Application, which is the software component of interest. Chapter 3 focuses on analyzing the current state of the Scanning Application to find improvement points as bases to redesign the component. Chapter 4 presents the main project stakeholders that have been identified. Later, their concerns are gathered, and communication channels are defined.

Chapter 5 discusses the system requirements of the project. These requirements were formulated after investigating the source code, reading the documentation, and discussing with the experts. Chapter 6 discusses the new design of the Scanning Application. This discussion covers the design methodology used in this project, the plan to integrate the new design into the existing system, the design strategy, the object-oriented analysis and design of the Scanning Application, some design alternatives and the non-functional requirements. Chapter 7 presents the verification and validation process used in order to confirm that the new design fulfills the underlying requirements. Chapter 8 and Chapter 9 presents the conclusion and possible future works respectively. Chapter 10 discusses how this project was managed. The discussion includes the project plan, process and control, and information management. Finally, Chapter 11 discusses the experience that the author gained during the project.

2. Domain Context

This chapter introduces the domain in which the project was conducted to help the reader understand the context of the project. The introduction includes a brief explanation about the concept of photolithography, the ASML photolithography machine, the Wafer Positioning subsystem inside ASML machine, and the Wafer Positioning and Scanning Application, which is the software component of interest.

2.1 Photolithography in IC manufacturing industry

Photolithography, as the name suggests, is a specific lithography process in which light is used to imprint patterns. In the typical lithography machine, a pattern of a design or picture is carved in a flat stone or metal plate. Afterwards, chemical substance is applied to the plate and then this plate is used to imprint the pattern on a piece of paper or other media. The process of photolithography, however, is slightly different than lithography process. In photolithography, the pattern is created on a photomask that allows light to pass through and imprint a pattern on a photoresist substrate.

Photolithography concept is an integral part of the IC manufacturing process. Figure 2-1 shows a simplified process description on how ICs are manufactured. At the beginning prepared silicon wafers are coated with photoresist materials. Afterwards, the photolithography machine is used to imprint a layer of IC design pattern on the silicon wafers. Next, the silicon wafers are baked, and the area unprotected by the photoresist is etched. This left the pattern on top of the silicon wafers. This process is repeated until a 3D structure of silicon is created.

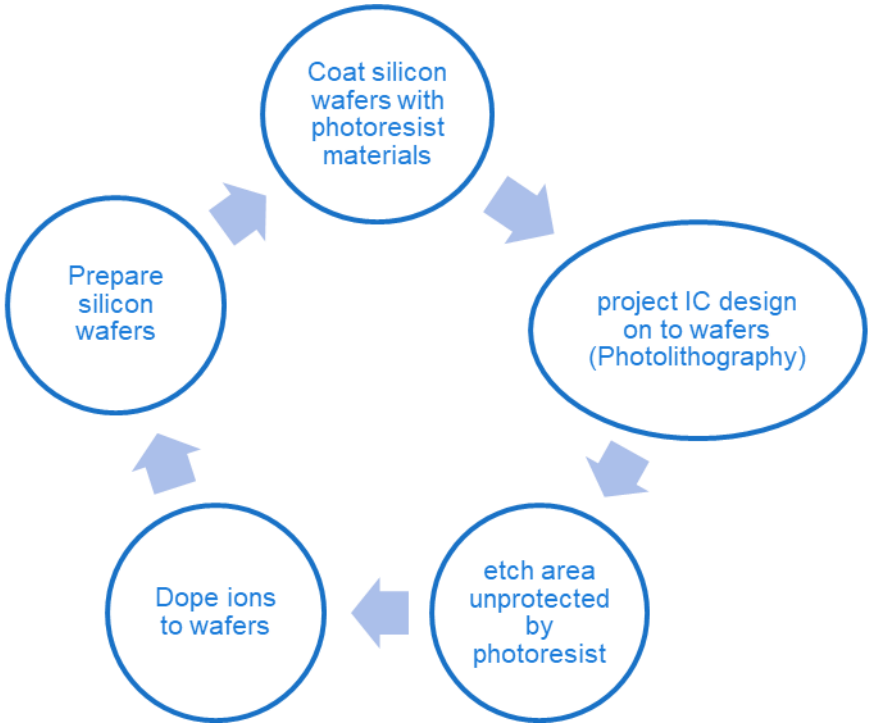

Figure 2-1 Simplified IC manufacturing process[1]

[1] The simplified process was inspired from an article on http://www.lithoguru.com/scientist/lithobasics.html

Photolithography holds the key to produce smaller IC desired by the industry. By reducing the wavelength while keeping the resolution of the projected image, smaller IC design can be imprinted on the silicon wafers. As a result, more transistors can be produced inside an area of silicon wafers, which means faster and cheaper ICs can be produced.

2.2 ASML Photolithography machine

ASML is one of the companies that produces photolithography system for IC manufacturing process. The company was founded in 1984 and was part of electronics giant Philips and chip-manufacturer Advance Semiconductor Material International (ASMI) [1]. Currently, ASML is a global leader in the semiconductor patterning products and services. The latest invention of this company is the TWINSCAN NXE photolithography systems, which use Extreme Ultra Violet (EUV) light with a resolution of 13.5 nm.

ASML's photolithography systems are responsible for imprinting the IC design patterns on top of the coated silicon wafers in the IC manufacturing process. The quality of the printing is determined by the quality of the image and the overlay accuracy between layers. To explain about such complex system better, the ASML's photolithography can be divided into four main parts: Illumination, Projection, Reticle Handler and Stage, and Wafer Handler and Stage.

Figure 2-2 shows a TWINSCAN NXT photolithography system divided into four parts. The Illumination and Projection systems are working together during expose. The Illumination generates the light needed for the photolithography process. For instance, the NXT machine generates Deep Ultra Violet (DUV) light. The Projection system aligns the lenses and mirrors to project the IC design on to the wafer. The Levelling and Alignment system is responsible for measuring the physical profile of the silicon wafers. The Reticle Handler and Stage prepare the photomasks with IC design patterns needed to filter the light. Finally, the Wafer Handler and Stage are responsible for loading/unloading wafers to/from the chucks and moving the wafers to a certain position (e.g., load and unload position).

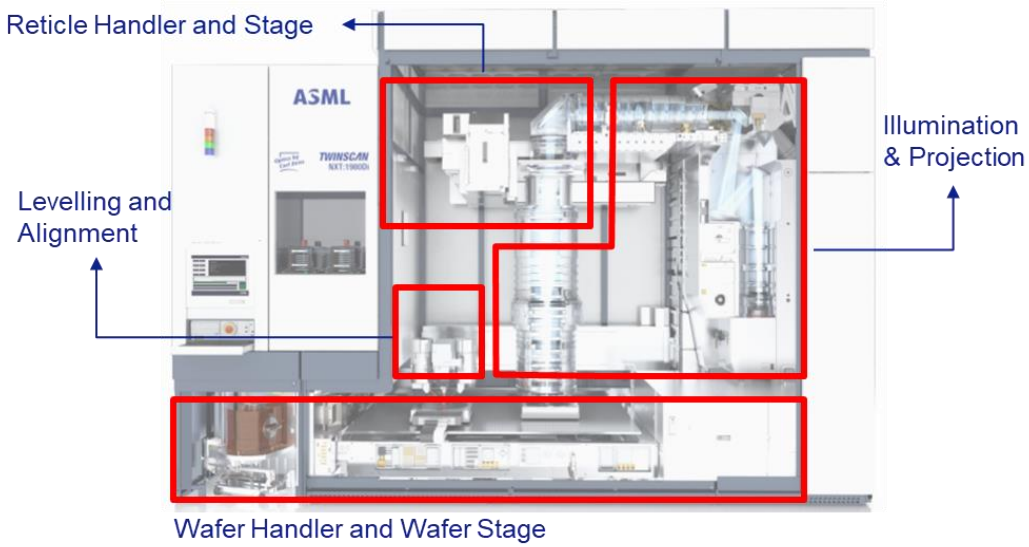

Figure 2-2 ASML TWINSCAN NXT System[2]

[2] Source: ASML Netherlands B. V.

The photolithography process inside this machine works as follows. At the beginning, the Wafer Handler loads a preconditioned wafer into the Wafer Stage. This preconditioning is essential to align the position of the wafer and to control its temperature. Inside the Wafer Stage, the wafer is placed on a Wafer Positioning module or commonly called a Chuck. The two terms are used interchangeably inside this report. Afterwards, the Chuck brings the wafer to the measurement position in which the wafer is measured. After the measurement is finished and the system obtains the physical measurement of the wafer, the Chuck moves the wafer into the expose position. In this expose position, the photolithography process happens.

During the exposure, the Reticle Handler and Stage prepare the photomask containing the IC design pattern. At the same time, the Illumination system generates the light and the Projection system passes the light through the photo masks and focuses the image to the specified resolution. Meanwhile, the Chuck moves the wafer to a certain position to expose a part of the wafer to the image. Finally, after this process ends, the Chuck brings the wafer near the Wafer Handler to unload it.

2.3 Wafer Stage and Wafer Positioning Module

The Wafer Stage subsystem, as can be seen in Figure 2-3, has two Wafer Positioning modules (or Chucks) inside it. This configuration enables the system to perform wafer measurement and exposure at the same time, hence boosting the throughput. In accordance to that, the Wafer Stage is divided into two sides, the measure and expose sides. The two Chucks should always be on the different sides, except when they swap position.

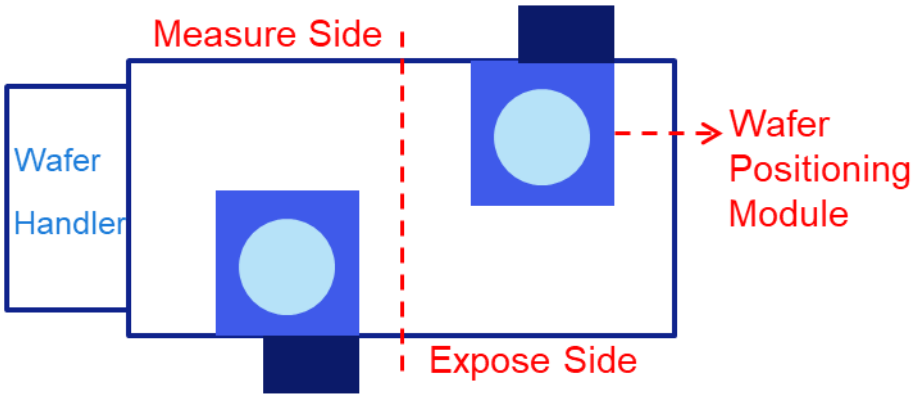

Figure 2-3 Wafer Stage subsystem

The Wafer Positioning module is mainly responsible for accurately and safely positioning the wafer in the Wafer Stage in six degrees of freedom. Figure 2-4 shows a domain model of the Wafer Positioning module depicted using UML class diagram to shows hierarchical composition. This diagram was extracted based on the description of the subsystem [2]. To transport the wafer accurately, this module is equipped with two kinds of actuators: long stroke and short stroke. The long stroke actuator handles extended movement and the short stroke actuator handles high accuracy movement. These two actuators work together in which one of them being the main actuator and the other follows the main one. Each actuator has a set of control modes to select depending on the type of movement (e.g., extended or high accuracy). The Wafer Positioning module navigates itself in the Wafer Stage by using two types of coordinate systems, namely the Baseframe and Metroframe coordinate system. The former is used for extended movement, and the latter is used for the high accuracy movement.

The Wafer Positioning module is also responsible for transporting the wafer safely, which means not to break the wafer during the movement. To gain a high throughput, the TWINSCAN machine would ideally set the Wafer Positioning module to move the wafer at the maximum speed and acceleration. However, that is not always possible due to certain conditions. For instance, sudden acceleration and deceleration could damage the wafer. Hence the Wafer Positioning module must set the performance properties (e.g., velocity and acceleration) conforming to specification. Also, to further enhance the safety of both the Wafer Position module and the wafer, the concept of constraints for the movement was introduced. For instance, the position constraints are defined to prevent the Chucks to move outside the Wafer Stage.

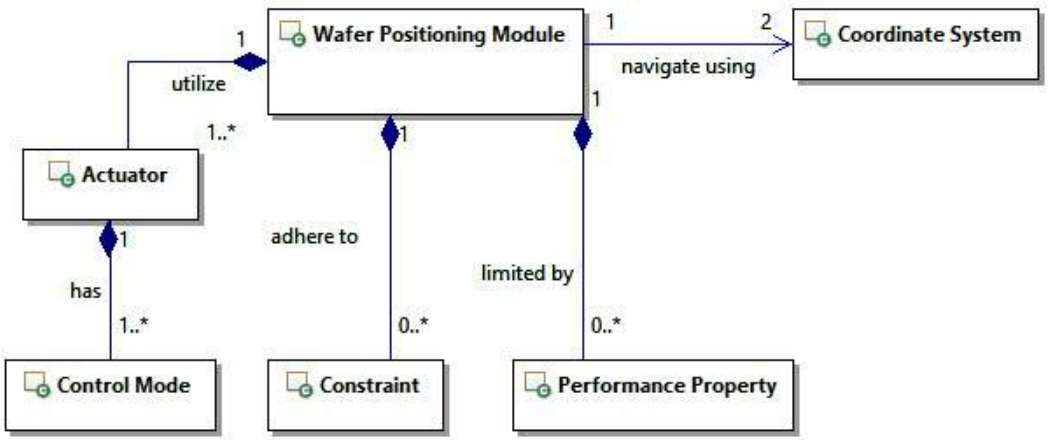

Figure 2-4 Domain model of the Wafer Positioning module

The Wafer Positioning module is controlled using a software driver. Figure 2-5 shows a simplified ASML software driver's stack consists of an Application layer and a Peripheral layer. The application layer's task is to provide high level functionality to the client of the driver. On the other hand, the Peripheral layer's job is to communicate with the actuator controller.

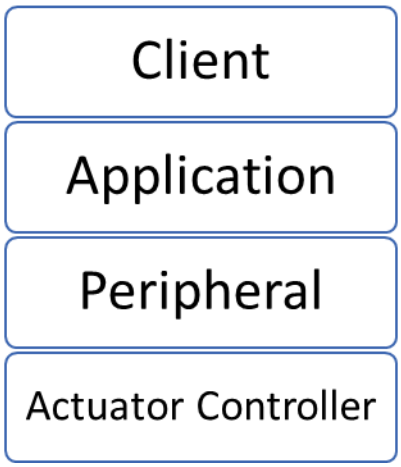

Figure 2-5 ASML's driver software stack (simplified)[3]

[3] Source: ASML Netherlands B. V.

2.4 Wafer Positioning and Scanning Application

The Wafer Positioning and Scanning Application is one of the software components within the Application layer of the Wafer Positioning module software stack. This component provides abstraction for other modules or subsystems to utilize the positioning related functionality of the Chucks. Throughout this report, the terms software component and application are used interchangeably to refer the Wafer Positioning and Scanning Application.

To provide abstraction for the client means hiding the process of moving the Chuck to a position from the client. This way, the clients do not need to know which actuator to trigger, how much performance properties are limited, and what constraints to follow. The clients focus on determining the destination location and the desired performance. The application, on the other hand, needs to process that information together with the information sent by the client.

In general, the Wafer Positioning and Scanning Application is responsible for abstracting the following functionalities:

1. Absolute Positioning: to position a chuck, whether it carries a wafer or not, to a specific location on the wafer stage,
2. Synchronized Positioning: to execute a series of actions that are queued beforehand when triggered by the Synchronization component,
3. Position Information Provider: to provide position related information of a chuck, such as its actual position and its state, for the clients.

2.4.1. Absolute Positioning

Absolute positioning means putting a chuck at a certain position on the wafer stage. This move is a blocking move, which means the application cannot serve another request from the client until the currently served request is finished. The destination to where the Chuck moves should be provided by the clients. Furthermore, in case of wafer displacement, for instance due to unideal wafer loading process, the clients are responsible for calculating the corrected destination.

The absolute positioning works as follows. Upon receiving requests along with information about positioning specification from the clients, the Wafer Positioning and Scanning Application parses the information to get positioning adjustment parameters and performance properties for each axis (in 6DoF). This application's main job is to forward this request to the Peripheral, which represents the main actuator. To get which Peripheral is in charge, the application looks at the application control mode specification.

2.4.2. Synchronized Positioning

Subsystems inside the TWINSCAN machine need to work together to perform their tasks. For instance, during a photolithography process, the Wafer Stage, the Projection system, and the Illumination are working together. If a subsystem does not work properly, then the image is not perfectly aligned and the ICs are considered as defects.

For these subsystems to work together in harmony, a synchronization mechanism depicted in Figure 2-6 is used. This figure uses notation similar to the UML sequence diagram for simplicity. The arrows represent communications between the Synchronization component and the Subsystem, and the vertical lines shows the chronological order of the communication. The box on the vertical line represents a process execution. This synchronization mechanism comprises of two processes, namely the negotiation process and the execution process. This process is orchestrated by a piece of software called Synchronization component. During the negotiation, a Synchroniza-

tion component asks all subsystems joining a synchronized action to give the estimation of the time needed to perform such action and the time needed to prepare the action. After the Synchronization component gathers all timing information, it asks the subsystems to queue the action and sends the timing information. Afterwards, the component triggers the synchronization action and the subsystems execute the actions that have been queued. Once the actions are performed, the clients can get the result via the Synchronization component.

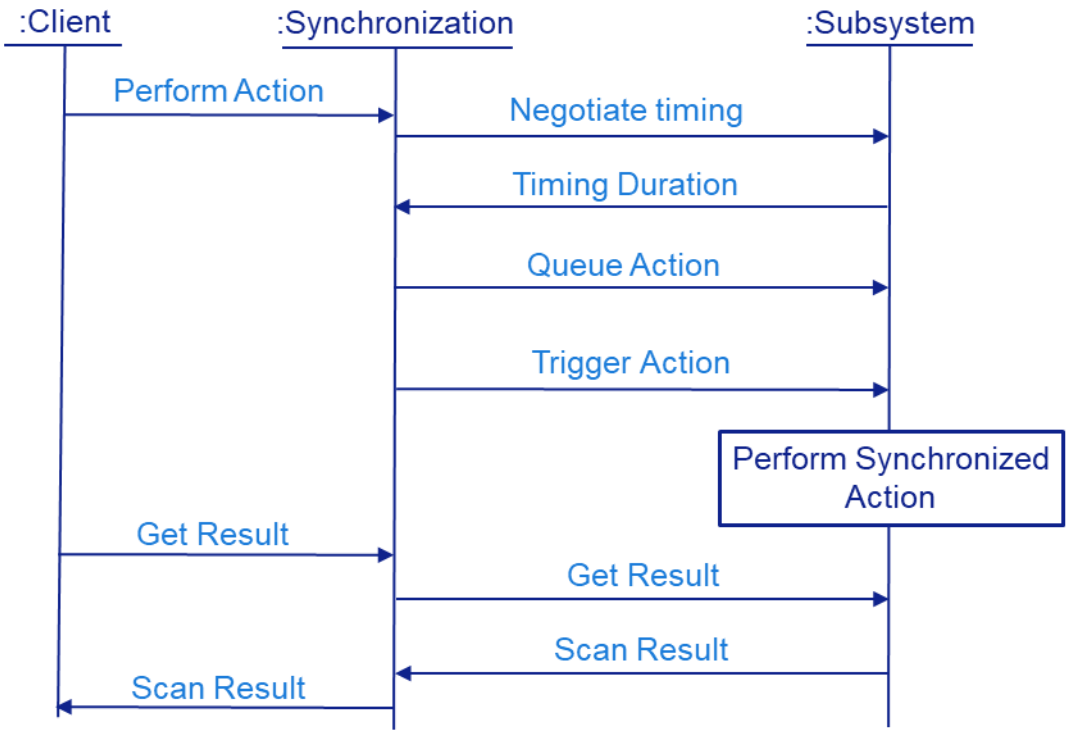

Figure 2-6 Synchronization action process: timing negotiation, queue action, trigger queued actions, and get action results

The Wafer Positioning and Scanning Application is responsible for providing the synchronization facility and interface for the Synchronization component. The job description of the application is actually similar to when it serves an absolute positioning request. The application needs to process the data provided by the client and forward it to the peripheral. The difference is that the Scanning Application has to handle requests of different synchronized action types.

Wafer Positioning module offers various types of synchronized actions. Depending on the types, the Scanning Application has to serve the request differently. For instance, a synchronized action type requires the application to override default synchronization settings while another type does not. Another example is that a certain type requires pre and post-action to be executed before forwarding the request to the Peripherals.

In addition to the synchronized action types, the Wafer Positioning module also offers several types of movement for each axis. For instance, there is a straight forward type movement, where the Chuck moves a particular axis linearly from position A to position B. More complex type movements, for instance circular movement, are also offered by the Wafer Positioning module [2]. Each type of movement has different set of parameters and must be handled differently.

2.4.3. Position Information Provider

The clients of the Wafer Positioning module need position related information from the wafer positioning module to fulfill their business goal. For instance, to determine a new destination, the clients may need information about the current position of the Chuck. The requested information includes the coordinate system, last set point and the state of the Wafer Positioning module. The clients request this information from the Wafer Positioning and Scanning Application.

The application is responsible for providing the information as it knows where to get the information from. For instance, information about the actual position of the Chuck is obtained from the Peripherals. Furthermore, the Wafer Positioning and Scanning Application caches some of the information in the Application level, such as the coordinate system.

2.4.4. Currently Used Design

The Wafer Positioning and Scanning Application was designed using pseudo-Object-Oriented (OO) style. The system was decomposed into classes based on the functionalities. However, object-oriented concepts (e.g., polymorphism or abstraction) and design techniques (e.g., design patterns) were not applied. Furthermore, the design was implemented in the C programming language.

Figure 2-7 shows the functional decomposition of the Wafer Positioning and Scanning Application[4]. As can be seen from the figure, the application is broken down into several parts based on the functionalities. The absolute positioning functionality is handled by the Positioning Application part; meanwhile the Scanning Application serves requests related to synchronization actions. Other than these two parts, there are the Control Mode part that abstracts the control modes of the peripherals and the Constraint part that manages movement constraints. The Wafer Positioning and Scanning Application class provides external interfaces for the clients. It behaves as a façade [3], which delegates function calls from the clients to the inner software elements.

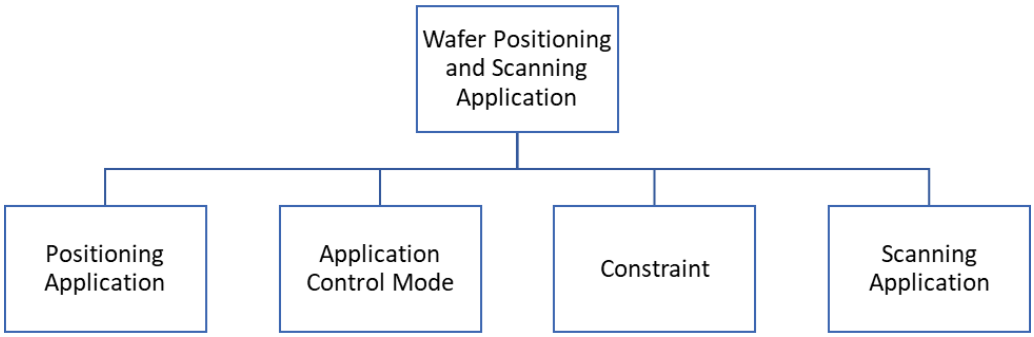

Figure 2-7 Functional Decomposition of the Wafer Positioning and Scanning Application

[4] This is figure was obtained by reverse engineering the source code. Only high-level functional decomposition is presented on the figure.

The absolute positioning functionality of the Wafer Positioning and Scanning Application is straightforward as it only forwards the request to the peripheral with minimum to no data processing. The synchronized positioning functionality of the Wafer Positioning and Scanning Application is not as simple as the absolute positioning functionality. As mentioned in the previous section, to perform a synchronized action, the clients depend on the Synchronization component, which is outside of the Wafer Positioning and Scanning Application. Before the action can be executed, negotiation to get timing information happens between the client and the Scanning Application.

Figure 2-8 shows a UML activity diagram to illustrate how the Scanning Application handles a synchronization request[5]. The boxes represent processes and the arrows represent the process flow. In general, upon receiving a request, the Scanning Application parses the information from the client. Afterwards, the Scanning Application uses both client information and default scan information to compose action parameters. These parameters include the destination specification, active constraints, and performance properties. Next, the application determines the control mode switching. Finally, the application forwards the request along with the action parameters to the peripherals.

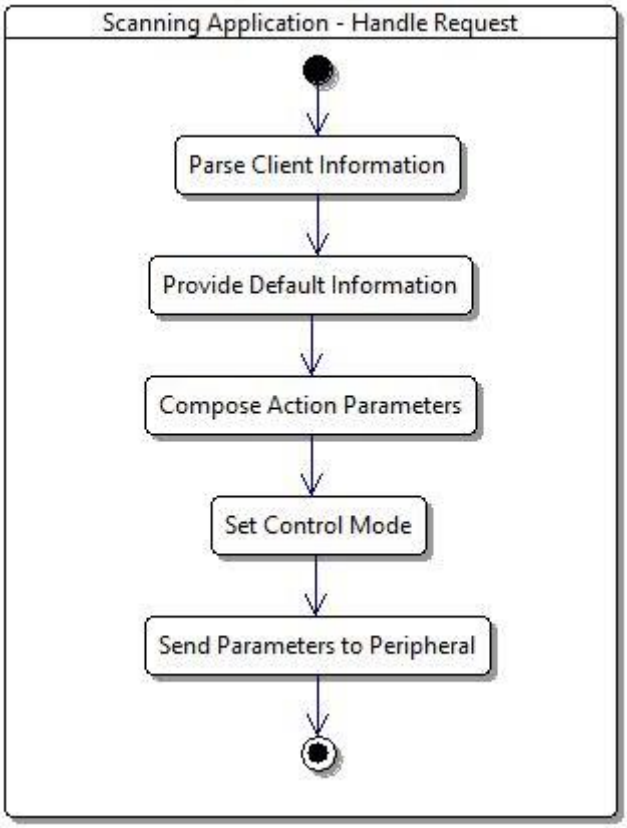

Figure 2-8 Process of handling synchronized action request inside the Scanning Application

[5] This figure is obtained by reverse engineering the source code

3. Problem Analysis

ASML always strives for producing better machines, which achieves higher throughput and smaller dimension. When new technologies are introduced, the subsystems must be adapted and evolved to accommodate the changes. For instance, when a more advanced positioning control algorithm was introduced and better technologies are supported, the Wafer Positioning module had to consider not only the velocity, acceleration (first derivative of velocity), and jerk (second derivative of velocity), but also snap (third derivative of velocity).

Over the years, the Scanning Application has grown into a complex class (i.e., it has 77 methods and ten thousand lines of code) as new requirements are incorporated. With the insights that have been accumulated about this component, ASML concludes that redesign is necessary. This chapter focuses on analyzing the current state of the Scanning Application to find improvement points as bases to redesign the component. The main goal is to have a new design, which incorporates the insights that have been accumulated over the years and implement the new design with an improved tooling.

3.1 Problem Definition

The Scanning Application is responsible for providing synchronized action interfaces for the clients and composing action parameters to send to the peripherals. The currently used design puts all responsibilities inside a single class. As a result, the Scanning Application class becomes a class that holds too many responsibilities. Having too many responsibilities could indicate that the number of methods inside the class is high, the cohesion level between methods is low, and the coupling is high (as many classes would need to use the class). The number of methods inside a class, the lack of cohesion between methods, and the coupling between classes/objects are well known metrics suite for object-oriented design [4], which are related to the design complexity [5].

The Scanning Application is implemented in the C programming language, which is not optimal to realize object-oriented design (i.e., extra code is needed to realize some OO concepts, such as abstraction and polymorphism). The implementation has led to a situation where many branches and code duplications (to implement different synchronized action and movement types) are abundant (i.e., more than 10% code duplication according to TIOBE code quality measurement tool [6]) because the class abstraction and object instantiation concepts do not exist in the C language. As a result, the cyclomatic complexity [7] increases (i.e., average score is 12 according to TIOBE tool), showing that the implementation is complex. This situation, together with the complexity of the design, makes the component less flexible, which increases the cost to incorporate new requirements into the component.

The complexity of the design and implementation of the Scanning Application that accumulated over the years brings technical debt. First, the learning curve to understand this component is steep. Second, it is hard to apply unit test to the component. Third, the cost to incorporate new requirements is relatively high.

3.2 Project Goal and Scope

The goal of this project is to redesign the Scanning Application component using the object-oriented paradigm to reduce the complexity so that its maintainability quality in terms of modifiability, testability, and analyzability increases without compromising the current functionality and performance. To achieve the goal, the following tasks were formulated:

1. Capture functional and non-functional requirements of the Scanning Application from the source code, domain analysis, and experts;

2. Redesign the software component using the object-oriented modelling techniques while fulfilling the functional and non-functional requirements;
3. Implement a prototype of the new design using C++ language;
4. Validate the new design against the main use cases of the Scanning Application.

The scope of the project does not include software components that utilize Scanning Application (clients). Meanwhile, peripheral components (hardware drivers used by the Scanning Application) could be considered inside the scope if necessary.

3.3 Assumptions

The assumptions made for this project are as follows:

1. The current implementation of the Scanning Application (source code) is the newest version of the component documentation.
2. The basic functionalities of the component have no major change from the Element Performance Specification (EPS) document (2011) [2].
3. The design and prototype implementation will be tested in the NXT machine testing environment.
4. There will be no major machine architecture changes during the project period.

3.4 Constraints

This project has several constraints to limit architectural decisions. These constraints are as follow:

1. The external interfaces of the component should remain the same. The clients should not need to change anything to use the component.
2. The implementation of the new design should use C++ language with ASML tool [8].
3. The duration of the project is 10 months.

4. Stakeholder Analysis

Every system has parties that could affect or be affected by the system. These parties are usually called stakeholders. Stakeholders could be organization, team, or people. All stakeholders have concerns that need to be addressed during a system design. However, since there could be too many stakeholders (and their concerns) in a project, addressing all concerns is very difficult if not impossible. Hence, system designers need to prioritize stakeholders based on their importance and take into account their concerns during a system design. This chapter presents the main project stakeholders that have been identified. Later, their concerns are gathered, and communication channels are defined.

4.1 Stakeholder Identification

In this project, a stakeholder analysis matrix [9] was used to help identifying important stakeholders. This matrix classifies stakeholders into four categories based on their interest and power towards the project. These categories become a reference to engage the stakeholders appropriately. For stakeholders who hold high power and high interest, they need to be managed closely. For stakeholders who hold high power and low interest, they need to be kept satisfied. For stakeholders who hold low power and high interest, they need to be kept informed. Finally, for stakeholders who hold low power and low interest, they need to be monitored.

Figure 4-1 shows the stakeholder analysis matrix for this project. The horizontal axis represents the amount of interest a stakeholder has toward the project. Meanwhile, the vertical axis represents the power a stakeholder holds that could affect the project. As can be seen from the figure, several stakeholders were identified and mapped into the matrix.

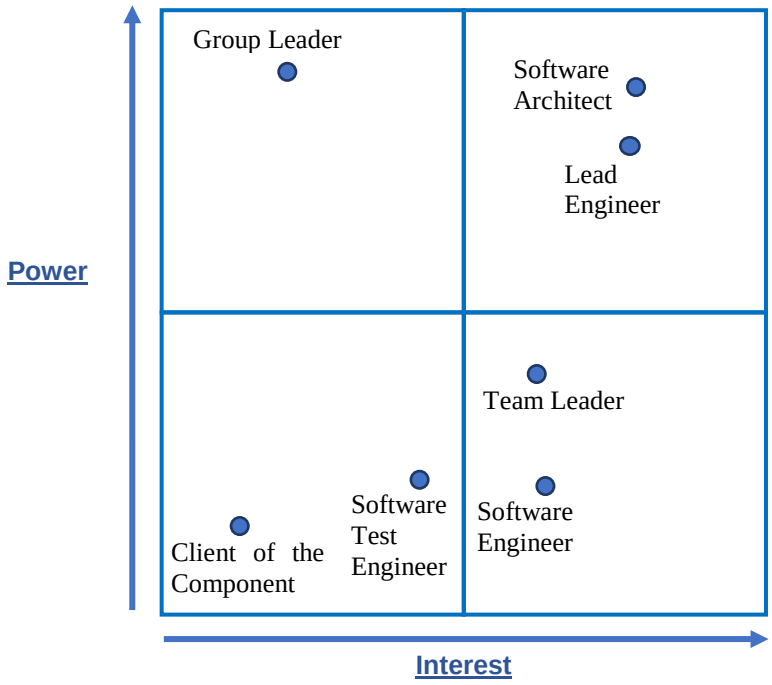

Figure 4-1 Stakeholder analysis matrix

4.2 Stakeholder Concerns

As mentioned, all stakeholders have concerns. Their concerns are summarized in Table 4-1.

Table 4-1 Stakeholder Concerns

No	Stakeholders	Concerns
1	Software Architect (ASML)	• Make sure that the engineers quickly understand a design • Solve the problem in the current design • Make sure that the solution has the same functionality and quality as the current design
2	Lead Engineer (ASML)	• Give the engineers direction for some implementation problems • Help the engineers to understand a design and component as quick as possible • Easily incorporate a new requirement into a component
3	Group Leader (ASML)	• The project schedule and planning are sensible • The deliverables are met within the time limit
4	Team Leader (ASML)	• Help a new team member to understand the wafer positioning domain faster • Protect the team from external disturbances • Make sure that the quality of work of the team is acceptable
5	Software Engineers (ASML)	• Understand a domain and design quickly • Do not need to change many parts when updating a component • Implement a design correctly
6	Test Engineers (ASML)	• Make sure that the engineers implement a design correctly • Perform test to a component easily
7	Clients of the component (ASML)	• Does not need to change anything if the design of the wafer positioning application change

4.3 Stakeholder Involvement and Communication Plan

The key stakeholders in this project were the Trainee, Software Architect, and Lead Engineer. These key stakeholders were involved in the decision-making process for this project. There were several possible communication channels between these stakeholders. Direct verbal communication was mainly used for short discussions and questions. In case verbal communication was not possible, communication via Skype message and email were always available. In addition, weekly meetings were organized to discuss important topics, such as decision to make or design proposal.

The Group Leader held a huge authority that could affect the project. However, they were not interested in the technical detail of the project. They were involved in the decision-making process related to their area of interest. A communication via email was preferred for the Group Leader.

The team leader and the software engineers had less power but high interest. They were involved by listening to their concerns and updating the status of the project occasionally. The clients of the component were the least important stakeholder because the nature of this project (redesign). Ideally, they do not need to know that the component changes and should be able to use the interfaces of this component like they normally do. They could be reached via direct verbal communication, skype message, or email.

5. System Requirements

This chapter discusses the system requirements of the project. These requirements were formulated after investigating the source code, reading documentation, and discussing with the experts. The requirements fall into two categories: functional and non-functional requirements. The functional requirements describe the tasks that the system should accomplish. These requirements also define the behavior of how the system completes the task. On the other hand, the non-functional requirements describe the quality of the system and other requirements related to constraints. However, before going into the requirements, the use case scenarios of the Scanning Application is discussed.

5.1 Use Case Scenarios

Use case scenarios show how external entities interact with the Scanning Application. Figure 5-1 shows the UML use case scenarios for handling synchronized action request[6]. To perform a synchronized action, the client must negotiate the synchronization timing with all participating subsystems. In case of Wafer Positioning module, the client asks the Scanning Application to calculate the duration of a synchronized action. Afterwards, the application composes parameters and asks the Peripheral to get duration of the described action from the actuator controller.

Once the client gets the duration, it asks the application to calculate the duration needed by the Wafer Positioning module to prepare for the synchronized action (e.g., duration to move to the start position). The application follows the similar step as in calculating the action duration.

After getting timing requirements from all subsystems, the client determines the overall action and preparation duration of the synchronized action and queue the action. Afterwards, the action is triggered by the synchronization bus. Finally, the client waits the action to finish and retrieves the result via the application.

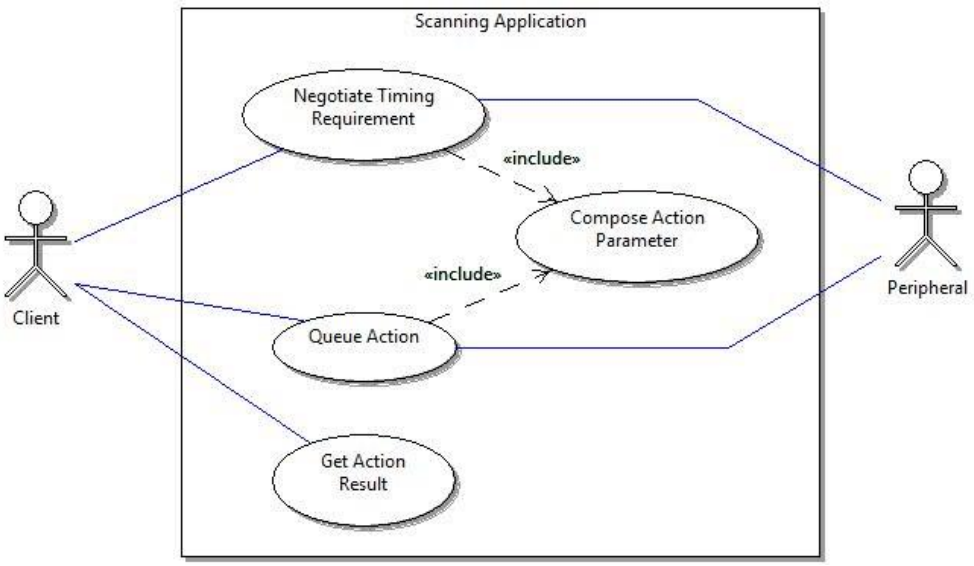

Figure 5-1 Use Case Scenario of handling synchronized action request

[6] This figure is obtained by analyzing the documentation, asking the domain experts, and reverse engineering the source code.

5.2 Functional Requirements

The functional requirements for the Scanning Application can be seen in Table 5-1. These requirements are informal requirements formulated from asking the domain experts, reverse engineering the current implementation, and reading the documentation. In general, the Scanning Application must provide interfaces for the clients to negotiate timing, queue action, and get action result (R1). Upon serving a request via the interfaces, the application must compose action parameters to send to the peripheral (R2-R6). Also, the Scanning Application is tasked to forward the request together with the parameters to the correct peripheral (R7).

Table 5-1 Functional requirement of Scanning Application

No	Requirements
R1	Scanning Application shall provide synchronization facility interface for the clients.
R2	Scanning Application shall be able to queue multiple scans without having to retrieve results in between.
R3	Scanning Application shall provide default scan related information.
R4	Scanning Application shall ensure that the peripheral is in the correct setting before/after performing a synchronized action.
R5	Scanning Application shall determine the parameters that need to be sent to the peripheral.
R6	Scanning Application shall handle action related to synchronized action result when a synchronization action is finished.
R7	Scanning Application shall forward incoming requests to the correct peripheral.

5.3 Non-functional Requirements

Table 5-2 lists the non-functional requirements of Scanning Application. These requirements determine the quality of the Scanning Application in the context of this project. These requirements were gathered from the system analysis and discussion with the domain experts.

Table 5-2 Non-functional requirements of the Scanning Application

No	Short Name	Requirement
NF1	Performance	The performance different between the currently used design and the new design should be less than 1 millisecond.
NF2	Modifiability	The Scanning Application shall be extensible in term of features.
NF3	Testability	The Scanning Application shall be able to be tested with unit tests.
NF4	Design constraint	The Scanning Application shall adhere to ASML design constraint and guidelines [8].
NF5	Interfaces	The external interfaces of the Scanning Application must not change.

6. System Design

This chapter discusses the new design of the Scanning Application. This discussion covers the design methodology used in this project, the plan to integrate the new design into the existing system, the design strategy, the object-oriented analysis and design of the Scanning Application, some design alternatives and the non-functional requirements.

6.1 Design Methodology

In this project, the iterative design development was intended as the process would be easier to control. The design method used in this project is called Attribute-Driven Design (ADD) [10]. This method is an iterative method in which a design milestone is produced at the end of each iteration. This design milestone can be a system decomposition or a design of a system element. ADD defines steps in each iteration as follows:

1. Select a part of the system to design.
2. Trace all architecturally significant requirements (ASR) for that part.
3. Create and test a design of the selected part.
4. List all remaining requirements and select a part of the system for the next iteration.

In this project, each iteration lasted for four weeks. Figure 6-1 shows the initial milestones for the iterations. The first milestone was component decomposition, which divided the software component (the Scanning Application) into smaller parts. In the next iteration, second design milestone was planned. These iterations kept going until there was no part left to design. The milestones could change depending on the result of at the end of iterations.

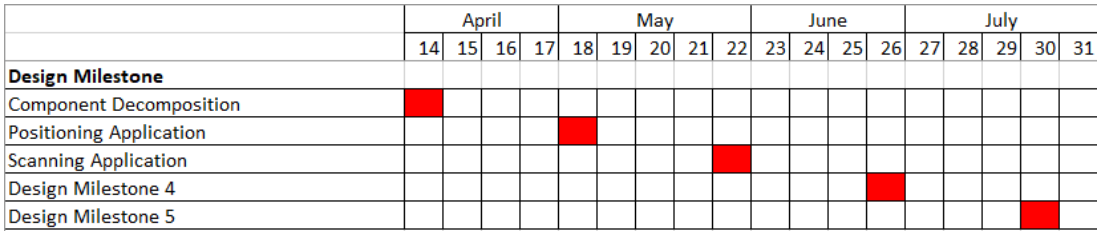

Figure 6-1 Initial design milestones

A design milestone is accepted if it passes the following general criteria:

1. The design is documented.
2. The design is reviewed.
3. The design is approved by the key stakeholders.

When a design milestone does not pass the mentioned criteria at the end of iteration, the milestone is discussed since it may be too big to be completed in an iteration and needs to be split into smaller design milestones.

6.2 Integration Plan

The Scanning Application is a part of a layered software architecture [10]. The advantage of the layered architecture is that a layer can be modified without affecting the layers above or below by making sure that the interfaces and the behavior stay the same. Consequently, this project forbade modifying interfaces to other layers.

Keeping the interfaces the same made software integration in the layered architecture simpler. To enable frequent integration that fits within iterative way of working, a method called Strangler Application was used. The idea of the strangler application is to defer interface calls to the newly redesigned part of the system while letting the old component coexist to serve the component that has not been redesigned. This concept is similar to Façade [3] design pattern. Once the new part is fully integrated, the old component part is removed. The process repeats itself until the old component are replaced completely.

6.3 Design Strategy

The goal of this project is to improve the modifiability, analyzability, and testability of the Scanning Application by redesigning. These quality attributes are strongly related to the complexity of a software design. In general, the more complex a software design is, the harder it is to analyze, to modify, and to test. Although there is no silver bullet for reducing the complexity of a software design, the following tactics are commonly used while creating a software design [10].

Distribute the responsibilities – The size of a class does not necessarily mean that the class is complex. However, the size could indicate that the class might have more than one responsibility. Besides, having too many lines of codes (e.g., implementing many responsibilities or having many branches) in a file makes it harder for the developers to understand the source code. Reducing the size of the class by distributing the responsibilities helps to increase the readability of the source code. Moreover, a smaller class is easier to test as it has fewer paths to cover.

Reduce coupling between the classes – Strong coupling between classes or objects is the root cause of a ripple effect while modifying software components. Coupling can be seen as a dependency between classes. The more a class is using methods of other classes, the stronger the coupling. Reducing the coupling between classes helps improving locality of change. Using abstraction is one of the common methods to reduce coupling between classes.

Increase cohesion inside a class – Cohesion can be seen as how relevant a method with other methods inside a class is. Lack of cohesion could indicate that the class attempts to handle too many responsibilities, which are not related to each other. Meanwhile, ideally, a class should only have a single responsibility. Keeping high cohesion boosts analyzability and encapsulation.

In addition to these tactics, a design principle called SOLID principle [11] is used as a guideline when designing an object-oriented design. The SOLID design principle consists of the following aspects.

Single responsibility – a class should have only one responsibility. Having a single responsibility inside a class enhances cohesion as functionalities related to that responsibility are group together. It also enhances the encapsulation of the class and prevents defect from rippling across other modules.

Open-close principle – a class should open for an extension and close for modification. This principle means that upon incorporating a new requirement, a class or module should be able to add the requirement by extending its behavior rather than changing the source code. This principle can be achieved by using dynamic polymorphism and static polymorphism techniques. Dynamic polymorphism is a technique where different form of concrete class with the same interfaces is instantiated during runtime.

This technique is commonly called the Strategy pattern [3]. Static polymorphism is a technique in which templates or generics are used to realize polymorphism. While, complete open-close principle is difficult to be achieved, partial application of this principle can make significant improvement in the structure of an application [11].

Liskov substitution – a child class should be substitutable for the base class. Inheritance is one of the object-oriented concepts where a class inherits behaviors of another class. The rule of thumb of the inheritance relation is when a class has an is-a relation to another class. For instance, a cat is an animal, hence a cat inherits the behaviors of an animal. This relation, however, is not always correct in the context of object-oriented design. For example, circle is a special kind of ellipse where the foci coincide [11]. This relation is semantically correct. However, the Liskov substitution principle is not fulfilled as not all behaviors of an ellipse can be realized with a circle (e.g., set two different foci).

Interface segregation – having many client specific interfaces are better than one general interface. A general-purpose interface typically has many arguments as the interface is expected to provide many functionalities. Moreover, a general-purpose interface would normally cause a control coupling where the behavior of a class or module is decided by the client. To avoid this problem, the general interface should be divided into more specific interfaces that provide specific functionalities. Besides the functionalities, the segregation can also be focused on the clients (i.e., each client has its own interface).

Dependency inversion – a class should depend on the interfaces instead of concretions. Interface is generally more stable than a concrete class. Hence, depending on the interface makes the design more flexible as the change in the concrete class does not require change from the client class.

6.4 Analysis and Design

The Scanning Application is a software component that handles requests from its clients, performs data processing, and sends the parameter to the next software component. Based on the description, this application can be modelled by Figure 6-2. This figure shows hierarchical structure of the Scanning Application using UML class diagram notation. This model is obtained as a result of analyzing the domain, consulting the domain experts, investigating the source code, and reading the documentation. By referring to the classes and their relationship, a UML class diagram can be produced.

Basically, the application receives information relevant to a request from the client and parses this information. For each request, there can be one or many information passed by the client. Upon receiving a request, the application composes necessary action parameters (e.g., axis adjustments) using the client information and the default action information of that specific request. The number of parameters created varies depending on the request types. Finally, the application forwards the request along with the composed parameters to the peripheral.

In this configuration, the Scanning Application is burdened by the tasks of handling client's requests, composing parameters, and forwarding the requests. In addition to that, this application also manages all action parameters of the requests. The coupling between the application and the action parameters is concerning as the application will have to adapt if the action parameters change. In this situation, encapsulating the requests into objects that hold their own parameters seems to be a good idea.

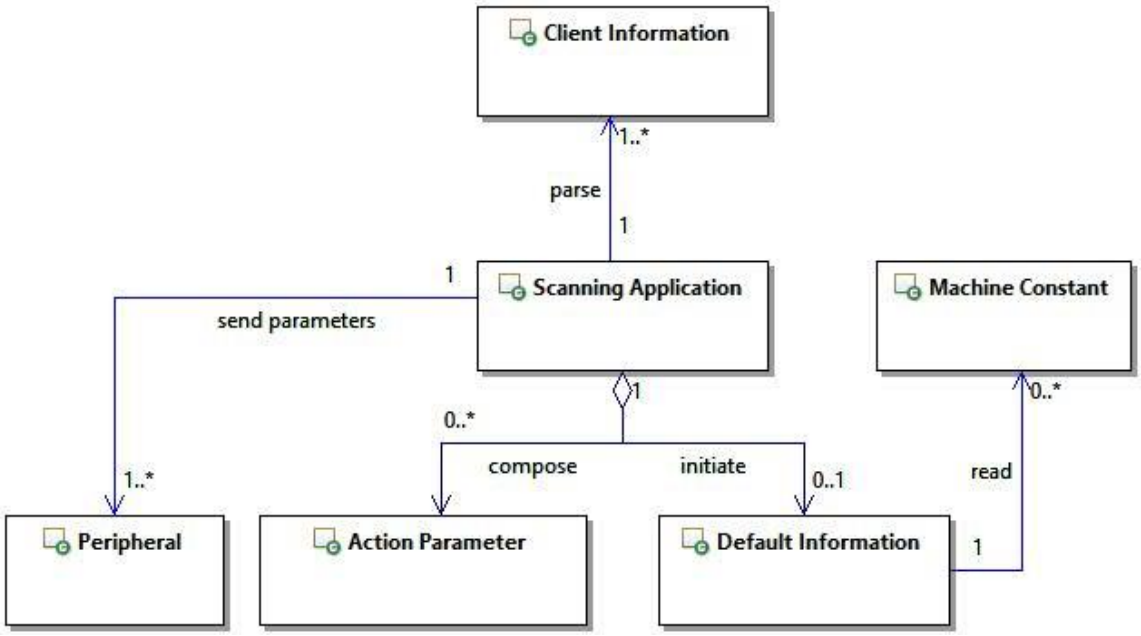

Figure 6-2 Hierarchical structure of the Scanning Application

SA Request Encapsulation

Figure 6-3 shows the class diagram of the Scanning Application with request encapsulation. In this design, the Scanning Application delegates the task to compose the action parameters to the request objects. This design limits the responsibility of the application to only providing interfaces for the clients and handling only a single request at a time. The design shifts the task to manage multiple request objects to the Scan Admin. Making a request into an object is commonly known as the Command design pattern [3].

The encapsulation cuts the coupling between the Scanning Application and the action parameters. The SA Request class provides an abstraction for the Scanning Application to compose the action parameters. This encapsulation also enhances the cohesion as now the relevant information and methods are grouped in a request class.

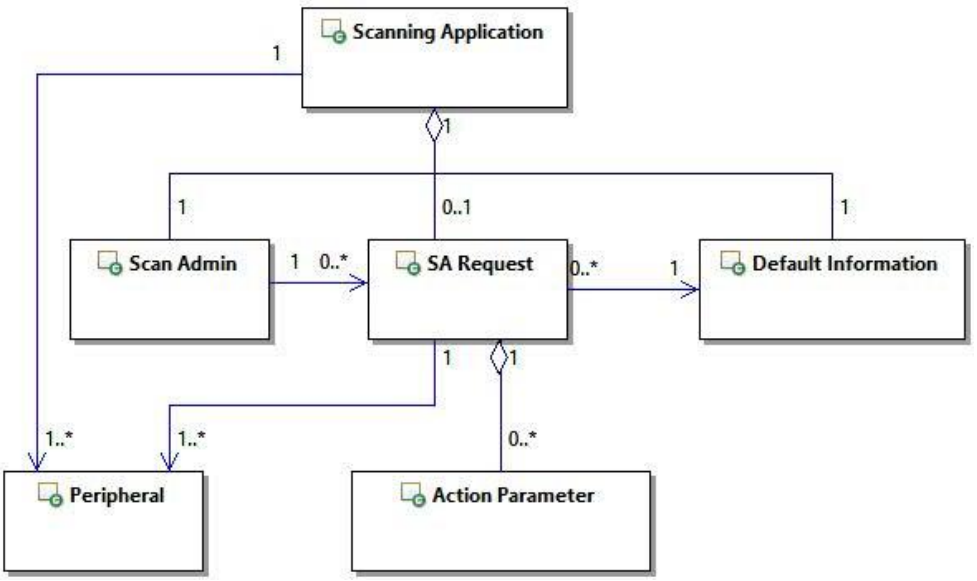

Figure 6-3 Scanning Application UML class diagram with SA Request encapsulation

With the encapsulation, the way the Scanning Application internally processes requests is now slightly different. Figure 6-4 illustrates the sequence diagram when the Scanning Application handles synchronized action request. Upon receiving a request to negotiate the timing requirement or queue an action, the application inquiries the Scan Admin to get the object of the request. Each request object is unique and identifiable by its action id. In case the object does not exist, the Scanning Application instantiates a new request object and registers it to the Scan Admin. Afterwards, the Scanning Application invokes the negotiate or queue method from the request object and lets the request object handles the rest.

During the creation of the SA Request object, the action parameters are also composed. This process is hidden from the Scanning Application to make an abstraction. When the application handles a request with respect to an already created request object, for instance to queue an action after timing negotiation, the application only needs to call the method from the request object.

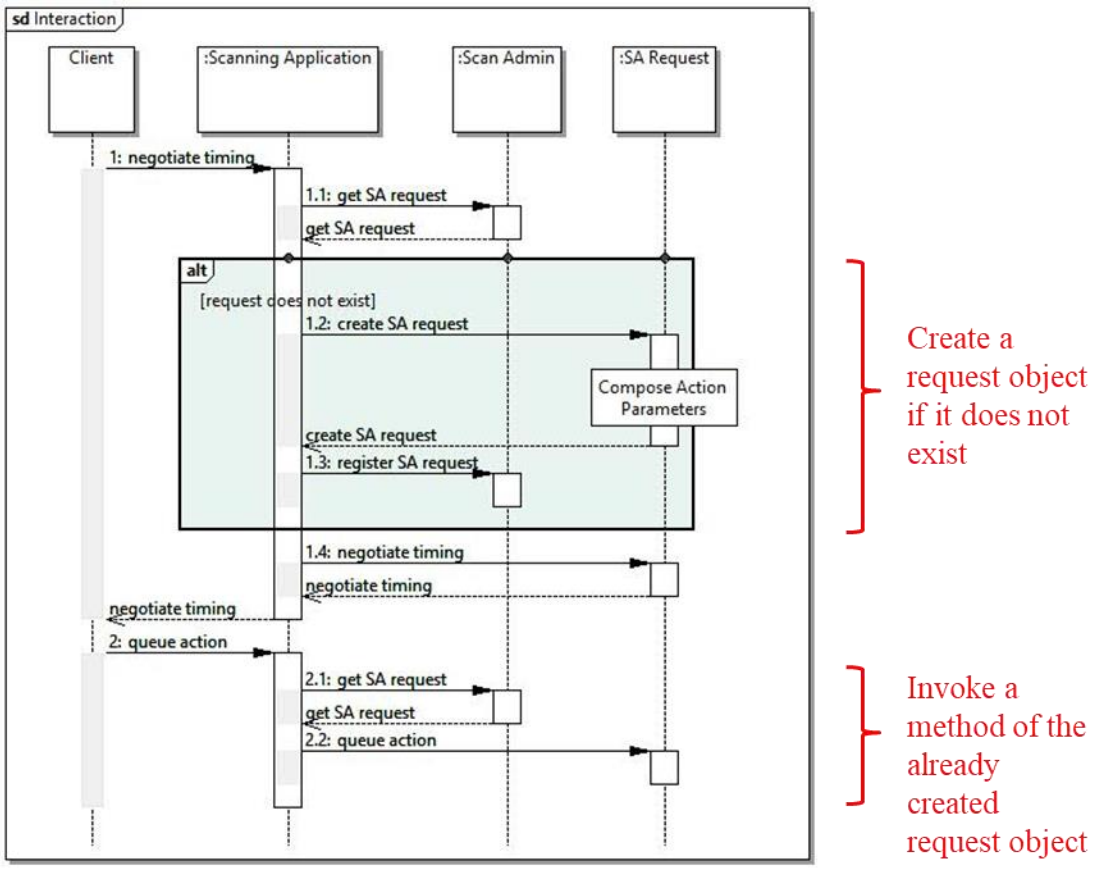

Figure 6-4 UML sequence diagram of the Scanning Application with SA Request object when handling synchronized action request

Inside SA Request

The SA Request class is now responsible for creating the action parameters. To avoid this class from being complex, the creation of the actions parameters is delegated to the classes representing the parameters. Within the context of the Wafer Positioning module, the required action parameters are the destination, control mode, constraint, and scan performance properties.

Figure 6-5 shows the hierarchical structure of the SA Request class. The Control Mode class is responsible for determining the control mode setting of the Wafer Positioning module. The Axis Adjustment processes information related to the destination where

the Wafer Positioning module should move to. The Constraint class determines the active constraints for the request. The Scan Performance Properties is responsible for clipping performance to the maximum permitted values. The multiplicity of the parameters is set from zero to many as different types of request may need different number of parameters.

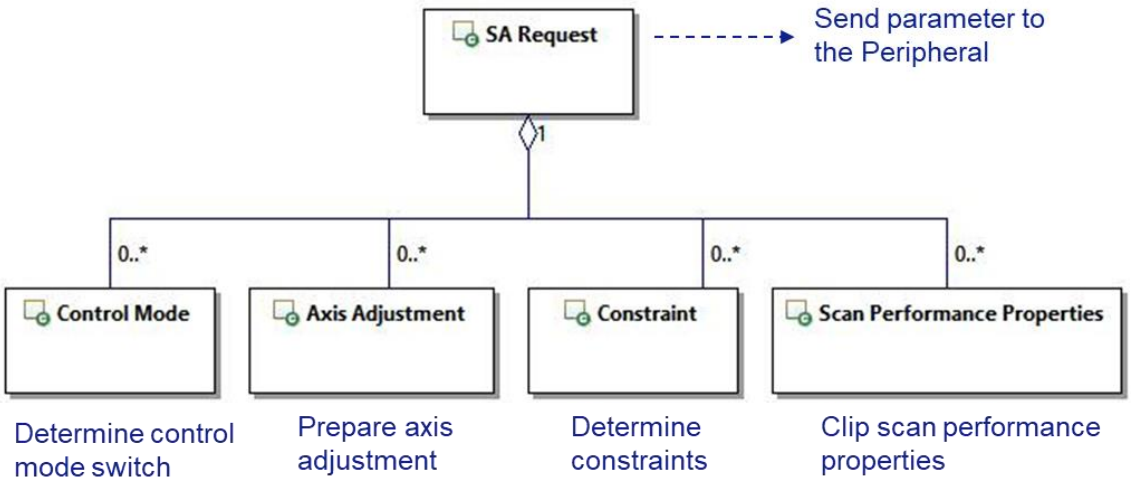

Figure 6-5 UML class diagram of the SA Request

SA Request Variants

There are several variants of the SA Requests. These variants, in general, do not differ much in term of actions needed to prepare the request. The significant difference lies in the information that they receive from the client and the action parameters they need to provide. Figure 6-6 shows how this point is addressed. To accommodate different types of SA Requests, the request class is turned into an abstract class from which concrete request classes inherit the behavior. During operation, the Scanning Application instantiates the concrete request objects depending on the types of the requests and interacts with the request objects via the interfaces. This design follows the dependency inversion principle by making use of polymorphism and inheritance concepts of the object-oriented paradigm.

To further decouple the Scanning Application from the concrete request classes, a creator class called SA Request Factory is introduced. This class provides an interface for the application to instantiate concrete request objects. As a result, the Scanning Application does not depend on the constructor of the concrete request classes and only holds a reference to the creator class. This arrangement utilizes the Factory Method design pattern [3]. This approach is also applied to address the various axis adjustment types.

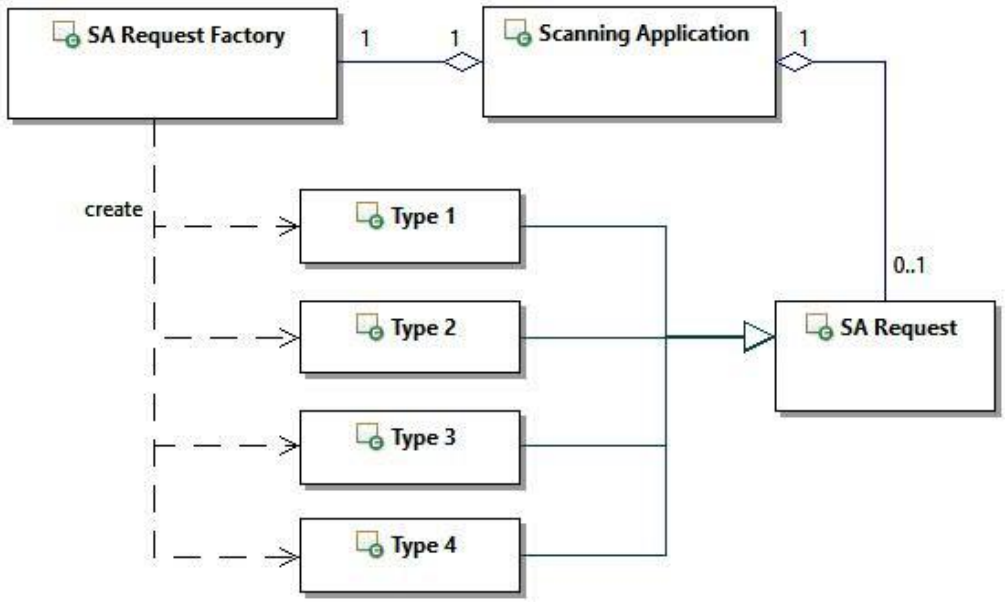

Figure 6-6 UML class diagram of the SA Request variants with the factory method

6.5 Design Alternatives and Decision

Before the design described in the previous section was proposed, several design alternatives were put into consideration. This section discusses the design alternatives and the rationale behind the design decisions.

Creating an SA Request from Scan Admin

The proposed design puts responsibility for creating an SA request object to the Scanning Application as this class has the information needed. An alternative design is to create an SA request from Scan Admin. When the Scanning Application receives a request, this class asks the Scan Admin for that specific request object. If the Scan Admin does not have the object, this class creates the request object and gives the reference to the Scanning Application. Hence, the Scanning Application does not need to know whether the request object exists or not as the Scan Admin always returns the request object reference when asked.

The disadvantage of this alternative is that the Scan Admin has more responsibilities than the proposed version. This class needs to handle information that it does not need (i.e., information about what request object needs to be created). Therefore, the Scan Admin class becomes more complex than it should be (i.e., storing request objects). Furthermore, testing this class becomes harder as more methods need to be covered.

With respect to this alternative, it was decided to create the SA request objects from the Scanning Application. The rationale of this decision was to limit the responsibility of the Scan Admin so that the Scan Admin class is easier to understand and to test. Also, since the Scanning Application holds the information about the request object, it is just logical to create the request object from this class.

Prepare SA Request parameters through function calls

The proposed design prepares SA request parameters during the creation of the request object to hide the process from the Scanning Application. An alternative to the design is to prepare the parameters through functions called by the Scanning Application. This alternative results in less complex SA request object creation. However, the disadvantage is that the Scanning Application needs to know when and what parameters to create. It was decided to prepare the parameters during SA request object creation to

hide its process from the Scanning Application. Hiding this process from Scanning Application is essential to provide a better abstraction.

Abstract SA Request Factory and Parameter Factory
The proposed design separates the two factories for simplicity. An alternative design is to combine the two factories and create a creator class, which the only one knows about the factories. This creator class acts as a mediator, which reduces coupling between the factories and their clients. This alternative design improves the modifiability of the component when a new factory is introduced or when other classes need to use the factory. It was decided to separate the factories because in the foreseeable future, there will be neither a new factory nor a new factory client introduced. Also, the factories and their clients (i.e., Scanning Application and Adjustment Parameter) have a one to one relationship. Hence, the mediator class is unnecessary. Furthermore, separating the factories results in simpler testing.

6.6 Non-Functional Requirement Analysis

To achieve a design that fulfills the non-functional requirements, several tactics were used, such as responsibility distribution, polymorphism, and factory method [3] pattern.

Class Division and Responsibility Distribution
The complexity of the Scanning Application is an inevitable result of putting too many responsibilities in a single class. To reduce the complexity, the monolith Scanning Application class was divided into smaller classes, namely Scanning Application class, Scan Admin class, Default Information class, and SA Request class. The monolith class is divided in such way so that the smaller classes hold specific responsibilities, which can be mapped to the functional requirements. By dividing the class, the complexity of the class decreases, thus increasing analyzability and unit testability. Moreover, since the coupling between classes was kept low, the locality of change, which relates to modifiability, could be achieved.

This division, however, introduces a performance penalty as the number of function calls between Scanning Application and the Peripherals increases. This performance difference is acceptable though as these classes are deployed in a single process, making the additional function calls cost only tens of micro seconds.

Polymorphism
As explained in Chapter 2, the Scanning Application handles several synchronized action types in which the axes could have various movement types. The original design, given limitations when it was created, puts all types inside a single class. When a request comes, the Scanning Application checks the type of the request and prepares action parameters accordingly. This situation results in code duplications and branches. To reduce the code duplication and branches, the abstract SA Request class together with polymorphism were introduced into the design. The concrete request classes are subclasses of the abstract SA Request class. Upon receiving a request, the scanning application instantiates a request object whose type is specified. This configuration is commonly called the Strategy design pattern [3].

Factory Method
To further increase the modifiability and testability, SA Request Factory was introduced. This factory is responsible for instantiating correct SA Request objects based on the requested types. The factory reduces the coupling between the Scanning Application and the concrete request classes. Furthermore, adding a new type of SA Requests becomes simple as the developers only need to create a concrete request class and register it to the SA Request Factory.

6.7 Implementation

In this project, a prototype was built to show that the new design works as intended. There were three kinds of prototypes. The first prototype was for the Positioning Application. Since this component did not change from the design perspective, the implementation only ports them to the C++ implementation. The second prototype was the Scanning Application. This application underwent a massive redesign to achieve the non-functional requirements specified. The third prototype was the integrated prototype to see how these proof of concept implementations work with the complete software system.

The implementation part is only relevant to ASML since the prototype was developed using ASML proprietary technology. A more detailed explanation about the implementation is provided in the confidential part exclusively for ASML.

7. Verification and Validation

The new design aims to improve the modifiability and testability qualities of the software component while maintaining the performance quality and functionality. This chapter presents the verification and validation process used in order to confirm that the new design fulfills the underlying requirements.

7.1 Verification

The design process used in the project was an iterative method. For each iteration, a design element was proposed. To verify the process and the design element produced, weekly meetings were organized. Also, in several occasions, meetings with the experts from related domain were held to verify the design and give design feedback. The verification of the proof of concept implementation is using the same process as the design verification.

7.2 Main use cases functionalities

A prototype implementation was prepared to validate the functionalities. Due to time limitation, only agreed main use cases were implemented and validated. The main use cases are initialization, move to absolute position, get current position, and perform a synchronized action. The functionality validation was performed by using an internal tool called Devbench and Testbench, which allow the Wafer Positioning subsystem to be controlled freely. Devbench is a computer program that simulates the behavior of the system. Meanwhile, Testbench is a testing platform, which uses machine simulation that involves real hardware, hence adds more integrity to the test result. In this setting, the prototype implementation was integrated into the rest of the system so that no stub is needed. Also, the behavior was compared to the original implementation to validate the functionalities.

Figure 7-1 illustrates the interface of the internal diagnostic tool, which shows the state of the Wafer Positioning subsystem after initialization. The left figure shows the state of the application layer, which is the layer where this project was conducted. The right figure shows the state of the peripheral layer, which was unchanged. After initialization, the Chuck 1 moved to position -900 mm in the Y axis. As can be seen in both figures, the application and peripheral layers moved to the designated position. This result validated that the initialization functionality was fulfilled.

Figure 7-1 Wafer Positioning subsystem after initialization

Figure 7-2 shows the interface of the same tool during move to absolute position. The left figure shows the application layer interface where the command to move the Wafer Position to a certain position can be controlled. The figure on the right shows the visualization of the chucks' positions inside the wafer stage. As can be seen from the figure, the chuck moved to the destination filled on the application interface. This shows that the move to absolute position functionality worked. Besides, the get actual

position functionality can be said to be achieved since the tool uses this function to update the field of actual position on the left figure.

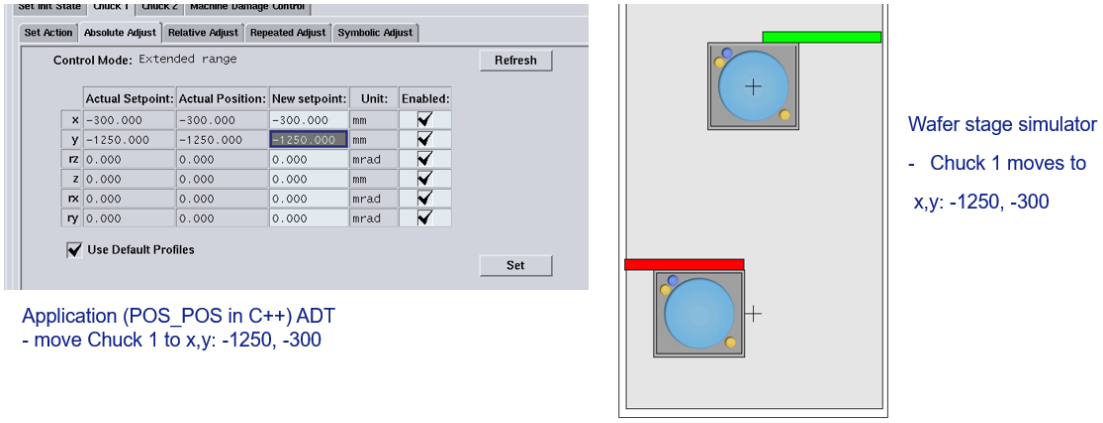

Figure 7-2 Functionality test for move to absolute position

Figure 7-3 shows the functionality validation for the synchronized action. The figure on the right shows the interface of the Scanning Application used to queue synchronized actions. The figure on the left shows the visualization of the chucks' positions. The validation was performed by queueing several actions from the tool, and then executing the action. Afterwards, the tools performed the negotiations and trigger the synchronized actions. As a result, the chuck moved to designated positions based on the queued actions. Finally, all main use cases were successfully validated using the tool.

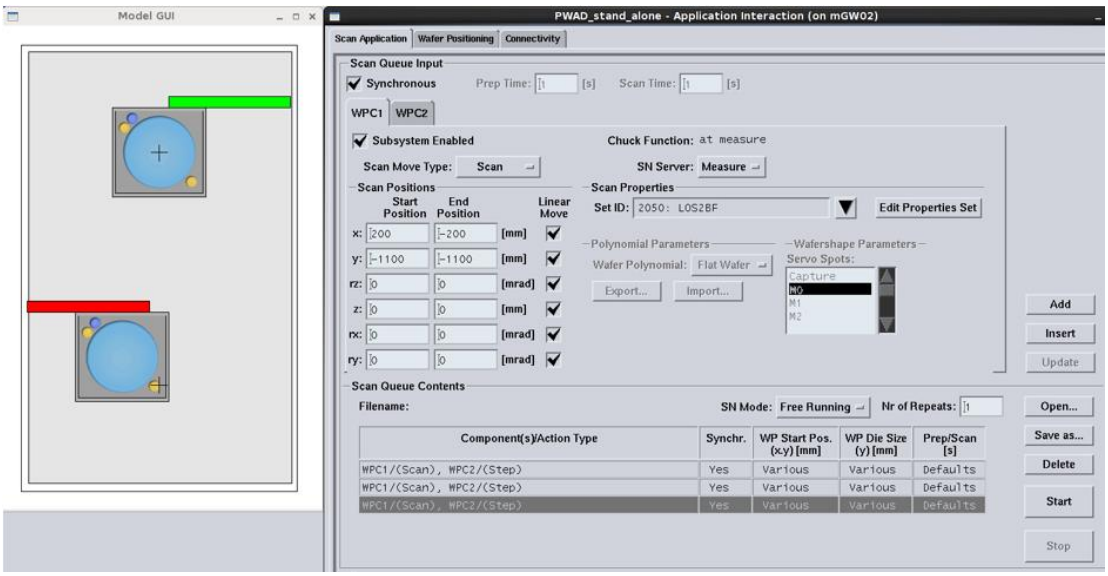

Figure 7-3 Functionality test for synchronized action

7.3 Performance Quality

The performance quality was validated by taking timing measurement from the prototype implementation. The timing measurements were taken using a tool that collects timestamps when a program enters and leaves function. The performance requirement matters only during operational mode. Hence, the initialization performance measurement is not relevant.

Figure 7-4 shows the boxplot of the performance measurement of the move to absolute positioning use case. The vertical axis represents the response time in microseconds. The blue plot shows the performance of the original design and implementation, meanwhile the orange plot shows the performance of the new design and implementation. To produce this plot, fifty measurements were taken. As can be seen from the figure, the new design and implementation appears to be faster than the original one by roughly 100 us (0.11% faster). This result was produced mainly due to implementation differences since the designs were similar. In the C implementation, many if statements were used to check the results of function calls. This is ASML way to handle exceptions. In the C++ implementation, the exception handler is improved by using try and catch, hence, improves the performance.

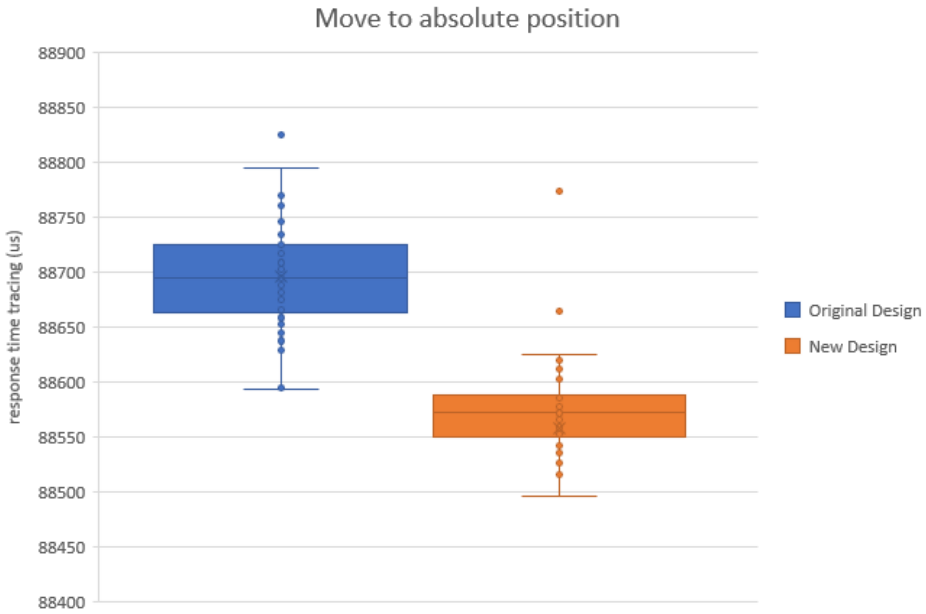

Figure 7-4 Performance measurement for move to absolute position use case

Figure 7-5 shows the boxplot of the performance measurement for synchronized action use case. The synchronized action consists of timing negotiation process (calculate action duration and preparation duration), queue action, and the wait action ready to get the result. The blue plots represent the original design and implementation and the orange plots represent the new ones. The vertical axes represent the response time in micro seconds. As can be seen from the picture, during calculate action duration, the new design was slower than the original one by around 50 us (1% slower). This was because the new design created the request object and constructed the adjustment parameter objects. However, during the calculation preparation duration and queue action, the new design was faster by around 200 to 300 us (4% gain). This improvement was achieved because the new design already had the information about the request and did not need to create or process the information anymore. The wait action ready of the new design was slower by 100 us (3% slower) as the new design needed to construct and copy the result values.

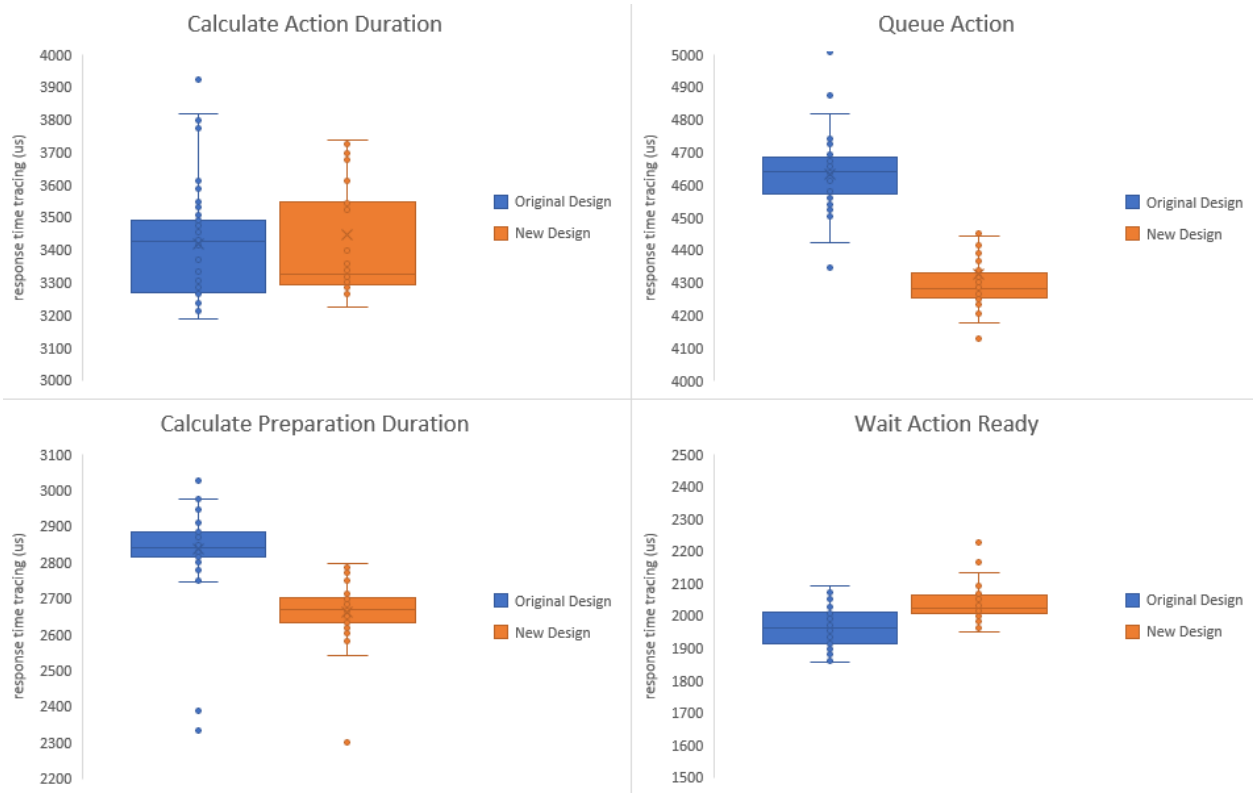

Figure 7-5 Synchronized action performance measurement

Based on the performance measurements for move to absolute position and synchronized action, the new design fulfils the performance measurement. The maximum performance difference was 450 us (4% gain), which was still in the allowed range. Moreover, the performance of the synchronization negotiation improved, and subsequently, could lead to further potential improvements, such as queueing short actions consecutively.

7.4 Modifiability Quality

The validation process of the modifiability quality was performed by means of a qualitative method. This section shows how the new design improves the modifiability of the original design. Compared to the original one, the new design obtained better modularity by dividing the class of Synchronization Application into smaller classes, which hold limited responsibilities. As a result, locality of change is achieved since changes only need to be applied to necessary classes. For instance, Figure 7-6 illustrates the process of adding a new synchronized action. When a new type of synchronized action is introduced, the developers only need to implement the concrete class for the new action and modify the respective factory. The rest of these classes remain untouched.

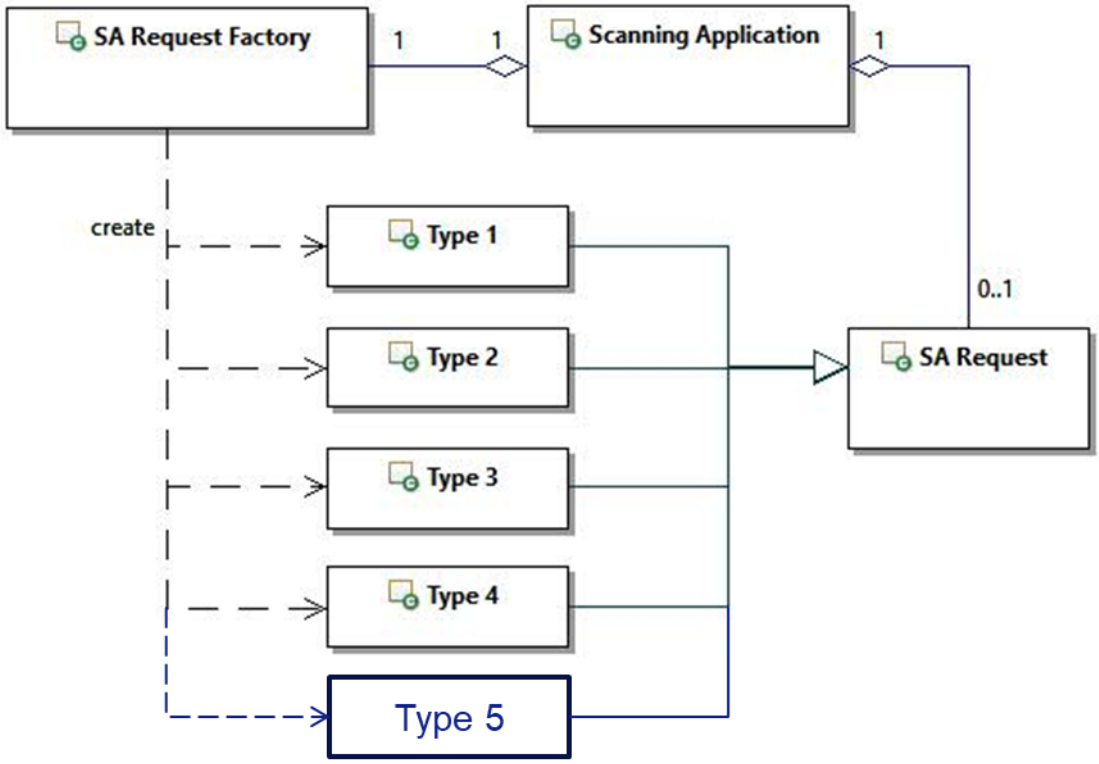

Figure 7-6 UML class diagram of the Scanning Application with respect to the SA Request Factory

7.5 Testability Quality

Similar to modifiability validation, the testability validation was also performed using qualitative method. This section shows how the testability has improved. In the original design, the Scanning Application is designed and implemented in a single class. Since this class is extremely big, applying a unit test to this class is practically too hard if not impossible. The type of test applied to this component is the high-level functional test. By dividing the responsibility, and subsequently limiting the number of methods in a class, performing unit tests is now possible.

8. Conclusion

Inside ASML TWINSCAN machines, the Wafer Positioning module is an essential subsystem that is used by other subsystems. Within this positioning module, there is a software component called Wafer Positioning and Scanning Application, which controls the movement of the Wafer Positioning module. This component provides abstraction for higher-level applications to move wafers at a certain location and time.

While new requirements for the Wafer Positioning and Scanning Application keep coming, the current design of the component does not scale well. This situation results in a superclass that is complex and inflexible to change. Also, since the previous changes were not all documented, the software component becomes difficult to understand.

This project proposes a new design of the Wafer Positioning and Scanning Application using the object-oriented design principles (Section 6.4). In this project, the complexity of the component was reduced by dividing the Scanning Application into smaller classes and distributing the responsibilities evenly. This report shows that each class inside the Scanning Application is small and easy to understand. Another benefit of dividing the Scanning Application into smaller classes is that employing a unit test is now possible.

The code duplications and branches were removed by using the concept of polymorphism and interfaces. This option was not possible in the current C implementation due to ASML software development tools' limitation. To further increase the modifiability and testability, the Factory Method design pattern was used. This pattern concentrates the complexity of the creation in the factory and provides abstraction for the Scanning Application to instantiate Synchronized Action (SA) requests and Axis Parameters.

This report shows that the new design meets the functional requirements for the main use cases (Section 7.2). For the non-functional requirements, the design scales well when a new type of SA request comes. The developers only needed to add a concrete class of the new request and update the factory (Section 7.4). Furthermore, a unit test is now possible with the smaller classes.

The performance suffers a slight penalty during creation of a request object but gains improvement for the rest of synchronization negotiation process (Section 7.3). This penalty is expected as we add new objects in-between calls when serving the request. The performance difference is acceptable as the value falls within the permitted range.

9. Future Work

Due to time constraint, this project could not realize all ideas based on the gained insights. These further improvements are included in possible future work.

1. Redesign the Peripheral level of the driver using the object-oriented concepts. Currently, the Peripheral software components are designed as a single big class. This design potentially leads to similar difficulties that the Wafer Positioning and Scanning Application had. Redesign these components into object-oriented designs could reduce their complexity and improve the analyzability, modifiability, and testability.

2. Make the prototype implementation of the Wafer Positioning and Scanning Application into a production implementation. The prototype implementation meant to be a proof of concept implementation to show that the design works and obtain performance measurements. To make this implementation into a production implementation, the following steps need to be followed. First, implement all features that have not been implemented. Second, perform thorough testing.

3. Come up with a memory management strategy for the component. The new design of the Wafer Positioning and Scanning Application involves creating objects of request. Not handling these objects properly leads to memory leaks.

4. Revisit the data structure inside the Wafer Positioning and Scanning Application. In the project, the way data is structured was taken for granted from the data definition of the previous design. While updating the external data structure has a massive impact on the system, modifying the internal data structure only affects the Wafer Positioning and Scanning Application. Study of the data model could be a possible future work to improve the component.

5. Investigate the possibility of making the solution more general so that it can be applied to other Scanning Applications in different subsystems. The ideal situation would be that the proposed solution is directly applicable for similar applications. However, the solution is designed within the context of Wafer Positioning module. Although the basic principle of the Scanning Application is roughly the same, the details that follow need to be addressed properly. For instance, the Scanning Application of the Balance Mass may have a different set of request types and parameters. To identify what is common and what is unique between the Scanning Applications of the different subsystems could be a sensible way to start.

10. Project Management

In general, successful projects are the ones that are properly managed. Poorly managed project could lead to project extension or, if such luxury is not available, poor quality results. This chapter discusses how this project was managed. The discussion includes the project plan, process and control, and information management.

10.1 Project Plan

"If you fail to plan, you are planning to fail". This quote by Benjamin Franklin indicates how importance a plan is. While some people argue that not everything needs a plan, this kind of project is definitely one of the things people would want to plan. In this context, to plan means to put milestones in a project time frame. While the time frame of the project was a given information since the beginning of the project, the milestones needed to be defined. In this project, the milestones were defined by breaking down the project into smaller division called Work-Breakdown Structure.

Figure 10-1 shows the work structure for the project. As can be seen from the figure, the project was divided into three main categories: documentation, analysis, and design & implementation. For each category, the milestones were defined. Based on these milestones, the initial plan of the project was conceived. As the project unfolded, new milestones (i.e., smaller milestones obtained by further breaking down the milestones) could be added. This mechanism is discussed in the next Section.

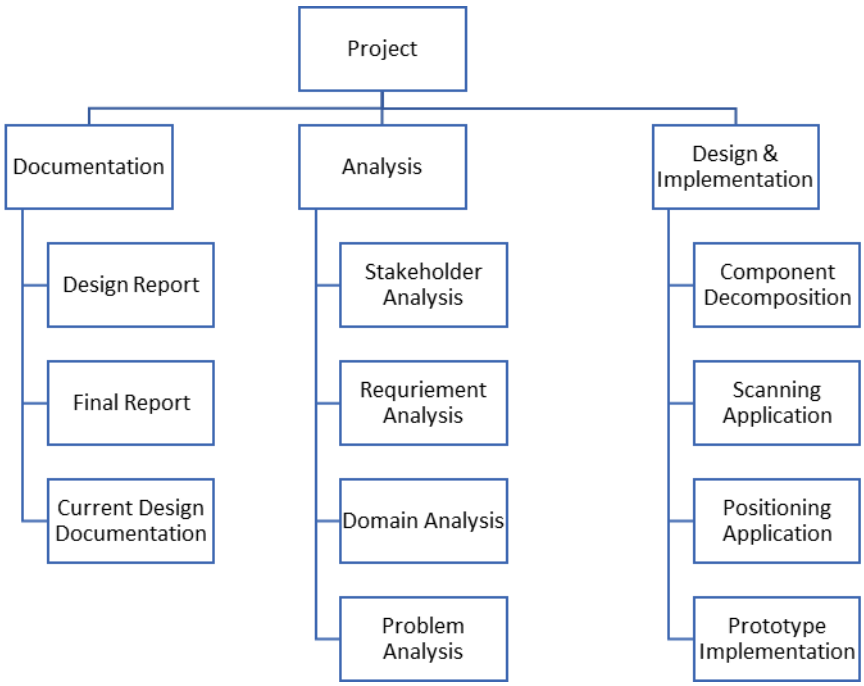

Figure 10-1 Work-breakdown structure of the project

Once the milestones were identified, they were put onto the time frame. The decision to when the milestones were planned was discussed with the key stakeholders. Many considerations can be discussed to determine when is arguably the best time to plan the milestones, such as the importance, deadline, or difficulty of the milestones. In this project, the milestones were mostly planned bases on the priority.

10.2 Process and Control

A plan lays down a path to the end goal, execution gets us there. There is only one problem, good execution often proves to be more difficult than what was expected. To make sure that the project is well executed, the process and control need to be properly defined. This section explains the process definition and control method of the project.

10.2.1. Process Definition

The project was executed in a semi-agile way. The milestones defined could be seen as the user stories. They also had criteria to be qualified as done. For instance, produced documents and proposed designs had to be reviewed by the supervisors and implementation must be tested and approved.

The monthly meeting, where the progress was reported and plan for the next month were presented, replaced retrospective meeting. Often, these meetings resulted in new milestones or changes in the plan. To keep track of these milestones and schedule, Milestone Trend Analysis (MTA) was used. The MTA is explained in Section 10.2.2.

To present the progress and discuss any impediment, weekly meetings were organized instead of daily stand-up meetings. In this way, the iterative aspect of agile can be obtained while efficiently reduced the number of meetings.

10.2.2. Milestone Trend Analysis

Milestone Trend Analysis (MTA) was used to keep track of the schedule. MTA is a graph that is constantly updated during entire project. This graph consists of two time axes: vertical for planning time and horizontal for actual time. For each unit of time, the milestones are evaluated whether they are achieved, delayed, or canceled. After some time, the trend of each milestone emerges. The trends indicate whether the project moves according to plan or delayed.

Figure 10-2 shows the MTA of this project. The vertical axis represents the planned time in week, and the horizontal axis represents the actual project time in week. The diagonal line is where the planned time coincides with the actual time. In other words, placing a milestone on the right side of the diagonal line is impossible as it means planning something in the past.

Each point represents a milestone that is visited each week. If a milestone is not achieved at the planned time, this milestone is updated along with other milestones that depend on it. As a result, the graphs of those milestones move upwards indicating delays in the project. If a milestone does not change, the graph moves horizontally, which means that the milestone is still on schedule. If a milestone is achieved before the planned time, the graph moves down. If a new milestone is identified, a new graph can be added. On the other hand, if a milestone is canceled, the graph of that milestone stops there.

The MTA has several advantages. First, MTA is easy to learn and use. In this project, Excel sheets were enough to realize the MTA. Second, MTA makes the project estimation easier. The trends imply whether the project goes according to the plan or the project needs extension. Furthermore, corrective actions can be executed as soon as the project goes to the wrong direction. Third, MTA is easy to communicate with the stakeholders. The MTA is intuitive and the change in the plan can be seen clearly.

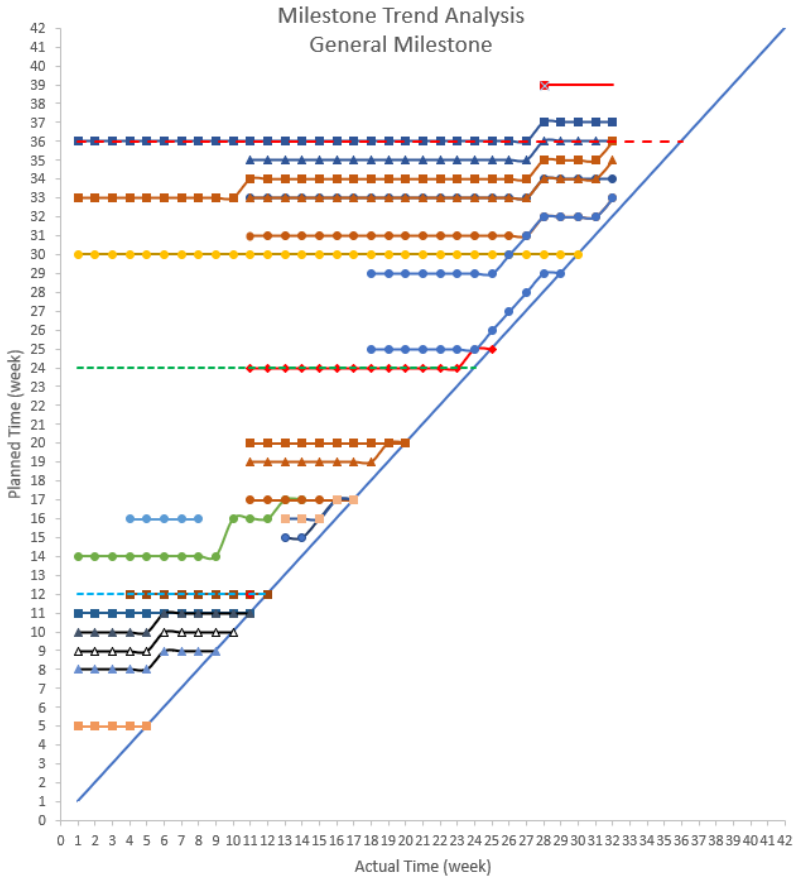

Figure 10-2 Milestone Trend Analysis of the project

10.2.3. Decision and Action Management

A project needs regular progress evaluations to make sure that the project goes to the right direction. Typically, these evaluations result in decisions and action points, which are agreed by the stakeholders. In this project, all decisions and actions were documented in decision and action registers respectively. The decision register is important as a reference point if somewhere in the future stakeholders want to change the course of the project. Meanwhile, the action register works as a simple reminder. The examples of the decision and action point registers used in the project can be seen in Table 10-1 and Table 10-2 respectively. The decision register contains information about the decisions, the date and occasion when the decisions were taken, and the people who agreed with the decisions. The action point register contains information about the actions, person responsible for executing the action, when the action was defined and the deadline, and the status of the action point.

Table 10-1 Decision Register for the project

No	Decision taken	Date	Occasion	Attendees
1	Not all use cases should be implemented. Only main uses cases matter.	8-Feb-19	Weekly Meeting	Trainee, BOK, MKHP
2	The second design alternative is agreed.	5-Apr-19	Weekly Meeting	Trainee, BOK, MKHP

Table 10-2 Action Point Register for the project

No	Action	Person responsible	Defined (date)	Deadline (date)	Status
1	Send meeting invitation for next monthly meeting	Trainee	08-Jan-19	09-Jan-19	Done
2	Propose dates and time for fixed future monthly meeting	Trainee	08-Jan-19	09-Jan-19	Done

10.2.4. Risk Management

Every project has risks. This section discusses risks that were encountered during the project. For each risk that was identified, levels of likelihood and impact were assigned. The risks that had high-medium, medium-high, or high-high pair of likelihood and impact were analyzed to determine the mitigation and contingency actions. The identified risks in this project can be seen in Table 10-3.

The risks were revisited every monthly meeting. If a new risk was identified, this risk then discussed to define the likelihood and impact, and subsequently, determine the mitigation and contingency actions. Besides the new risk, the old ones were also discussed to redefine the likelihood after executing the mitigation actions or redefine the impact in case the risk occurs and the contingency actions were executed. When a risk was no longer valid or, it was written from the scope.

Table 10-3 Risk Register of the project

No	Risk	Likelihood	Impact	Mitigation	Contingency
1	The solution biased by the current design	High	High	Plan regular reviews by domain experts and designers during design phase	Analyze the design and prepare design revision and review
2	Overlook non-functional requirements	Medium	High	Plan requirement set review with domain experts regularly	Address the non-functional requirement during the next iteration of the design
3	The design is impossible to be implemented	Medium	High	Plan regular reviews by domain experts and designers during design phase	Revise the design on the next design iteration
4	Miscommunication between stakeholders	Medium	High	Organize regular meeting with key stakeholders, and inform other stakeholders regularly	Organize a meeting to clear the misunderstanding between stakeholders
5	Does not engage people during design	Medium	High	Aware of people that need to be engaged and arrange meetings with them	Start arranging meetings with people who could help with the assignment
6	The domain knowledge information is limited	High	Medium	Plan to gather the domain knowledge early and identify domain experts for help	Create a domain model to validate the trainee's understanding of the

					domain with the domain experts.
7	The system environment change	Medium	High	Select a system version and use it as a benchmark and development point	

10.3 Information Management

This section presents how information was managed during the project. The discussion includes documentation management and communication management.

10.3.1. Documentation

In the lifetime of this project, two kinds of documents were managed: static and dynamic. Static documents are documents that will not be updated in the future, such as meeting minutes and presentations for progress meetings. Dynamic documents are documents that may be updated during the project time like the registers, design documents, and final reports. All documents were saved in a local shared folder, which was accessible by the stakeholders.

10.3.2. Communication

Keeping the stakeholders updated is important to make sure that everybody involved in the project is on the same page. Within ASML, weekly meetings were organized to present the progress of the project and discuss any impediment. In addition to these weekly meetings, monthly meetings were scheduled with the University Supervisor to discuss the monthly progress, the plan for the next month, and project evaluations. Besides, the project progress was sent to the University Supervisor every week.

11. Project Retrospective

This chapter discusses the experience that the author gained during the project. The first section explains about the challenges that the author faced throughout the entire project. The second section presents the lesson that the author learned at the end of the project.

11.1 Challenges

During the project, the author face challenges that he had to overcome. In these sections, two main challenges are presented. The first challenge is a common challenge that a PDEng Trainee usually encounters during a graduation project, such as understanding a new domain or learning a tool. The other one was a challenge unique to a project, such as reverse engineering legacy code.

11.1.1. Understanding the domain and learning the tools

ASML is a big company, which has its own body of knowledge. At the beginning of the project, the author was overwhelmed with the vast knowledge that ASML has. To fully understand the domain in which the project was conducted took approximately two months of project time.

In addition to understanding the domain, the author had to learn the tools that ASML uses to develop software. These tools are developed internally within ASML with limited documentation and examples compared to off-the-shelf tools. Furthermore, the learning curve of these tools are relatively steep for a person who uses them for the first time.

11.1.2. Reverse-engineering from a source code due to lack in documentation

In this project, the author faced a situation where the latest form of the design lives within the source code. In other words, the component that the author aimed to improve has an outdated formal documentation. To completely understand the current design of the software component, the author needed to extract the design from the source code. It appeared to be a challenge to the author as the codebase was quite large and hard to understand.

11.2 Lesson learned

Looking back at the project, the author learned some lessons that might be useful for people who have similar projects. These lessons were mostly from the process point of view.

11.2.1. Understand the domain as fast as possible

In general, understanding the domain is essential to grasp the real problem and come up with the real solution. In a project with a strict time limitation, understanding the domain early means that we have more time to deeply investigate the problem and explore alternative solutions. In this project, the author felt the benefit of understanding the domain from top to bottom. Knowing the big picture of the system helped the author to comprehend how the software components work together. Also, capturing the domain in a domain model helped to communicate and clarify the level of our domain knowledge. In addition to that, actively asking questions to the experts was ultimately helpful for understanding the domain.

11.2.2. Ask people around and do not wait

In the company, you are surrounded by the experts. However, do not expect that they will approach you and tell you what you need to know as they have their own job to do. Hence, you need to be proactive and ask as many questions as possible, especially at the beginning of the project. Also, new insights can be obtained by discussing your project with other people.

Glossary

OO	Object-oriented
EPS	Element Performance Specification
LoC	Line of Code
OOTI	Ontwerpersopleiding Technische Informatica
WP	Wafer Positioning
DoF	Degree of Freedom
IC	Integrated Circuit
DUV	Deep Ultra Violet
EUV	Extreme Ultra Violet
Cohesion	Degree of internal consistency within parts of a design
Coupling	Degree of interdependency between parts of a design
ADD	Attribute-Driven Design
ASR	Architecturally Significant Requirement
SOLID	Object-Oriented design principle by Robert C. Martin
SA	Synchronized Action
Devbench	A computer program that simulates ASML machines
Testbench	A test platform that uses machine simulation that involves real hardware

Bibliography

[1] "ASML's History," ASML, [Online]. Available: https://www.asml.com/en/company/about-asml/history. [Accessed 4 September 2019].

[2] C. van Antwerpen, "EPS WP Positioning Scanning Application," ASML, Veldhoven, 2011.

[3] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Element of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[4] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transaction on Software Engineering*, vol. 20, 1994.

[5] M. Badri and F. Toure, "Empirical Analysis of Object-Oriented Design Metrics for Predicting Unit Testing Effort of Classes," *Journal of Software Engineering and Applications*, vol. 5, pp. 513-526, 2012.

[6] P. Jansen, "The TIOBE Quality Indicator," 10 October 2018. [Online]. Available: https://www.tiobe.com/files/TIOBEQualityIndicator_v4_3.pdf. [Accessed 15 March 2019].

[7] T. J. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering*, Vols. SE-2, no. 4, pp. 308 - 320, 1976.

[8] B. Schoenmakers, "The Yellow Book," ASML, Veldhoven, 2017.

[9] S. D. Mascia, Project Psychology: Using Psychological Models and Techniques to Create a Successful Project, New York: Routledge, 2016.

[10] L. Bass, P. Clements and R. Kazman, Software Architecture in Practice 3rd Edition, Addison-Wesley Professional, 2013.

[11] R. C. Martin, "Design Principles and Design Patterns," 2000.

About the Authors

Wahyu Utomo received his bachelor's degree in Electrical Engineering from Bandung Institute of Technology (ITB), Indonesia, in 2014. Later, he continued his study in Embedded System master's program in Eindhoven University of Technology (TU/e). He did his master's thesis in IMEC-NL, which is one of the research companies located in the High Tech Campus (HTC) area Eindhoven. His thesis project was to design and implement an initiator election algorithm for a state-of-the-art communication protocol for wireless sensor network.