Tampere University

Anh Huy Bui

DEVELOPMENT PROCESS FOR SOFTWARE GENERATED FROM MATLAB

Master of Science
Faculty of Information Technology and Communication Sciences
Examiners: Asst. Prof. David Hästbacka
M.Sc.(Tech.) Olli Suominen
May 2022

ABSTRACT

Anh Huy Bui: Development process for software generated from MATLAB
Master of Science
Tampere University
Embedded Systems
May 2022

Nowadays, modern devices are becoming extremely smart by embedded with advanced software. However, defects are more likely to occur when the software complexity increases. As a result, there is a need for a proper development and testing procedure to ensure defect-free software. Multiple software development process models were studied to achieve that goal.

To fulfill the demand for developing new technology, MATLAB (Matrix Laboratory) has become a powerful tool widely used among developers. MATLAB provides a coding environment, interactive user interface, and a vast number of libraries in every field of science. However, a limitation is that the software developed using MATLAB cannot operate on general platforms. Fortunately, it is possible to generate C/C++ software from a given MATLAB code. The problem is ensuring the dependability and quality of generated software.

This thesis analyzes MATLAB characteristics in terms of developing software. Then a software development process model based on other research is proposed. The model studies all phases, from planning to coding and testing, emphasizing the difference in generating process to coding. Finally, the model is evaluated for effectiveness when developing software from MATLAB and the functionality and dependability of generated software are also under investigation. The conclusion of the thesis will justify if developing software by generating from MATLAB is a good approach in general.

Keywords: Software development process model, Software Quality Assurance, MATLAB, MATLAB Coder

PREFACE

The research work in this thesis is a part of a project from F4E (Fusion for Energy) cooperated with Tampere University of Technology.

Firstly, I would like to express my sincere appreciation for my supervisor, David Hästbacka, for all of his guidance. Mr. Hästbacka has been very supportive to me of not just the thesis work but the whole study process. I was overwhelmed but thankful for his generosity.

Secondly, I would like to thank my team member in the research project, Mr. Olli Suominen and professor Atanas Gotchev for giving me this chance to work on this research. It was the greatest opportunity given to me when I needed it the most. Moreover, Mr. Suominen was not only a co-worker, but he was a good friend that made my work at Tampere University much easier.

Finally, my gratitude is to F4E members Mr. Ruiz Morales Emilio, Ms. Pintos Cabrera Nicole Alejandra, and Mr. Kolokotronis Efstathios. They all were contributory throughout the project time. In addition, my knowledge has been enriched by working with excellent people.

This thesis would be given to my family as my greatest gift.

Tampere, 7th May 2022

Anh Huy Bui

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

API(s)	Application Program Interface(s)
CERT	Computer Emergency Response Team
GPU	Graphics Processing Unit
IEC	International Electrotechnical Commission
MATLAB	Matrix Laboratory
MISRA	Motor Industry Software Reliability Association
OWASP	Open Web Application Security Project
SDLC	Software Development Life Cycle
SQA	Software Quality Assurance
XP	Extreme programming - a software development process model

1. INTRODUCTION

1.1 Background

Over the past decade, there has been a dramatic increase in the number of new technologies, including AI, Cloud Computing, the Internet of Things, and Blockchains. It may sound unfamiliar, yet the innovation presents itself in everyday devices, among which a modern smartphone, a transport control system, and an autopilot feature on cars are prime examples. As the size and complexity of these systems increase, there may be more possibilities for severe problems to occur. Therefore, improving software dependability is considered a critical factor in the software development process.

The term *software* is extremely common in various fields of science. In simple words, most forms of software act as a middle man between the users and the devices, receiving commands from the users and giving orders to the devices for execution.

It is also important that the term *software quality* be clearly defined in the context of this thesis. Multiple definitions of *software quality* are found. In general, *software quality* can be understood as a measure of how well software is structured or designed. The measurement is based on sets of standards such as ISO 9001/9000-3, ISO/IEC 9126, and ISO 26262:2018, which define different rules to be applied in their corresponding science field. The act of evaluating the quality of software is called *Software Quality Assurance (SQA)*. The main purpose of the SQA process is to propose a procedure for assuring software dependability. Software dependability indicates how reliable, safe and durable software is. In other words, regardless of whether the software fulfills the user's expectations or not, it must not behave unpredictably or uncontrollably. It can be observed that ensuring users' safety in technologically innovative industries has always been considered a decisive factor.

Lacking a reliable procedure to assure software quality could result in severe consequences. For example, therac-25, a radiation therapy machine, accidentally killed four people and left two with lifelong injuries (from 1986 to 1987) [1]. Therac-25 was an upgraded design of Therac-6 and -20 when many manual controls were removed, leaving the computer to handle sensitive cases. The action that caused the machine to malfunction was when the user switched between *X-ray mode* and *Electron mode* within setting

up time (about 8 seconds). As a result, an x-ray beam was fired at maximum power. Although many tests were conducted, tragedies had already occurred before the software issue was identified. A more recent example is when Tesla recalled 158 000 cars over a defect that increased the risk of crashes. Several different reasons have been proposed for this problem. One cause was the failure of a media console unit, which resulted in the loss of rear-view camera images, i.e., a failure of the computer vision system. Fortunately, there has not been any reported accident caused by these errors.

Among diverse types of software, computer vision is one of the most popular technology. Computer vision is a scientific field in which computers are trained to identify collected samples of images, and they then process this information to recognize real-life objects. Some notable applications of this field are facial data being used as a smartphone security method, autonomous cars having the ability to sense their surroundings and operate with little to no human involvement, cameras in lecture theatres being capable of tracking and following the presenter's movements, and so on.

Within complex systems using computer vision, pre-programmed libraries are widely used to program the software to save time and effort. MATLAB is a common programming platform and an interactive environment widely used in computing science and visualization. One important feature of MATLAB is the built-in libraries, especially those used in computer vision. In comparison with OpenCV, another open-source library, the advantages of MATLAB include data presentation, user-friendly interfaces, the ability to optimize software on hardware (NVIDIA GPUs), and easiness in error debugging. [2]. MATLAB Coder is also a MATLAB feature that provides the ability to translate MATLAB code into different programming languages (C/C++), enabling it to be run on different platforms.

1.2 Problem Definition

As MATLAB is a powerful tool to help reduce programming effort, many applications are written using it. However, some devices can not execute that type of program. To design software in a general platform based on a given MATLAB program, the programmer should generate the required code rather than re-programming the software from scratch. In that case, generated code using MATLAB Coder is also considered a complete type of software itself. Even though the original MATLAB code may be fully tested, there is no proof of the correctness and behaviour of the code generated from it. There has been some research about software quality assurance, but the difficulties and advantages when applying to codes generated from MATLAB are currently unknown.

The software discussed in this thesis is a robot-world and hand-eye calibration. It is an application of computer vision which is already written in MATLAB programming language to make use of multiple CV APIs. After that, MATLAB Coder is used to translate this software into C++. Then a thorough software quality assurance procedure is required to

be applied to the translated program.

To conclude, this thesis attempts to answer the question: *How can one properly generate C++ software from MATLAB code and ensure its quality and dependability?*.

1.3 Methodology

1.3.1 Objectives

The main focus of this work is to propose a process of MATLAB code generation, design efficient testing phases to assure the software quality, and evaluate the effectiveness of MATLAB Coder in terms of effort saving and its output's performance. Besides, this thesis will introduce different standards of software quality, along with a few testing tools.

1.3.2 Work Description

In order to achieve the objectives, this thesis includes the following parts:

- Research on related study about software development processes.
- Introduction to MATLAB Coder's most important features for playing the center role in the software development process model proposed in this thesis.
- Explanation in details of design, implementation and testing phases.
- Evaluation of generated software and the process.

This document is structured as follows. Chapter 2 investigates the related research on the process of developing software. Before starting implementing a model of software development in chapter 4, chapter 3 explores the MATLAB Coder features. Chapter 5 analyzes the limitations and effectiveness of the software and the development process model.

2. RELATED STUDY ON THE SOFTWARE DEVELOPMENT PROCESS MODELS

2.1 Definition of *quality software*

In the modern world, people's lives are improved with many advanced technologies, and software can be considered the heart of innovation. There are multiple technical definitions of software. Strictly speaking, the term *software* can be defined as a set of instructions, commands, or programs that control the hardware or devices [3]. Applications in computers or smartphones are examples of this. The first design principle is the software's capability to complete the designed tasks. Nevertheless, with an increasing number of users' demands and non-stop innovation, big software companies such as Google, Apple, or Microsoft are developing what they call an *ecosystem*, in which all devices are connected and become a more complex system.

When the software systems become more complex, there is a higher probability of defects. User experience may be worsened by faulty video games or flight booking apps, but the flaws in human safety applications, such as an aircraft flight control system, are considered dangerous. In fact, there hardly be software with no defect, which may be called *quality software*. As books or novels can be found with errors made by the writers, software, a product of programmers, may have hidden defects. While a book may have hundreds of pages, a program with 1 million lines of code can fill up to 40000 pages [4]. It is impractical to trace for defects in each line of code. Nonetheless, a program with defects can still work. Because software does not get aged, it should always operate with what it is programmed to do. Therefore, software quality over time is not the problem to be solved.

The problem is software use cases. When there are multiple ways to operate a program, more mistakes are likely to be made during development. Thus, it is challenging for programmers to simulate and test all the cases. As an example, the accident of MV-22 Osprey killed four marines in 2001 [5]. Upon landing the aircraft, there was a failure in the hydraulic line. However, the problem did not go badly until the pilot pressed the reset button 8-10 times as instructed. The final investigation blamed the poor software test procedures.

It is a fact that software failures affect both end-users and many companies' reputations and brand value. For instance, in 2017, UK-based loan company Provident Financial was reported with 2.4 billion US dollars in financial loss [6]. According to IBIS World [7], the market size of Software Testing Services in the US has risen from 1.8 billion to 6.3 billion dollars over the last ten years (since 2011). It is proof that business people are aware of the importance of reliable software. Hence, in science, the concept *software quality assurance* or SQA is proposed.

2.2 Software quality assurance and different concepts

Software quality assurance was defined in much research. Agarwal [8] proposed SQA as an approach to the evaluation of software based on product standards, processes, and procedures. In "Software Quality Assurance: From theory to implementation" book, Galin [9] described SQA as "umbrella activities" which comprise design disciplines of quality assurance applied to all software development processes. Khazanchi and Sutton [10] gave the idea of "assurance services" to be a chain of activities performed by a trusted partner to validate business transactions with the objectives of lowering the risk and improve quality.

In summary, software quality assurance can be described by some points:

- SQA is a series of activities implemented in every stage of the software development life cycle.
- SQA is performed based on design requirements, agreement on software standards.
- SQA objective is to ensure software safety during operation, prevent unexpected behaviours and software failures.
- in terms of business, SQA should be integrated with the project cycle and suit the budgets.

There are other concepts that often be misconceived, such as quality control and software testing. Although there can be various explanations for these terms, quality control can generally be seen as a part of quality assurance. While quality assurance involves product design procedures that minimize potential risk, even before the product is manufactured, quality control is a process of detecting and handling issues or bugs after a product sample is manufactured. Software testing is the major activity of quality control, in which any potential defects should be identified and fixed. [11]

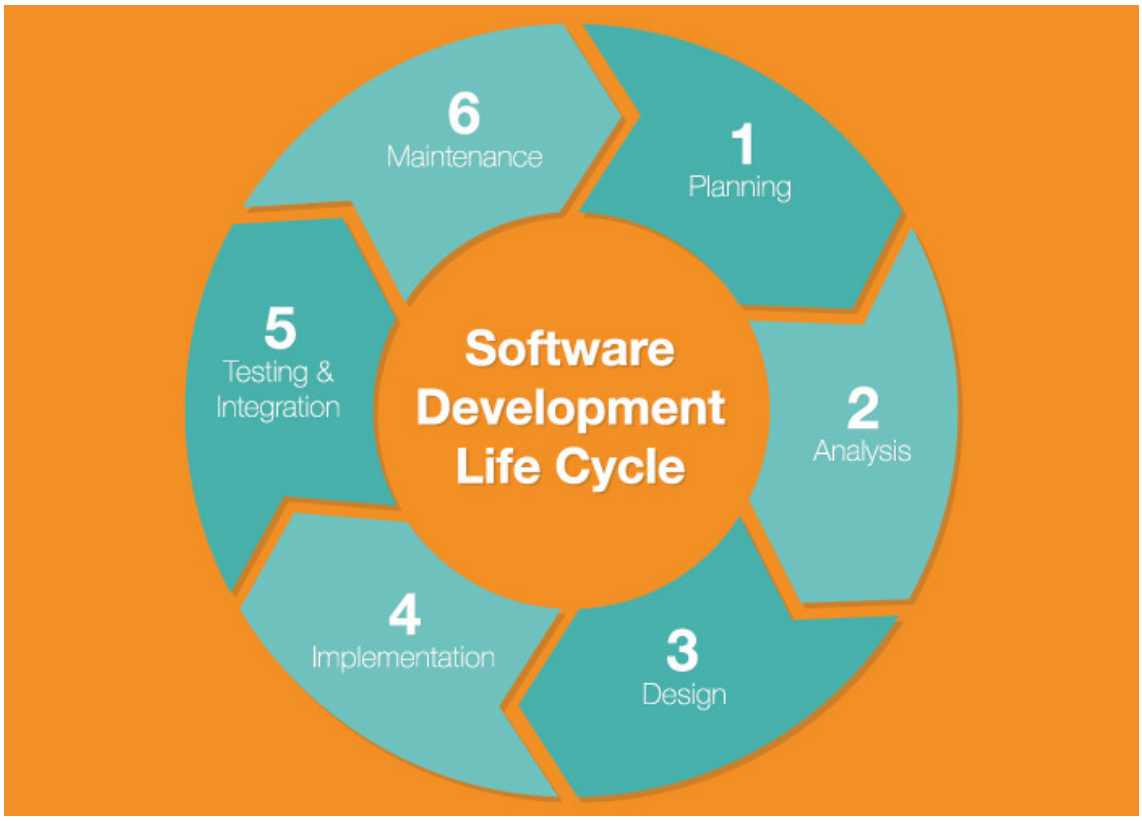

Figure 2.1. *Sofware development life cycle (SDLC). [13]*

2.3 The software development process

To go further, the implementation of SQA has its own life cycle. However, SQA can be integrated into the process of software development. This process, called the software development life cycle (SDLC), usually involves five stages: planning, designing, implementation, testing, deployment, and maintenance. [12]

2.3.1 Plan

Planning and analysis can either be separated or combined into the first stage of SDLC. In this stage, business partners agree on the ideas of the product, setting the cost, project objectives, and planning schedule. During this phase, the development team also plans the next activities or duties in the lifecycle. [14]

The function of SQA is to assess the quality processes and define what procedures to follow. In addition, within the software scope, SQA will specify general design standards, programming requirements, and rules. The activities are conducted by meeting and reviewing documents. [15]

It is worth mentioning that planning/analysis should be considered the most crucial SDLC phase. Some risks at this phase may be inadequate software requirements, incorrect

expense estimation, or misplanning project schedules. Unfortunately, not many of these issues are detectable until reaching the later phases. Consequently, a massive amount of money is lost, and the product fails to meet the deadlines.

2.3.2 Design

The design phase can also be called the prototyping phase. It is the stage of a project in which the development team determines the detailed design of the software, including system components, their functional responsibilities, and the connection between them [16]. In addition, some significant factors are decided: design tools, approach to problems and possible solutions, which platform that software will operate on, some security protocols and user interfaces. In some projects, an early version of software or prototype product can also be built.

The role of SQA is to evaluate the feasibility and efficiency of the method used in later stages and the quality of the prototype, whether it satisfies the requirements and standards defined in the planning phase. For example, the SQA team creates an assessment of different working designs to utilize a design plan for some criteria such as reliability, scalability, and maintainability. Subsequently, the effectiveness of the later testing phase is increased. [17]

2.3.3 Implementation

Implementation is one major phase of SDLC. Programming and coding are the main activities in which designs are translated into code [18]. In addition, other tasks such as reporting on code changes, design issues, and coding manual are also important. These documents will help the testers to trace potential defects if they exist. In case of any design issues or requirements are incapable of fulfilling, it may be necessary to start a discussion to prevent further problems.

It is noted that, unlike in previous stages, SQA may not play a separate role. Some SQA requirements done by the development team are following coding rules such as naming, organizing the code, and planning the unit test cases. The SQA team should evaluate the quality of the code and unit test plan to ensure all possible cases are covered.

2.3.4 Testing

Testing is the major activity of the SQA process. The priority of this phase is to ensure the software is developed to satisfy all design requirements. The second purpose is to test all possible cases, including ones not in the requirements. After this phase, software must not have any unpredicted behavior. [18]

There usually are different activities during testing: static testing, automated testing, regression testing, and performance testing. As testing is conducted, the test results and possible causes in case of failures should be documented and reviewed. It is crucial that any decision made after the review may affect the whole SDLC.

2.3.5 Deploy and maintenance

Deployment and maintenance is the last phase after the user-acceptance test is performed. The product should be ready for launch. However, when users detect possible errors, resolving the issues is one crucial duty. In addition, to improve the quality of the product, an update or a future release should also be considered. [18]

2.4 Software development model

Now that the five phases of SDLC are defined, the next step is to analyze the process order. Different types of models determine the sequence of the stages called software process model [19]. The five most popular software process models are waterfall, V-model, incremental/iterative, extreme programming, and spiral. These models show different benefits and shortcomings. Therefore, each of them is suitable for different kinds of projects.

2.4.1 Waterfall model

The waterfall model was first introduced in 1970 by Winston Royce [20]. Although waterfall has slightly changed since then, the structure of this process model is still similar to its name, with all production phases ordered in a linear sequence. As can be seen in figure 2.2, the arrow describes the flow of this process model. The arrows are one-way in figure 2.2. Therefore, a waterfall-model project, all the preceding phases must be completed before going to the later phases. In other words, information related to software requirements, designs, and testing methods must be agreed upon and documented during the first phases of the project.

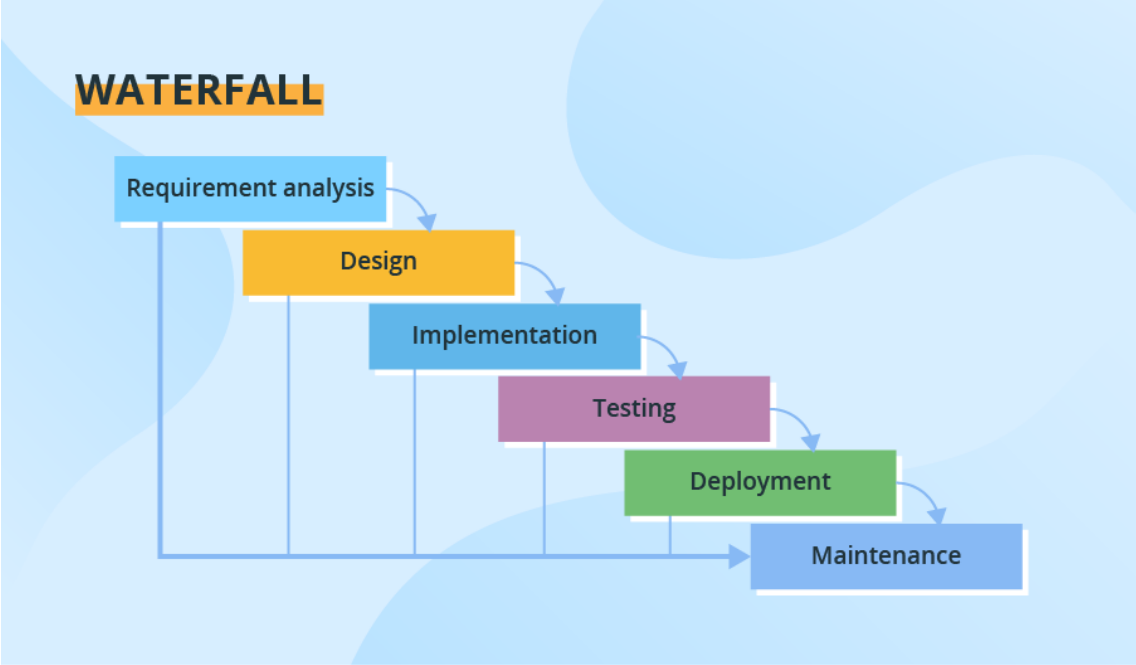

Figure 2.2. *Waterfall Model. [21]*

It can be agreed that the waterfall model is one of the simplest to implement. Therefore, it is suitable for small projects or projects that strictly follow some specific standards, budget, and schedule [19]. Nevertheless, the benefit of simplicity has a trade-off. Seemingly, this model cannot adapt to changes easily. It is incapable of foreseeing the end product until it reaches the deployment phase. For example, when the SQA team finds some defects at the testing phase, it could be needed to trace back to the implementation, design, or even planning phase to find the solution. Thus, in some charts, the arrows can be drawn in two directions.

2.4.2 V-model

V-model, also known as the verification and validation model, was created by Kevin Forsberg and Harold Mooz in 1991 following NASA's "Vee" chart [22]. First glance at figure 2.3 reveals that V-model is also a sequential process model, starting from the left branch of the *V* letter to the right. Specifically, the left describes earlier phases concerning analysis/planning and designing, while the testing phases are shown on the right branch. The significant difference between the waterfall and V-model is the testing phases. That is, for every design phase is planned on the left branch, there is a corresponding testing phase for that design on the right of the *V* shape. It means that it test procedures must be ready along with the implementation of design phases. An extension of the V-model is the W-model, proposed by Kung-Kiu Lau [23], which consists of two V-models for component development and system development.

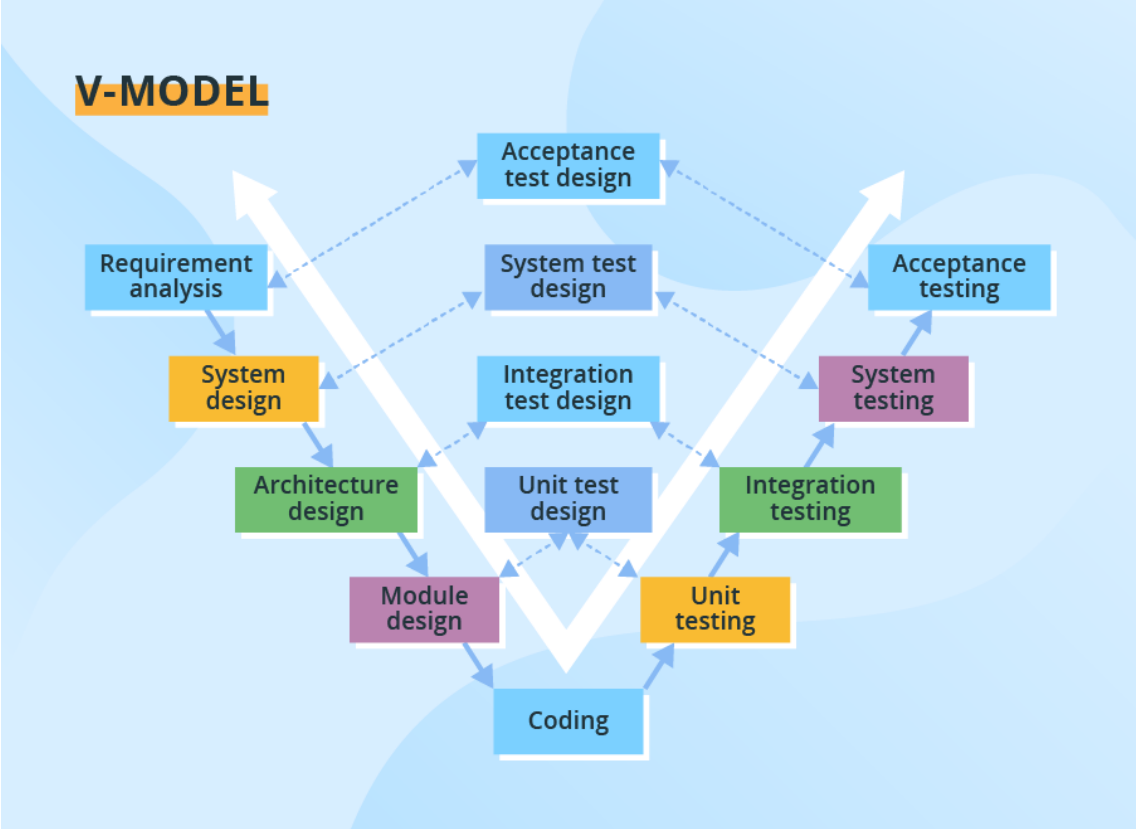

Figure 2.3. *V-Model. [21]*

With all characteristics mentioned, V-model is widely used in large companies. Because emphasizing testing, it provides remarkable quality control processes. However, although defects in requirements or design can be detected in the early phases of the project, the engineers must have in-depth knowledge and experience working with this type of model. As a result, despite giving very effective processes, the V shape is also the most expensive and time-consuming model.

2.4.3 Incremental and Iterative models

Incremental and Iterative models (IID) are said to appear even before 1970, although the name was not officially used. Some research believes that in the article "Managing the Development of Large Software Systems" (1970), the idea stated by Winston Royce was, in fact, "Incremental and Iterative models", not the waterfall. Also, in the 1970s, Harlan Mills promoted the iterative process model [24]. The history of the development of the models is a long story.

The IID and waterfall models are usually compared as they share the same principle of a linear sequence of processes. In other words, IID can be considered an extended version of the waterfall model. Looking at figure 2.4, in the incremental model, after all the requirements are examined in the planning/analysis phase, the software design is divided

into small parts or modules. These modules are then handled separately, independently, and possibly simultaneously. [25]

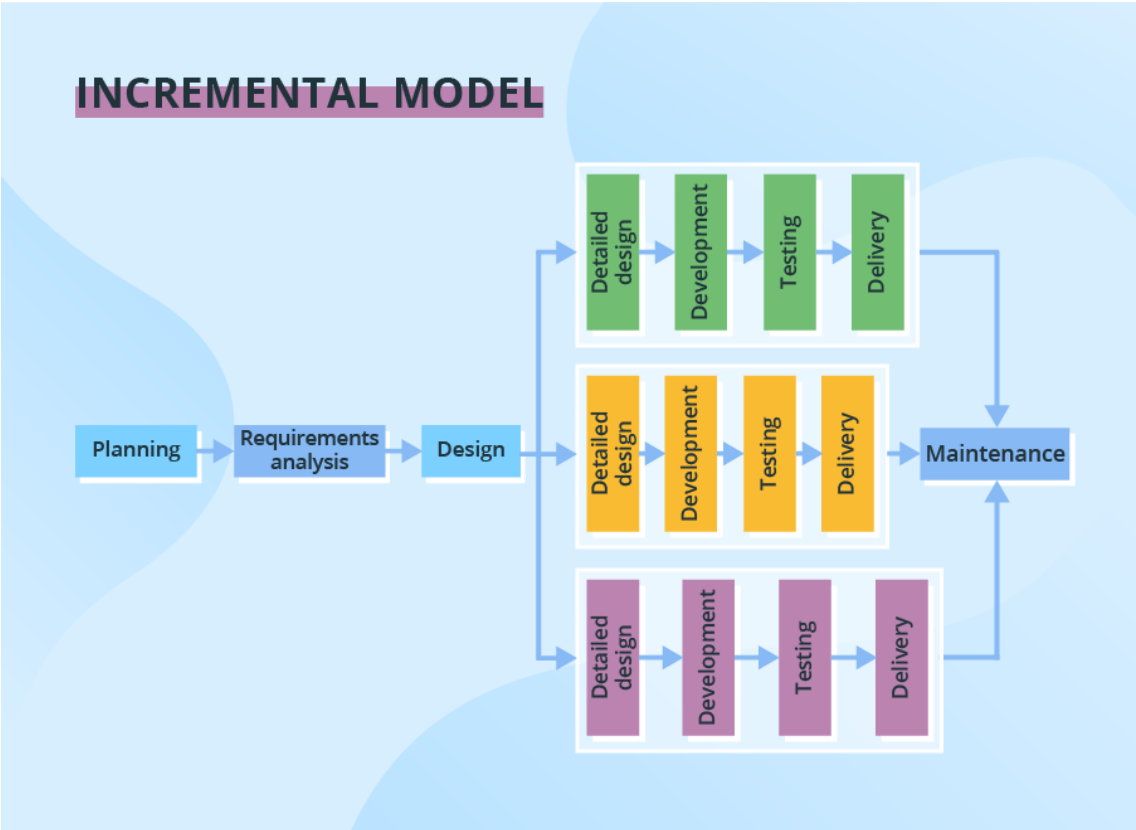

Figure 2.4. *Incremental model. [21]*

The iterative model, as shown in figure 2.5, shares many common characteristics with the incremental model. The difference is that software is developed in cycles in the iterative model, and new features and additions can be implemented in each iteration. Thus, software developed with this model is very flexible with changes. [21]

The IID has an advantage in quickly adapting to new software requirements or features, and testing and fixing bugs in smaller parts of the software. As a result, it is suitable for time-constrained projects or frequently updated software. To successfully apply this model, however, good project management skills are required.

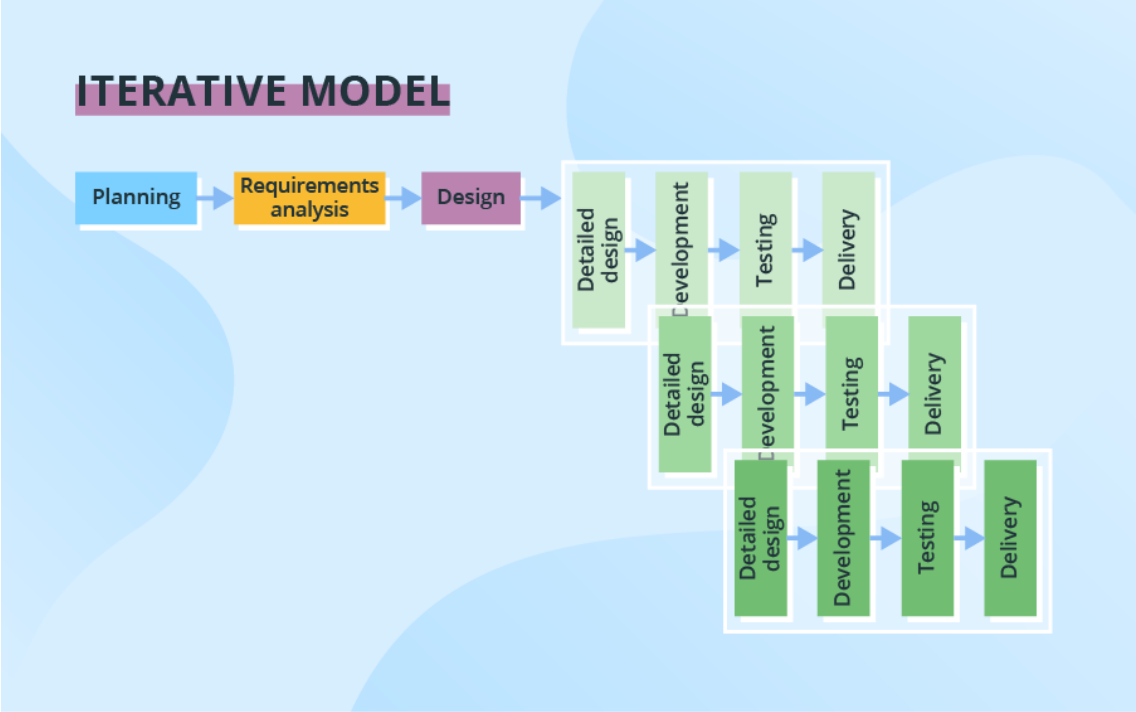

Figure 2.5. *Iterative model. [21]*

2.4.4 Spiral model

The spiral model was first described by Barry W. Boehm in 1986, although his thesis was not published until 1988. In that article, he claimed that the significant feature of the spiral model, in comparison with other types of models, was "that it creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process" [26]. Since then, it has evolved through the years based on the refinement of the waterfall model. In 2014, the originator published an enhanced version of the spiral model providing guidelines and principles for using the model. [27]

Seemingly, from figure 2.6, the spiral model is easier to be defined when it is put in correlation with the incremental or iterative model. The main activities, including plan, design, implementation and test, or identification, design, construction, and evaluation in some other research, are conducted in cycles. However, as described by Barry W. Boehm, the spiral model concentrates on risk management. Specifically, during phase 2 or the design phase, the software design is built, but the engineers are also required to analyze all possible uncertainty areas and alternative approaches. This step may include building and evaluating prototypes. After that, there is no changes or addition during the development phase. The next cycle then narrows down the risk, and new features can be added.

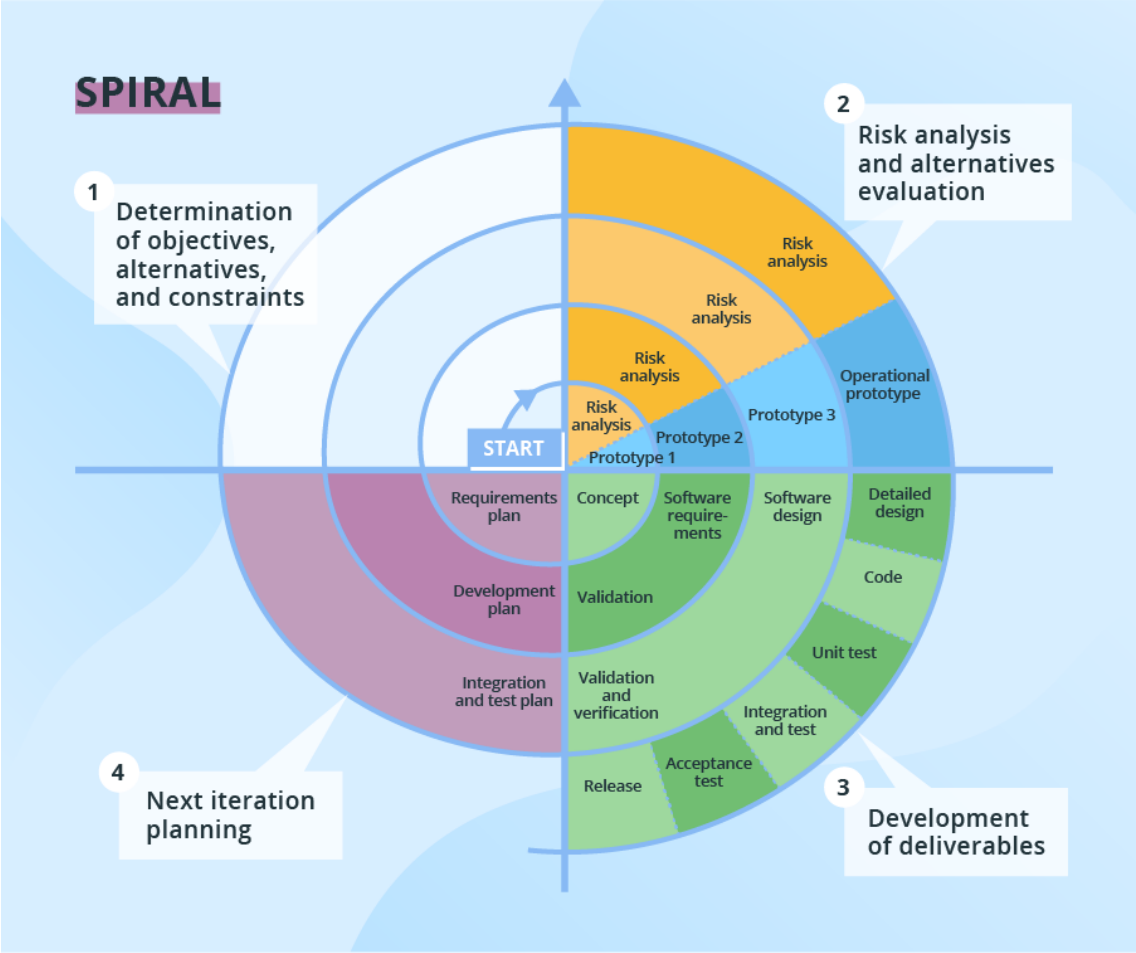

Figure 2.6. *Spiral model. [21]*

The spiral type may outweigh other software process models in some aspects. Firstly, it provides very low-risk processes as requirements are well analyzed at the beginning of the project. Secondly, it also has the ability to change the requirements in later software cycles. Thirdly, while the design can be divided into smaller parts, it helps developers with risk management. Yet, project management using the spiral model can be challenging as it can lead to never-ending loops [19]. Therefore, the spiral model is suitable for the cases such as when customers are uncertain about software requirements or projects with long-term commitment as there are possible changes in requirement.

2.4.5 Extreme programming

Of many software process models, extreme programming (XP) is unique. It was created by Ken Beck in the 1990s. He described it as "a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software" [28]. XP strongly emphasizes the process of coding, programming, and testing software. The word *extreme* in the name means pushing every activity to an extreme level.

The feature that makes XP unique is that XP does not focus on the order of the SDLC. On the other hand, it analyzes the design and implementation phases and simplifies them into four basic activities: coding, testing, listening, and designing. It means that along with designing, coding, and testing the software, developers need to listen and understand feedback and desires from customers. These activities are conducted in loops until the software is completed. Furthermore, XP is a discipline of software development [28] with 28 rules defined for engineers to comply with during the process. These rules include minor releases (frequently released an updated version of the software), coding standards (coding style and format), simple design (design must not be complicated and implemented as robust as possible), etc.

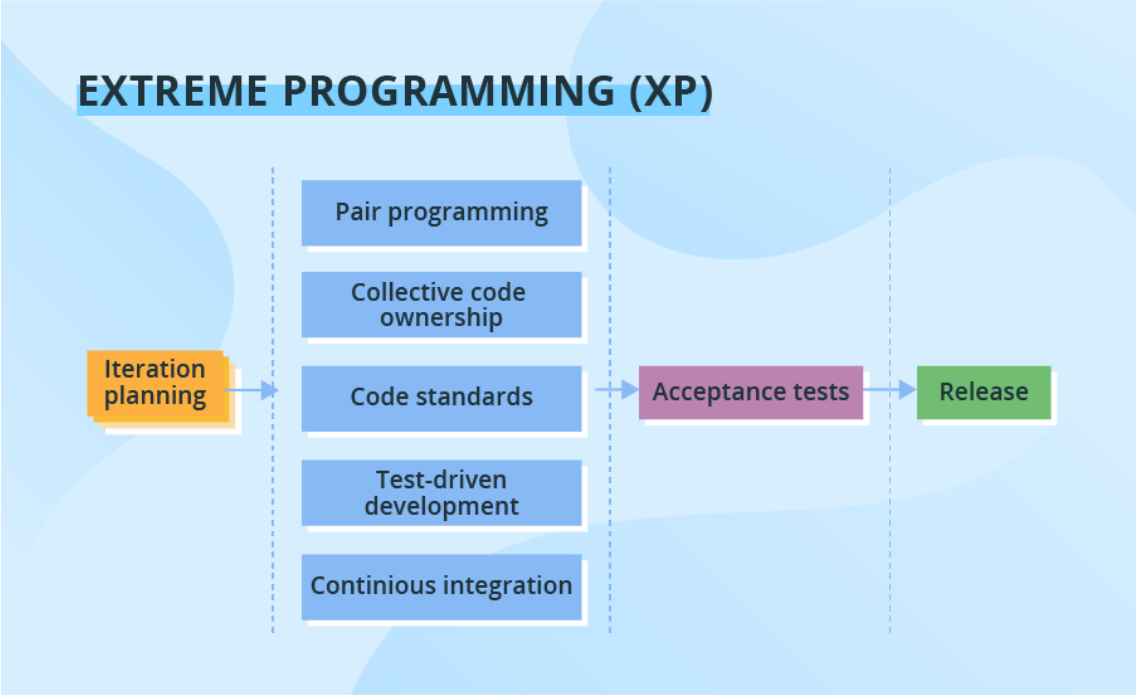

Figure 2.7. *Extreme programming. [21]*

XP is an efficient and reliable software development process. The continual cycle of developing, testing, and receiving feedback gives the software the flexibility in changes and updates. Furthermore, the defined rules guarantee reliable software with various types of tests. Nonetheless, XP has difficulty planning the schedule or budget. It is also not suitable to be implemented in a large team. Besides, [29] criticizes XP's weakness for being lack of documentation, requirements, and putting the stress on developers.

2.5 Software testing

While the implementation can vary between projects, there are some general principles in software testing. The first objective of software testing is to ensure software operates as expected or stated in the requirements. The next crucial target can be to ensure a defect-

free software that is capable of operating in undesirable situations without any unexpected behavior [30].

For testing technique, *white-box testing* and *black-box testing* was well explained with advantages and disadvantages in [31]. An additional technique, namely *grey-box testing*, together with the two was studied in [32]. The three testing technique were again discussed to improve for a better quality assurance in [33] by Jamil. These common techniques are then can be classified as *static testing* and *dynamic testing*.

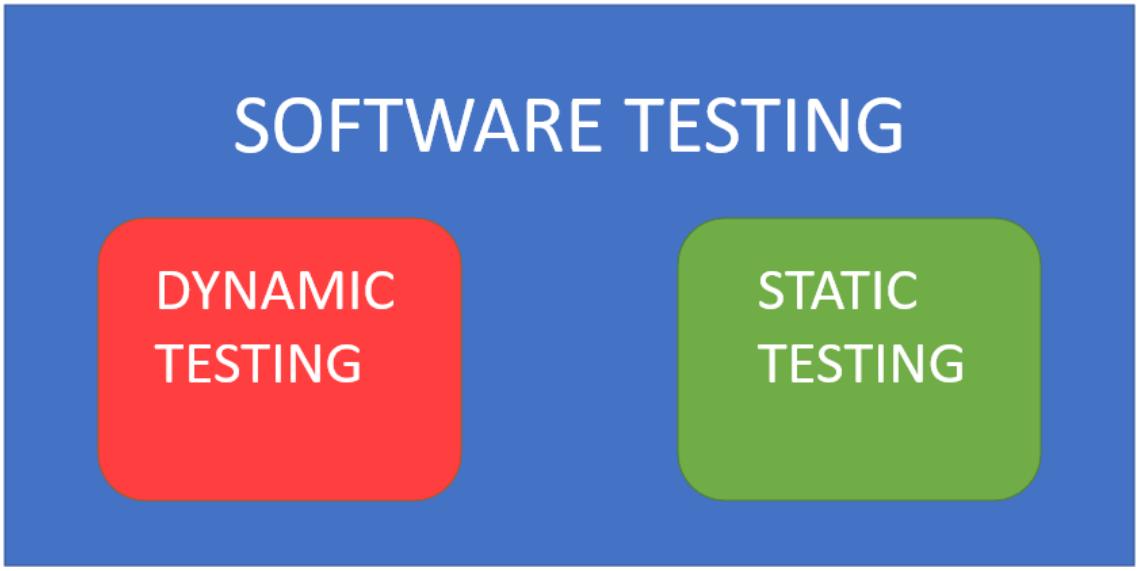

Figure 2.8. *Software Testing types.*

2.5.1 Software static testing

Static testing is a *white-box testing* that performs an inspection on the program code without executing the software. Static testing checks coding syntax, optimization, etc. to prevent defects even before the software is operated. [34] In addition, static testing can be used to ensure software against some coding standards, as below examples.

Table 2.1. *Examples of industrial coding standards*

Standards	Overview
MISRA C/C++	Motor Industry Software Reliability Association (MISRA) - a set of coding rules applied to C and C++ languages, widely used in embedded industries to ensure their safety, reliability and secureness.
CERT C/C++	Coding standards for C and C++ languages that are used to prevent security vulnerabilities in embedded system.
IEC 61508	Functional safety standards that depict requirements to the process of designing, implementing and operating of a safety related system.
ISO 26262	An adaptation of IEC 61508 for automotive industry - standards applied to functional safety in electronic/electrical systems on vehicles.
OWASP Top Ten	An standards of Open Web Application Security Project with information on security risks on web application.

These standards are essential as they may determine the implementation method of the design model for the target system. Firstly, there are explicit requirements of a safety system, which affect the process of developing and testing the system [35] [36]. Secondly, the coding standards define some particular rules or principles of how developers should write their code, which ensure the reusability of the code [37]. These rules are also divided into classifications. These classifications indicate that not all of the rules must be followed, but some can be ignored in exceptional cases.

Below is an example from MISRA C++ 2008 to give clear idea about coding guidelines.

Table 2.2. *Example of MISRA C++ 2008. [36]*

Rule	Classification	Rationale	Equivalence
MISRA C++ Rule 2-7-1	Required	C++ does not support the nesting of C-style comments even though some compilers support this as a non-portable language extension. A comment beginning with /* continues until the first */ is encountered. Any /* occurring inside a comment is a violation of this rule.[36]	CERT C Rule MSC04-C

Rule 2-7-1 from MISRA C++:2008 puts a restriction on the commenting part of C++ code, which uses /* and */ as its opening and closing that it cannot contain any /* symbol in between. Although the issue described does not concern the functionality of the software, and it may not even be detected as an error during the development process, the rule is classified as *Required* so to be fully compliant with MISRA C++, the coding of the software must meet this requirement. The rule is equivalent to rule MSC04-C from CERT C. Therefore, applying many coding standards to one project is not necessary.

It is apparent that the number of rules in these standards is overwhelming for programmers to perform manual checks. Therefore, it is needed some testing tools for this purpose. Some of the example tools that can be considered are in the below table.

Table 2.3. *Some tools for software static tests.*

Tools	Features
SonarQube	A powerful static code analysis tool that catches bugs and vulnerabilities with up to 29 programming languages supported, such as coverage features of 10 common web application security risks for Java, Javascript, and Python, MISRA C++ 2008 for C++.[38]
Helix QAC	Helix QAC is advertised as the best static code analyzer for Functional Safety and Standards Compliance.It is designed to focus on C and C++ with many supported standards including MISRA, AUTOSAR, IEC 61508, ISO 26262, EN 50128, IEC 60880, and IEC 62304.[39]
PC-lint	A C/C++ static analysis solution from Gimpel Software with the strength of simplicity, yet still has the ability to identify defects, potential bugs and supports necessary industrial safety standards. [40]
Parasoft C/C++ Testing	Parasoft C/C++ Testing is described as "a unified, fully integrated testing solution". It provides not only static code analysis, but also automated software testing capabilities. Besides, it can be integrated into CI/CD pipeline to speed up the software development process.[41]

2.5.2 Software dynamic testing

Software dynamic testing aims to identify software behaviors and trace bugs or defects in all possible cases. Therefore, unlike static code tests, dynamic tests must be performed on software in its operating state [30]. The targets of this type of testing must be described in software design requirements.

Dynamic testing can be divided into different sub-types depending on each project. However, in general, they can be seen in the graph below.

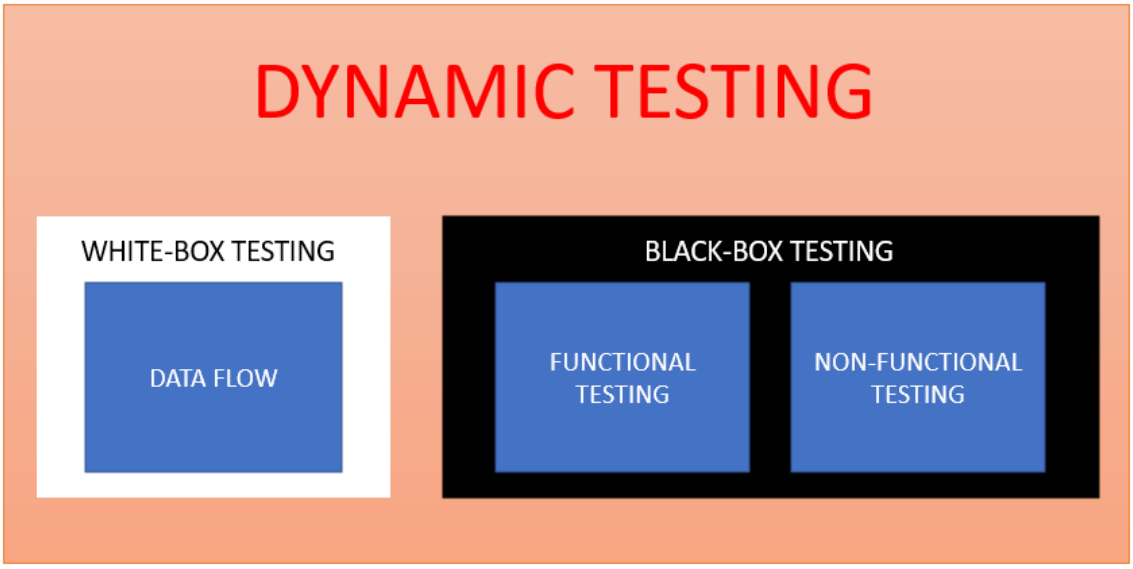

Figure 2.9. *Dynamic Testing types.*

The two types mentioned are white-box testing and black-box testing. White-box testing is the type of testing when programmers have insight into software architecture or code. These test results are an evaluation of the code in terms of complexity and coverage quality. Therefore, it is considered low-level testing. Black-box testing, on the other hand, is a higher level of testing as the program's internal structure is hidden, so testing can be performed from the users' perspective.

Though the testing types are acknowledged, to implement a systematic test suite, it is needed to detail what each test suite does.

- BLACK BOX TESTING

  Black-box testing can be classified into two different types: functional testing and non-functional testing [30]. As the name implies, functional testing validates all implemented features if they are well and correctly developed. It also ensures that all defined requirements are met and that the product is free of defects. [42] Thus, functional testing is the most important testing type. There are also different levels of functional testing.

Table 2.4. *Functional testing levels. [43]*

Level	Definition
Unit testing	Test cases are executed on the smallest testable units of the code, usually library functions. These units are independent from one another. Therefore, testing is performed in an isolated environment.
Integration testing	Test cases are executed on the integration of some units of the code or a module. These modules may not be independent . Thus, there can be involvement of other modules during testing.
System testing	Testing is performed when a complete system is ready. This level may or may not be required depending on project size.
User-Acceptance Testing	This testing level treats the testing targets as complete products under the user's perspective to verify if it is ready for business. Similar to system testing, this level may not be required in a small project.

Non-functional testing focuses on other aspects of the software: software run time/ performance and how many resources it costs when operating in different conditions. These features do not play such crucial roles as software functionality, yet, it has an impact on the user's experience [42]. The choice of non-functional test types varies between projects. Some of them are listed in the below table.

Table 2.5. *Non-functional testing types.[43]*

Level	Definition
Performance testing	Measurement of program run-time or response time to ensure it meets the desired requirements. Furthermore, this type of testing can check how many resources the program uses during its operating states.
Compatibility testing	Testing type ensures software behaviors when running in different platforms or environments.
Security testing	Testing checks whether parts of the software are user-accessible or protected as described in the requirements.

- WHITE BOX TESTING

  Regardings white-box testing, there usually is a type called *code coverage testing* or *data flow testing*. This type of test assesses the code quality in terms of how many code lines are executed during software operations or if there is an unreachable branch, data path, or control path in the code [30]. Moreover, it can help reduce the software size as it points out which parts of the code or libraries are not necessary for the current release then the programmer can leave them out. Although it will not affect the program's functionality, it may help improve the maintainability and scalability of the software in the future as programmers may want to update or upgrade their software.

  To sum up, the software testing model can be organized as in below graph.

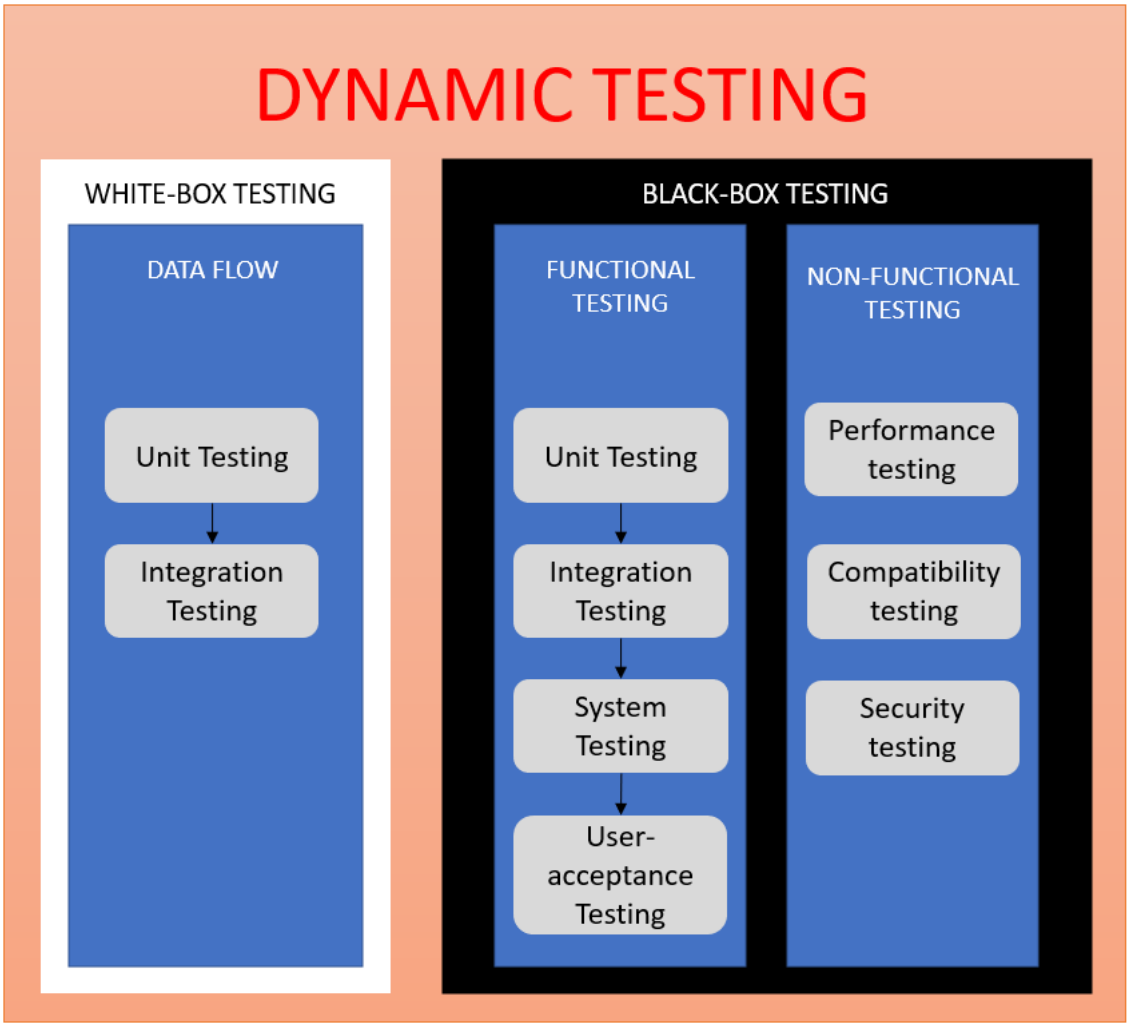

Figure 2.10. *Dynamic Testing in detail.*

2.5.3 Common testing tools

While programmers can do the tests by themselves, it is less time-consuming to use a third-party testing framework to perform the automated tests.

One of the most common tools to be named is Microsoft Unit Testing Framework for C/C++. This tool is an add-on of a well-known Microsoft IDE, Visual Studio. It has the advantages of performing various types of testing, including "behavior of the code in response to standard, boundary, and incorrect cases of input data"[44]. It also allows the generation of test cases even before the code is written. However, being a part of Visual Studio is a limitation for any programs not using MSVC compiler or designed to run on Windows environment.

The second C++ testing framework to be introduced is GoogleTest. It is developed by the Google Testing Technology team. It supports all three common OS, including Windows, Linux, Mac, and compilers such as GCC, MSVC, and clang. Regardings features, GoogleTest can perform all kinds of C++ software tests with emphasis on independency, reusability and organizability. The test results are informative and help with debugging the errors. It is currently used in multiple open-source projects [45].

Other testing tools are provided by either large companies or small groups of developers. Some worth mentioning are CppUTest, UnitTest++, QtTest, etc. These tools are also potent and featureful. While there are very lightweight and packed ones, the heavier others provide extra abilities. The choice of testing tools may depend on the size of the project.

Figure 2.11. *Some of testing frameworks.*

2.6 Model based testing

Model-based testing or model-driven testing is not a testing type but a testing technique that answers the question: *what the software is tested against?*. As investigated above, a software test suite is often generated or built based on agreement and explicitly stated in software requirements in the planning phase. Model-based testing is introduced as a testing method that constructs the model for testing and does not purely depend on requirements [46]. MATLAB [47] also suggests 3-step model-based testing, including creating the model, generating test cases, and validating the design with generated test cases.

In this thesis, with the assumption that the model is given, the two later steps will be analyzed. Like all software development models, requirements play the most critical role. In this type of testing, requirements are not only a pre-defined agreement but also come from the model's characteristics. These characteristics can be state transition, calculation results, and error handling mechanisms.

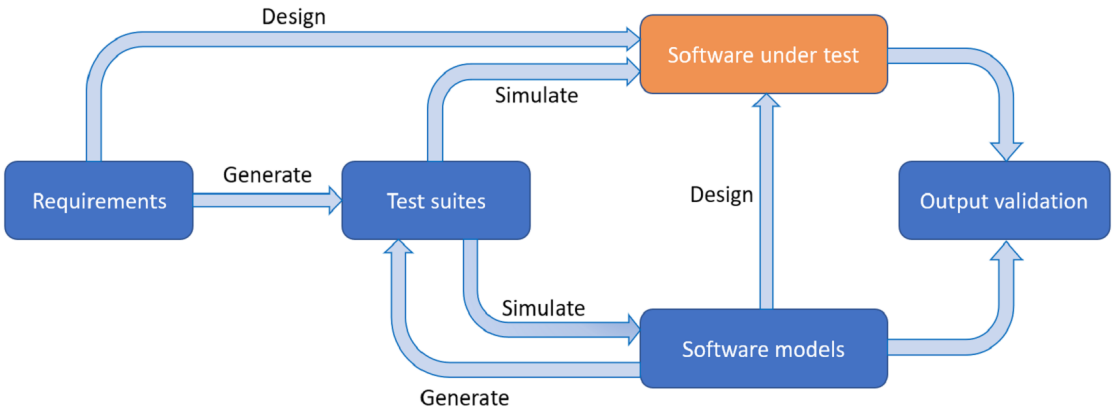

Figure 2.12. *Model based testing (figure based on [47]).*

In figure 2.12, model-driven testing is described with test suites generated by both requirements and the models. Then they are used for simulation of both the models and the designed software and whether their behaviors match is the result of this testing activity.

Commonly, two types of model-based testing are online and offline. They describe when or how the test suites are generated and executed. In the online testing, also known as on-the-fly testing, test suites are generated during test execution. It means that the designed software and models are operated simultaneously, and comparisons are set between their outputs or behaviors. On the other hand, offline testing is simpler with test suites generated by separately putting the model in operation; then the test is executed on the target. [48]

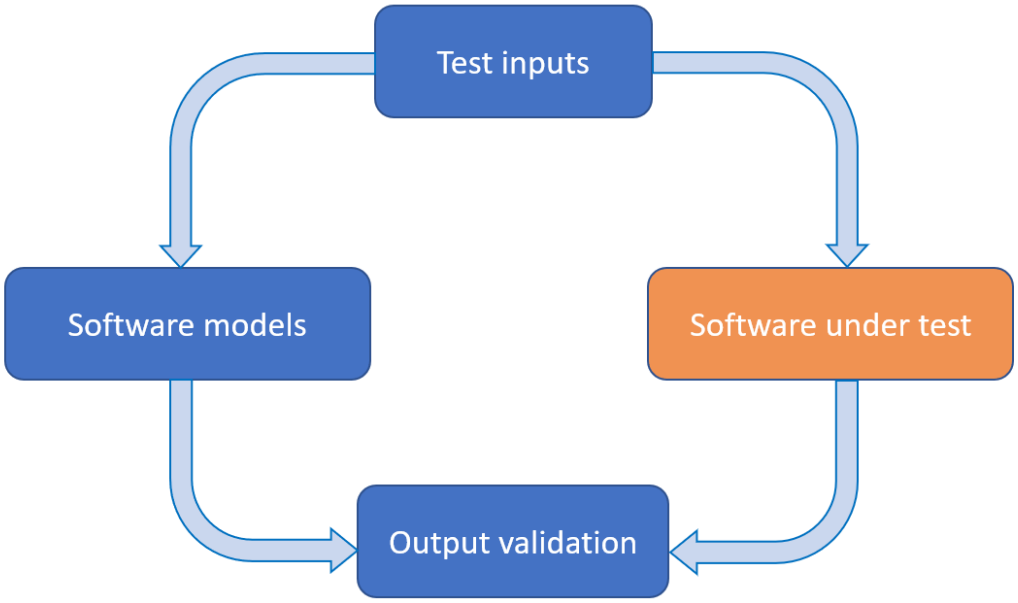

Figure 2.13. *Online model based testing.*

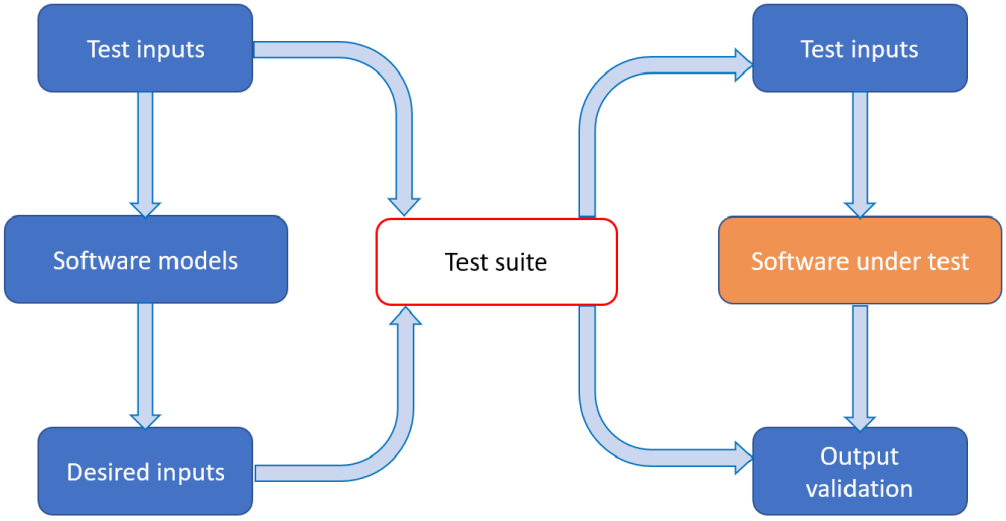

Figure 2.14. *Offline model based testing.*

It can be deduced that the online testing model should be more automated than the offline one as there is no need to create a test suite, but only the inputs of the tests are required. On the other hand, the offline testing model may require manual work to collect the test data. Nevertheless, it has the advantage that it can overcome the most significant limitation of the online testing model, that is, the requirement of designed software and its models can operate in the same platform or environment.

2.7 Quality assurance in Computer Vision

It is essential that computer-vision-based technology, considered a type of software, requires a process to ensure its quality when applied to safety-critical systems. Although computer vision (CV) technology is used in many scientific fields, there is little research about computer vision testing. Alternatively, in this field of CV, it is more common to use the term *performance evaluation* or *bench-marking* rather than *testing*. The reason for that is the design of CV software and the difference between it and other software types.

The two most crucial elements that make up a computer vision software are algorithm and sample of images called data-set acted as algorithm input. The most common testing method that is currently performed on computer-vision-based systems is to carry out the tests on pre-collected samples [49]. To be specific, verification and validation are two main steps of a typical quality assurance process applied in this field of science. Verification ensures no presence of software malfunction and unexpected behaviors. Validation performs a comparison of practical software output against desired ones. It can be predictable that injecting specific data-set as inputs into software during the validation process may give biased results. Therefore, although a CV software may perform well during tests, it might not function effectively with other sets of images. This means that there is a slight chance that validation does not reflect the true correctness of computer-vision-based software.

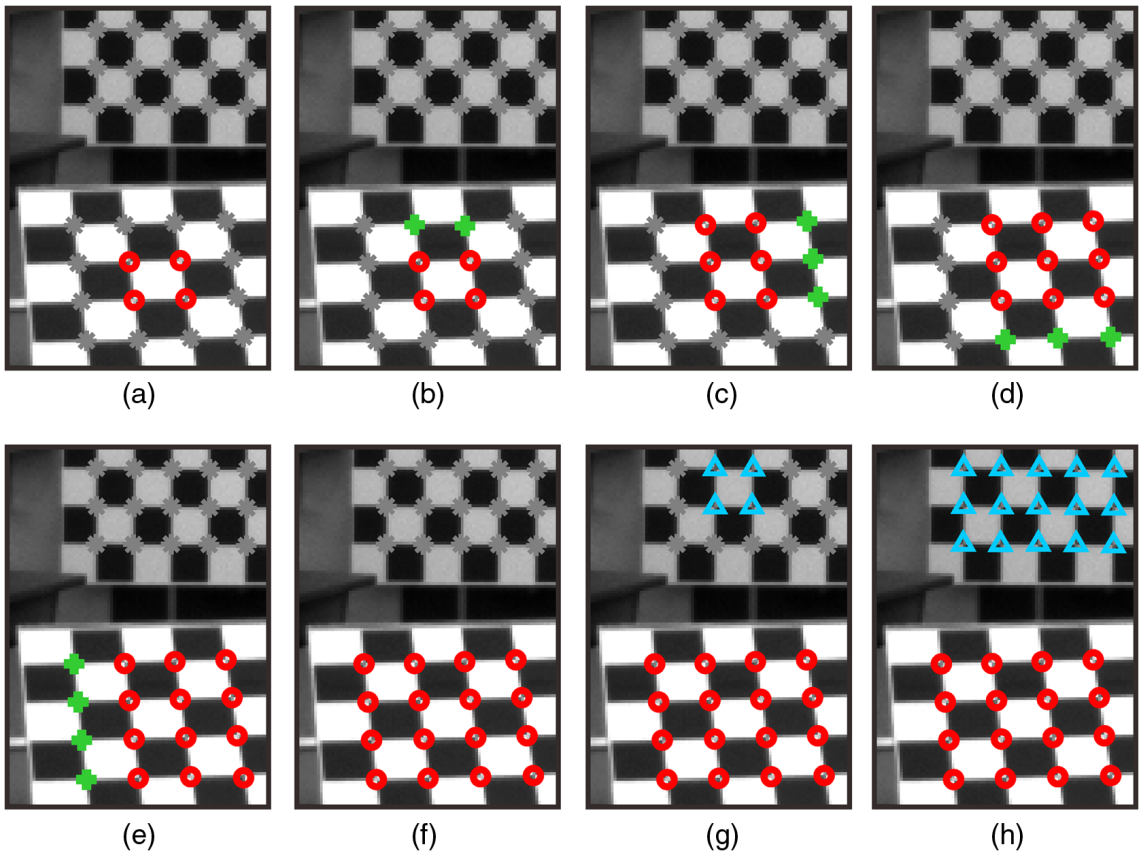

Figure 2.15. *An example of computer vision application with different input parameters.[50]*

The above photos are an example to describe the CV software behavior. The program is designed to give information about the pixels where there is a corner of the checkerboard. It is the fact that there is nothing called correct pixels to be expected, yet, any pixels near the corner can be accepted. Sometimes, one corner gets detected multiple times, but there are others that the program does not recognize. Now they are only two images. The results can vary significantly for the images that may be noisy, in low contrast, or get distorted. However, it can not be concluded that the program is wrongly designed or coded. On the other hand, it can be said that the program's performance is not good. Thus, the output of some CV software is greatly dependent on the input data and parameters.

Regarding testing, it is difficult to ensure if the CV software gives correct results in particular cases. While in ordinary software, programmers or users can verify the result with a reference number, for example, the result of 1 + 1 is always 2, in CV software, as in the above example, there is no factual data to test it against or it can pass this test but fail the others. A CV programmer may need to perform the test at the system level in those specific cases. The software and hardware are integrated and operate as a complete system and evaluate the software performance manually.

With all that said, trying to find a more effective computer vision test suite to overcome the difficulty is not within the scope of this thesis. Instead, the limitation is acknowledged and considered during this study.

3. MATLAB AND MATLAB CODER REQUIREMENTS FOR USAGE

MATLAB is the center of the development process discussed in this thesis, so the features of MATLAB coding language should be analyzed. In addition, there are some specific requirements of the MATLAB code so it can be generated into C/C++ using MATLAB Coder.

3.1 MATLAB

MATLAB, an abbreviation for Matrix Laboratory, was first written by Cleve Moler in the early 1980s. That version was built on the mathematics research papers by J. H. Wilkinson and 18 of his colleagues [51]. As the name implies, MATLAB's only function was a matrix calculator. Nevertheless, matrices are the key to many scientific fields. Over a few decades, MATLAB has now evolved into a powerful development tool. It has been integrated with a variety of scientific libraries and toolboxes.

Computer Vision and Image Processing are the two important toolboxes used in this thesis. It provides a basic format for images. For example, an RGB image is represented by a three-dimensional matrix of h x w x 3, with h as the height, w as the image's width while *3* for R, G, and B color channels. In addition to many well-made algorithms implemented, which involve color manipulation, and geometric transformation, users also can develop own functions with the help of mathematical tools from MATLAB. As good as it is, MATLAB still has some drawbacks. Firstly, its performance is not as good as traditional C programming language [52]. Secondly, it cannot be used on some devices with no capability of installing MATLAB. It is better to have a traditional C program than a MATLAB program in those cases. Therefore, MATLAB Coder is brought into play with the ability to translate MATLAB programming language into C/C++.

3.2 MATLAB programming language in comparison with C/C++

Before using MATLAB Coder to generate C code, it is important to understand what MATLAB language is and how it differs from C.

The first and most crucial feature of MATLAB language is that being a scripting language. A scripting language can be described as a more dynamic programming language. It does not compile the whole program; instead, it can execute every command as long as it is meaningful. Compared to C, which may detect logical errors during compilation, MATLAB can always execute the program and only stop when it runs into an error. MATLAB language, therefore, is more dynamic in terms of interacting with different users' input with a wide variety of power tools.

One key feature of a programming language is its data types. Fortunately, regarding the numeric types, MATLAB provides very similar types to C, which simplify their usage. It is noted that MATLAB also constructs single-precision and double-precision floating-point according to IEEE® Standard 754, in which a float number has 32 bits with 1 bit for sign, 8 for exponent, 23 for fraction, and a double, 64-bit number, has 1 for the sign, 11 for the exponent and 52 for the fraction.

Table 3.1. *MATLAB primitive types in comparison to C/C++ [53].*

MATLAB data types	Equivalent C data types	Storage size
int8	char, byte	1 byte
uint8	unsigned char, byte	1 byte unsigned
int16	short	2 bytes signed integer
uint16	unsigned short	2 bytes unsigned integer
int32	int	4 bytes signed integer
int32, long	long (Windows®)	4 bytes signed integer
int64, long	long (Linux®)	8 bytes signed integer
uint32	unsigned int	4 bytes unsigned integer
uint32, long	unsigned long (Windows)	4 bytes unsigned integer
uint64, long	unsigned long (Linux)	8 bytes unsigned integer
single	float	single-precision floating point
double	double	double-precision floating point
char array (1xn) single	char *	
cell array of character vectors	*char[]	

In addition, MATLAB also shares some methods that can be seen in C in regards to manipulating different-data-type numbers, such as `isnan`, to check for Not-a-number,

`isinf`, to determine if infinite number, and typecast for conversion between data types. However, although both MATLAB and C++ support complex numbers, the way they handle them are distinctively different. While MATLAB, from the users' perspective, defines complex numbers as a particular case of regular real number, with some methods to get the real/imaginary unit, C++ uses a predefined class in std::complex library.

It is worth mentioning that there are differences in the C/C++ storage size of these types when used in Windows and Linux. To be specific, signed and unsigned long determines 8-byte storage for the number in Linux; meanwhile, in Windows, the storage size is only 4 bytes. For being a scripting language, MATLAB data types do not depend on the operating system. MATLAB provides additional methods for retrieving minimum and maximum values of a number with the defined data type. For example, `realmax` and `realmin` return range of a double-precision floating-point number, the same for integer with `intmax` and `intmax`.

Therefore, it can be concluded that MATLAB and C/C++ may share a major similarity in data types.

The following key point of being a scripting language is the data-type assignment. The scripting language is highly flexible in choosing data types for variables. It is commonly known that in C or C++, declaration of the variables with suitable data types is a must and occurs before the variables are taken into use. MATLAB, however, implicitly assigns a suitable data type depending on the value assigned to the variables. In simple words, variables in MATLAB can be used without any declaration, and their data types are assigned automatically by the software. Nonetheless, if not explicitly stated, MATLAB will assume all numerical variables as double-precision floating-point numbers as they are the widest-range real numbers.

Another critical factor that is needed to be investigated during the code generation process is the difference in the data structures. While the most basic data containers are arrays in C and C++ or more special ones such as vectors, lists, stacks, sets, etc., can be found in C++, MATLAB treats all of its numerical objects as arrays. For example, a scalar in C/C++ can be defined with `integer` or `char` type, MATLAB defines it as a 1x1 array of `integer` or `char` . Users may find the words *vector* or *matrix* used in MATLAB documents; however, they are some specific cases of an array in which *vector* is a 1xN or Nx1 sized array (N > 1) and *matrix* is an array with at least three dimensions.

MATLAB does have some unique data structures of its own. Some of them are objects and cells. Cells, in fact, are arrays. However, it has an exceptional characteristic, that is, it can contain any kind of data, including text, objects, or numerical arrays. Cells are flexible for data arrays that does not have the same size. Objects are data structures to be used for specific purposes; for example, a camera object contains parameters of a camera, including radial distortion, tangential distortion, intrinsic matrix, etc. Objects act

similarly to C/C++ *struct* , therefore, they can be converted forward and backward.

3.3 MATLAB Coder - features and prerequisite for usage

MATLAB Coder is a tool to generate C and C++ code from MATLAB code for embedded hardware. MATLAB Coder supports a wide variety of toolboxes. It can also handle dynamically allocated arrays, namespaces and classes in C++. To begin using MATLAB coder, users should analyze the software requirements in term of what programming language to be used and which settings can be applied.

Table 3.2. *Difference in generated C and C++ from MATLAB code*

Generated C++	Generated C
Support overloaded functions	No support overloading of functions
Share the same identifiers in different namespaces	No ability to use namespace
Explicit type cast ("static_cast")	Cast operator
Standards supported: ISO/IEC 14882:2003 (C++03), ISO/IEC 14882:2011(E) (C++11)	Standards supported: ISO®/IEC 9899:1990 (C89/C90 (ANSI)), ISO/IEC 9899:1999 (C99 (ISO))

The following step would be to determine the compilers. Fortunately, MATLAB Coder supports a wide range of C/C++ compilers with three common operating systems: Windows, Linux and Mac. Along with it, there are a few properties of generated code that users may be required to choose before starting generation:

Table 3.3. *Properties of generated code*

Properties	Possible choices
Build-types	Source code, static library, dynamic library, or even executable
Interface type	Functions or methods
Hardware type/ device	Computer/microcontroller: Atmel, RISC, ASIC, Intel or AMD Environment: x86 , x64, ARM 11,... etc (based on the hardware)
Toolchain	Microsoft Visual Studio, Mingw compiler
Other features	Dynamic memory allocation, parallelization,... etc

There are still plenty of options to customize generated code. However, due to the size

of this thesis, only ones that play an important role in developing the software will be analyzed.

3.4 MATLAB Coder - code generation process

Unlike humans, MATLAB Coder, an application, is programmed to understand the code's functionality in a limited number of syntaxes. Thus, programmers may be required to modify the MATLAB code before it can be generated. The significant modifications concern arrays/matrices (including variable-sized arrays and array operations) and MATLAB built-in functions.

3.4.1 Modification to arrays for code generation

From the above sections, one difference between C/C++ and MATLAB regarding variable definition is that C/C++ always requires the variables to be defined with data type and size (when the variables are arrays). Thus, to generate C++ code, arrays and variables in MATLAB also require some modification.

- **Arrays that have a fixed-size**

  It is simple to use one of the below functions to handle these arrays. It is noted that if dynamic memory allocation is not used, it is required to set an upper bound for the largest possible size of the array as the program will allocate only a fixed amount of memory for it.

MATLAB code 3.1. *Example of defining arrays with known size for generation.*

```
1  function [x1, x2, y1, y2, y3] = defining_arrays(d1,d2)
2      % array with size 4x3 can be defined as these 2 examples
3      x1 = zeros(4,3);
4      x2 = ones(4,3);
5      % array with size given by a variable can be defined as
6      % line 12 and line 13.
7      % in case dynamic memory allocation turned off, it is
8      % required to check for upper bound of array dimension.
9      % Example upper bounds are 10.
10     assert(d1 < 10);
11     assert(d2 < 10);
12     y1 = zeros(d1,d2);
13     y2 = ones(d1,d2);
14     % it is noted that array size can also be 0, which is
15     % then understood as an empty array and can be checked
16     % for emptiness.
```

```
17      y3 = zeros(0,3);
18      S = isempty(y3); %true
19  end
```

Defining arrays using `zeros` and `ones` function does not only help to determine the size of arrays but also initializes the arrays with all 0 or all 1. Therefore, the choice of using which function depends on the situation.

- **Arrays that have a variable-size**

  Arrays mentioned at this point are the ones whose sizes are possibly changed during calculation. It is important to tell the application the possible upper bounds of the changeable sizes, or the generated code will be forced to use dynamic memory allocation, which can reduce the execution speed. Symbol `coder.varsize` gives the application information about a possible change of array size.

MATLAB code 3.2. *Example of defining arrays with variable-size for generation.*

```
1  function [x1, x2] = defining_arrays()
2      % y is defined as a fixed-size array
3      % x1 is defined as a variable-size array as its size
4      % changes on line 8 after concatenation
5      y = ones(4,4);
6      x1 = zeros(0,4);
7      coder.varsize('x1');
8      x1 = [x1 y];
9
10     % x2 is defined as a variable-size array with upper bounds
11     % are 4 and 12 for dimension 1 and 2 respectively
12     x2 = zeros(0,4);
13     coder.varsize('x2',[4, 12]);
14 end
```

Although MATLAB provides probably all possible ways to interact with arrays in terms of sizes, operations that concern the array dimensions are a bit tricky. It is the fact that `coder.varsize` allows the code to change the size of the array but not its dimension. For example, a 3-D array can not be assigned to a 2-D array. It could seem that this does not occur a lot, yet, the situation may occur in a straightforward array operator: concatenation. Example 3.3 may explicitly describe the situation.

MATLAB code 3.3. *Example of concatenating arrays with dimension expansion.*

```
1  function [x] = concatenating_arrays(images)
2      % assume "images" is width x height x number_images array
3      numImages = size(image,3);
```

```
4
5       %%%%%%%%%%% First approach %%%%%%%%%%%%
6       %  define "x" as a fixed-sized 3-D array
7       x = zeros(4,4);
8
9       for i=1:numImages
10          % get each image from image set.
11          currentImage = image(:,:,i);
12
13          % get some information from image
14          y = process(currentImage);
15
16          % Assume "y" is 4 x 4 array and
17          % "y" is concatenated into "x" along 3rd dimension
18          x = cat(3,x,y);
19      end
20
21      % remove initialized element (1st element) from "x"
22      % "x" is a set of processed data from image set
23      % size of "x" would be 4 x 4 x number_images
24      x = x(:,:,2:end);
25  end
```

One approach to solve this problem is defining the array as 3-D at the time of initialization, and the array is then assigned with desired data instead of concatenated. This approach has a drawback that the size of the array, which is 4 x 4 based on `y` in line 14, must be known when it is defined. Adding the use of cell arrays would help in this case. A cell array is a powerful object because it can store any type of data. Although there is still one requirement using cell arrays: every element of the cell array must be assigned a value, it is still manageable.

MATLAB code 3.4. *Example of concatenating arrays with dimension expansion.*

```
1   function [x] = concatenating_arrays(images)
2       % assume "images" is width x height x number_images array
3       numImages = size(image,3);
4
5       %%%%%%%%%%% First approach %%%%%%%%%%%%
6       %  define "x" as a fixed-sized 3-D array
7       x = zeros(4,4, numImages);
8
9       for i=1:numImages
```

```
10          % get each image from image set.
11          currentImage = image(:,:,i);
12
13          % get some information from image
14          y = process(currentImage);
15
16          % y is 4 x 4 array and is assigned to part of x
17          x(:,:,i) = y;
18      end
19  end
```

MATLAB code 3.5. *MATLAB Code example 3.4 with cell arrays.*

```
1   function [x] = concatenating_arrays(images)
2       % assume "images" is width x height x number_images array
3       numImages = size(image,3);
4
5       % when using cell arrays the above code would become:
6       % "c" is 1 x numImages cell array
7       c = cell(1,numImages);
8       for i=1:numImages
9           % get each image from image set.
10          currentImage = image(:,:,i);
11
12          % get some information from image
13          y = process(currentImage);
14
15          % y is 4 x 4 array and is assigned to part of x
16          c(i) = y;
17      end
18      % get the size of result array
19      row = size(c(1),1);
20      col = size(c(1),2);
21
22      % define "x" with the size
23      x = zeros(row,col, numImages);
24       for i=1:numImages
25          x(:,:,i) = c(i);
26      end
27  end
```

3.4.2 Modification to MATLAB built-in function

It is important to be aware that not all of MATLAB build-in functions can be generated into C++ code. Some may require changing their form or syntax, and others may support only a few options. In general, there are three cases regarding build-in functions that may happen while generating the code:

1. The functions are fully generatable. The code generation process can be done without modifications to the original MATLAB code. Most functions are supported to generate into C++.
2. The functions are not generatable. It is a MATLAB Coder limitation, and currently, the only solution is to change the algorithm behind the MATLAB code, then the way the code is written changes.
3. The functions are partly generatable, meaning they can be generated in some, but not all, of their syntaxes. The code should be re-written into different syntaxes to be converted. Let's analyze below example:

MATLAB code 3.6. *Example of modifying MATLAB built-in function.*

```
1  function [imagePoints] = example_generating_function(images)
2      % assume "images" is a W x H x N array:
3      % + W, H: image width and height
4      % + N: number of images
5      [imagePoints, ~] = detectCheckerboardPoints(images);
6      % normally, this function returns a set of image points
7      % with size M x 2 x N:
8      % + M: number of points detected in the checkerboard
9      % + 2: coordinates of point in image along x and y axis
10     % + N: number of images
11     % Unfortunately, above syntax is not generatable.
12
13 end
```

The function needs discussing in this example is `detectCheckerboardPoints`. As declared in MATLAB tutorial document, `detectCheckerboardPoints` can be written in several forms. Line 5 in MATLAB code 3.7 is the shortest and the most common syntax of the function to detect the checkerboard points in the image set and stack them up along the 3rd axis. Nonetheless, this syntax is not supported by MATLAB Coder for a generation. The generatable syntax is that the function can only detect the points in one image at a time. The modification made may seem fairly simple, as can be seen from line 13 to line 24: putting it in a loop to perform detection on each image and manually stack them up as in line 23.

MATLAB code 3.7. *Example of modifying MATLAB built-in function.*

```
1  function [imagePoints] = example_generating_function(images)
2
3      numImages = size(image,3);
4      % Assuming the checkerboard in the given image has 450
5      % points
6      imagePoints = zeros(450,2,numImages);
7      for i=1:numImages
8          % get each image from image set.
9          % currentImage is an W x H matrix.
10         currentImage = image(:,:,i);
11         [imagePoints(:,:,i), ~] = ...
12                     detectCheckerboardPoints(currentImage);
13     end
14 end
```

Nonetheless, that is still not the end of the story. The modification raises one problem: the number of points in the checkerboard must be known when the `imagePoints` array is defined in line 6, which is unlikely to happen because the image quality is not always sufficiently good for the algorithm to detect all the points in the checkerboard. This comes back to MATLAB code 3.4. In the end, combining MATLAB code 3.4 and the above modification should be enough to generate the function.

3.4.3 Code generation process

There are three features of code generation using MATLAB that should be taken into account.

1. Feature 1:

   The form of code used in the generation process must be a MATLAB function. To be specific, it packaged inside `function` and `end`, then stored in an `.m` file. It is worth mentioning that generated code using MATLAB coder defines and uses its libraries, including mathematics library and data structure. For instance, generated code of a function that calculates the square root of a real number does not use the standard `math.h` library from C++. As a result, the product of the code generation process is not only a single `.cpp` file (C++ source file), but consists of many others which act as MATLAB's mathematics library and data structure such as `sqrt.cpp` (square root), `cat.cpp` (array concatenation), `det.cpp` (array determination).

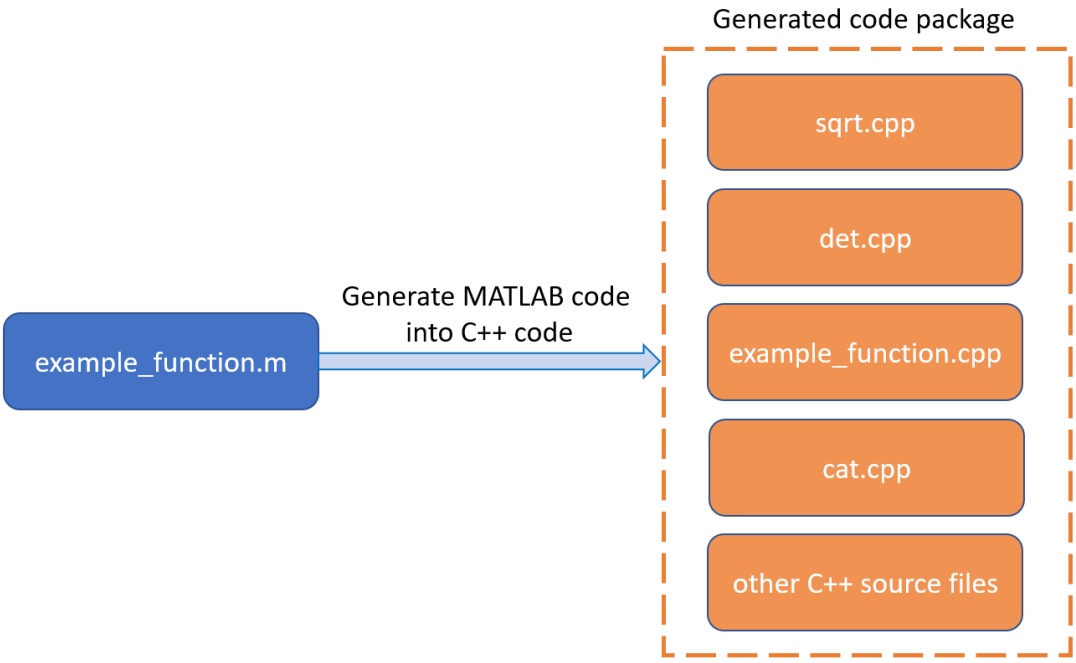

Figure 3.1. *A package of generated code.*

2. Feature 2:

   Another reason for which the `.cpp` files are called libraries in this thesis is how generation processes in case of multiple functions selected at once. As mentioned, the code generation tool takes `.m` files as input. If two `.m` files are generated separately, the products may contain some similar `.cpp` files. In contrast, when multiple `.m` files are selected in one generation time, the generated products do not show any redundancy of the library code. It means the library code is used and shared between the selected functions in the input `.m` files. This feature plays a decisive role in the project development process.

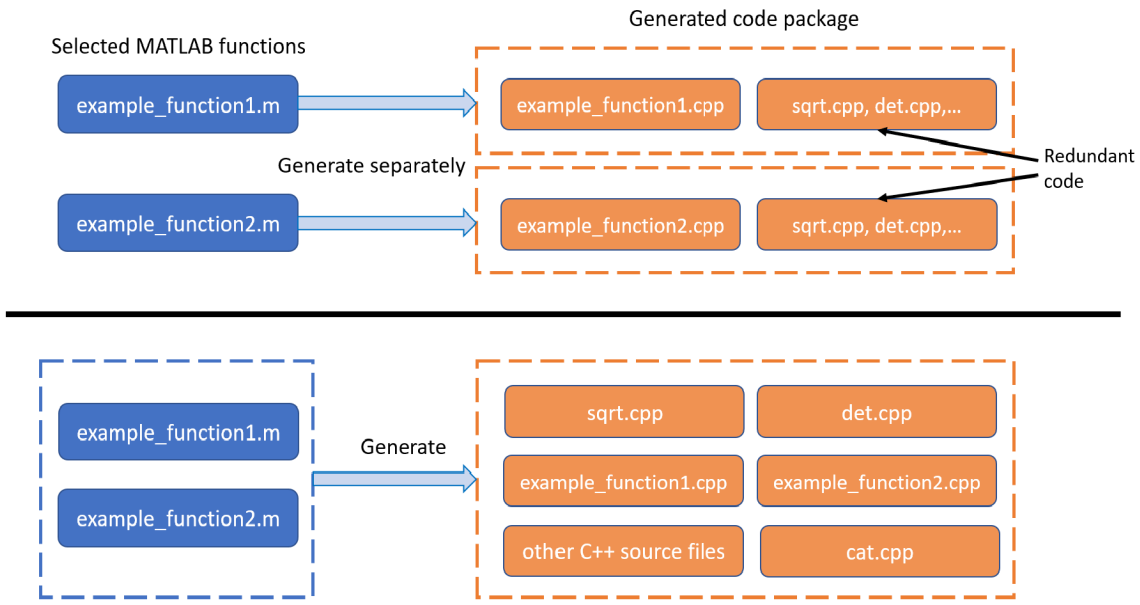

Figure 3.2. *Redundancy of generated code.*

3. Feature 3:

   Moreover, although certain generated functions may be considered functionally equivalent, the actual writing of the code generated is not consistent. In other words, the result of the generation process can be different between times of code generation. The difference, even though as minor as a change in variables' names or code writing style, may affect the development process of the whole project.

4. DESIGN AND IMPLEMENTATION OF CODE GENERATED FROM MATLAB

With all the essential features of MATLAB Coder studied, it is also important to use MATLAB Coder in a proper way. In other words, there is a need for a good software development process model with MATLAB Coder. All the phases should be included: Planning, Design, Implementation, and Testing.

4.1 Planning

Planning in the iterative model is the most critical phase. In this phase, software descriptions and requirements need to be thoroughly investigated.

4.1.1 Software description and requirement analysis

It can be considered that the process of developing software generated from a MATLAB code is a type of model-driven development in which the original MATLAB code plays the role of the model program. The first important step is to investigate software requirements. Unlike other types of software development, in this process, it is easily understandable that MATLAB model code itself should be the first and foremost requirement for the generated software. The generated code must be ensured to have the exact functionality and behaviors inherited from the MATLAB model software. The other requirements occasionally concern software behaviors or may be considered as some additional features.

Table 4.1. *Some suggestion on possible requirements*

	Requirements
User interaction	- Inputs and outputs should be user friendly. - Progress message must be sufficiently informative.
Operating system	- Operating system should be clearly defined (Linux, Windows). - Platforms/Tool-chains (MSVC, GNU) are used to build/ compile software code.
Functional	- Software operates in different states. - There must be control of failures/ exception handling. - Compliance with software safety standards.
Non-functional	- Software is designed into module (Modular design). - Software performance (resource cost, processing time).

4.1.2 Planning of development and testing process

As can be seen in chapter 3, the MATLAB model code is not always ready for code generation. Therefore, an intermediate step of modifying the MATLAB code is required before generating it into C++. This step is crucial as it affects the process of development and the testing procedure. Changes are made to the original MATLAB code, which is the model and should not be changed. Therefore, the testing process must be carried out carefully to ensure the fulfillment of the requirements.

In this type of development model, it is essential to point out some characteristics of the testing phase by answering the following questions.

- **The purpose of testing?**

  The testing process ensures software functionality and dependability, meeting users' requirements and safety standards.

- **What is software tested against?**

  The software developed based on a model must behave like the model in terms of significant functionality. The first test principle is to ensure that generated software and the original MATLAB code share similar behaviors in standard cases and possibly in error cases.

  The second principle is to follow the design requirements if they are defined differently than how the original MATLAB software behaves and additional features described in software requirements.

In summary, the generated software is tested against its original MATLAB code and design requirements.

- **Where/In which phase can the defects occur?** It is required to analyze coding phases that have a high probability for defects to occur to create a good test plan.

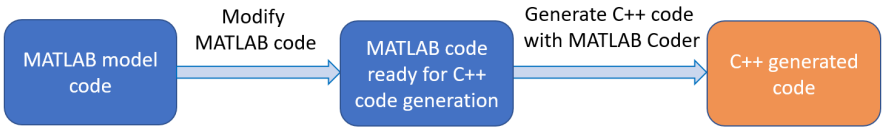

Figure 4.1. *Coding phases from MATLAB model code to C++ generated code.*

From the 4.1 chart, the first possible occurrence of defects or faults is in the MATLAB model program. As it is the model, it should be assumed to be tested and completely error-free. However, the model software may require some changes when its behaviors in particular cases are stated differently in the design requirements. In other words, the user's requirements may try to modify the model software behaviors in those cases. Therefore, it should not be considered a software defect, and there is no need for a testing suite for this.

The modification process, which is done to get the MATLAB code ready for generation, may cause software errors. Therefore, there should be a testing process to ensure the modification process make no change to the main functionality unless stated otherwise in the design requirements.

The last coding phase, generating C++ from MATLAB code, has the potential to cause unexpected defects. Therefore, this can be considered the most complicated phase that requires thorough testing.

With all three questions answered, it is suggested that there should be at least two stages of testing: testing after the modification process and testing after code generation. The objective of testing after MATLAB code modification is to ensure that modifying the MATLAB model code does not affect its functionality and behaviors. Because the modification is done in MATLAB environment, testing for modified code is, in fact, an online model-driven test. In contrast, testing C++-generated code is the offline version. The same test suites can be used in both stages as MATLAB code, and C++-generated code share the major functionality.

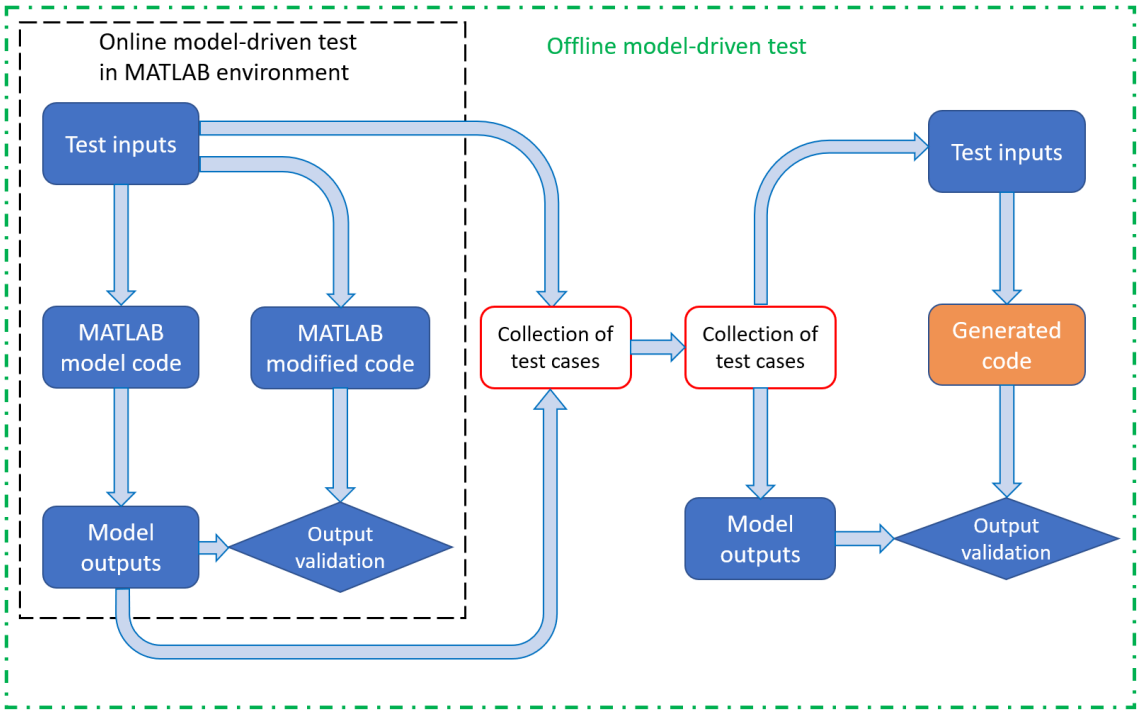

Figure 4.2. *Planning two testing stages.*

The first testing stage is simple, considering MATLAB modified code is tested against its original version, which is also a MATLAB code. On the other hand, the second stage should be divided into two levels: unit testing and integration testing. There can be system testing in some large projects, but the two lower levels should be sufficient within the scope of testing MATLAB code generation.

- **Unit testing**

  Theoretically, from section 2.5.2, unit testing is defined as testing performed on a unit function or the smallest testable code. Looking back at section 3.4.3, unit functions means every generated function with all `.cpp` files. This raises the question: *one MATLAB function is generated into multiple C++ `.cpp` files act as libraries. Is it indispensable to test for all of them?* From the author's perspective, it is not needed to do testing on them. Two reasons can give justification for this idea.

  1. The functions such as `det, sqrt, cat` and even more complex ones, for example, `computeFiniteDifferences` are MATLAB built-in functions, which means that there is no involvement of human in developing them, or it can be said that they was designed and fully tested by MathWorks team.
  2. The functions are then generated automatically with no modifications as a MATLAB Coder feature.

  So all automatically generated `.cpp` files are one feature of MATLAB using its built-in code. Assuming there is no error or defects during that process, a test suite for

them is not necessary. To be specific, the code that can be tested at the unit level is each function written in the `.m` files. Even if the assumption is wrong and there are faults in the MATLAB Coder, unit testing should fail due to that errors. The only challenge is tracing that defects. In summary, in this type of project, *unit testing is testing performed on a unit function or on the smallest testable code that is not a MATLAB built-in.*

- **Integration testing**

  Integration testing is executed on the integration of unit functions. Suppose the MATLAB model software comprises multiple functions within multiple `.m` files, testing on the combination of C++ generated functions against its MATLAB version is a must.

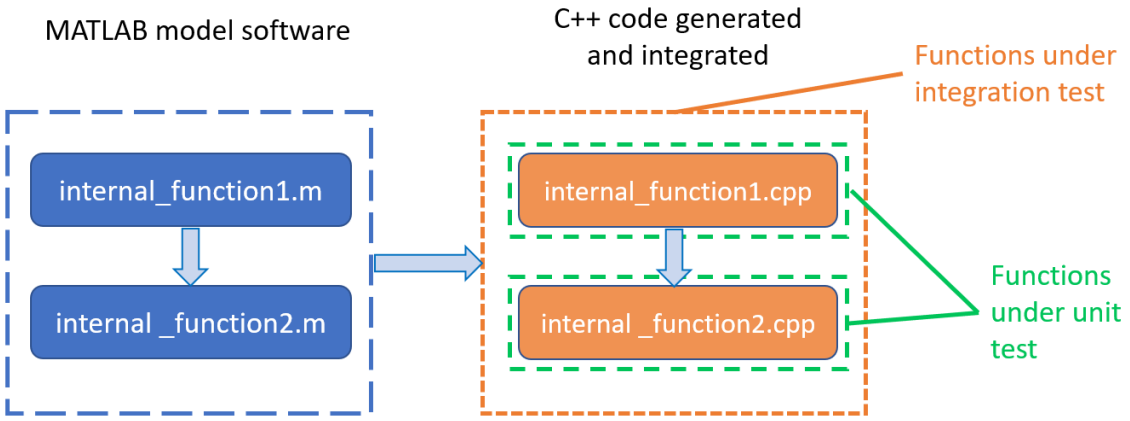

Figure 4.3. *Two levels of testing.*

Now that the overall testing plan is done, test suite or test cases are next to be defined. There is a significant number of testing types to be used. The most common and necessary ones are functional tests and boundary tests. In this type of project, the error injection tests should also be considered one crucial testing type.

Table 4.2. *Some testing types that can be applied.*

Testing types	Definition
Functional test	Functional test is a general term of all kind of testings regarding functionality of software. It validates application features against functional requirements.
Boundary test	Boundary test shares similar attributes with functional test. The difference is, in boundary test, the software is tested with its inputs changes within their valid range (from minimum to maximum as stated in design requirement).
Fault injection test	Software is injected with specific faults to validate its behaviors and error handling mechanism.

In terms of non-functional testing, code coverage and performance evaluation are considered beneficial. In addition, a static test with coding standards is an option depending on project requirements. MISRA compliance is advertised as supported by MATLAB Coder. Therefore, it will be used as a coding requirement in this thesis.

4.1.3 Software development process model

] Choosing the software process model (SPM) is important to optimize the time and effort and maximize the maintainability and scalability of the product. In the type of project that heavily depends on MATLAB as the primary tool to develop the code, the characteristics of MATLAB and MATLAB Coder should be considered when choosing the SPM. The most common and straightforward is the waterfall model. This model may suit small and one-time projects but shows challenges when updates or upgrades are implemented. A better SDLC to be suggested is the iterative model.

From the illustration 3.2 in Feature 2 in section 3.4.3, if the MATLAB functions are generated separately, the generated product may contain redundant code. Thus, in implementing a new feature or new function, the previous ones should also be re-generated. The generated code of previous functions is then overwritten; testing for it should also be re-executed.

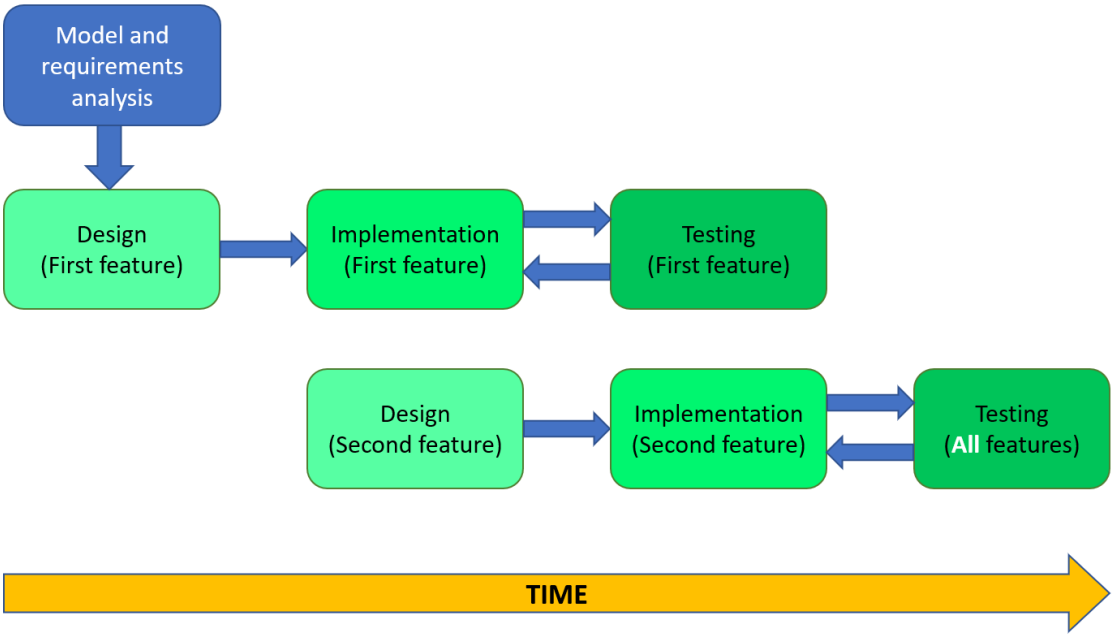

Figure 4.4. *Software development process for code generation.*

4.1.4 Introduction and Planning of demo project

To evaluate the effectiveness of MATLAB Coder, this thesis will apply discussed knowledge to a generation of computer-vision-based MATLAB software. The deliverables shall be the source code of C++ functions but shall be called *Application Program Interface(s) (APIs)* this thesis so as not to cause confusion with the term *functions*.

As the first requirement, two APIs' specification is described in table 4.3 and 4.4.

1. **calibrateCamera**

Table 4.3. *`calibrateCamera` specification.*

Inputs	• `std::vector<imageWrap>` `calibrationImages` images of the calibration target • `calibrationPatternParameters` `calibPatternParam` calibration pattern parameters
Outputs	• `intrinsicParameters` `camParam` the computed calibration results based on the input images
Functionality	Analyzes the set of input images, detects the presence of the chessboard calibration pattern and computes the intrinsic camera parameters. Internally, validates the inputs, calls the MATLAB Vision library implementation of calibration pattern finding and camera calibration and formats the calibration data according to the API specification.

2. **calibrateHandeye**

Table 4.4. *`calibrateHandeye` specification.*

Inputs	• `std::vector<imageWrap>` `calibrationImages` images of the calibration target • `std::vector<double4x4>` `robotPoses` The HTM of robot poses in the robot coordinate system corresponding to `calibrationImages` • `calibrationPatternParameters` `calibPatternParam` calibration pattern parameters • `intrinsicParameters` `camParam` the computed calibration results based on the input images
Outputs	• `double4x4` `handEyeCalibration` the computed hand eye calibration as a HTM based on the input images and the robot poses • `double` `reprojectionError` Error metric as computed by the algorithm • `double` `rotationError` Error metric as computed by the algorithm • `double` `translationError` Error metric as computed by the algorithm
Functionality	Computes the Hand-eye calibration using MilliShah's algorithm.

Additional requirements are listed in the below table. There can be other requirements regarding user interaction with the software. However, within this thesis's scope, only requirements related to developing code generated from MATLAB are listed.

Table 4.5. *Additional requirements for demo project.*

Requirements	Details
Platform	Intel Processors
Device type	x64
Toolchain	mingw64 (GNU)
Coding Standard	MISRA C++
Error tolerance	Software must ensure the accuracy of calculation results. The error tolerance rate must be discussed and agreed for testing.
Error handling	All possible errors should be explicitly documented. In case of running into an error, software must terminate with an informative message with the stop code.

4.2 Design

In the design phase, requirements are processed into detailed specifications, including software flow charts and a detailed plan for testing and implementation of the error handling mechanism.

4.2.1 Data flow charts

Data flow charts depict information about the connection between MATLAB functions. The functions are then generated into C++. It is suggested that there should not be any calculation or processing in the transition between the functions because that piece of code will not be generated.

Regarding two APIs, `calibrateCamera` API consists of two functions:

- `preprocessImages`
- `calibrateOneCamera`

It is worth mentioning that there are other internal functions which are called by `calibrateOneCamera`.

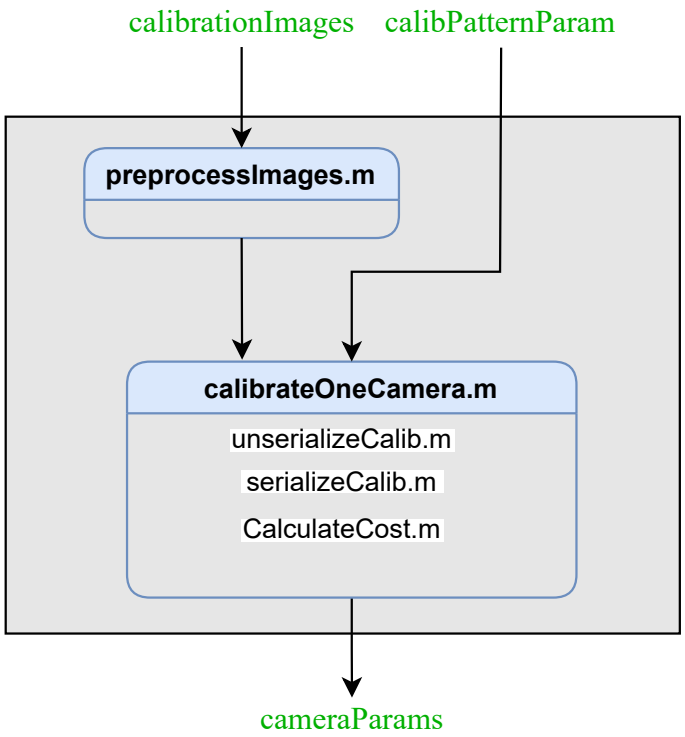

Figure 4.5. *`calibrateCamera` data flow chart.*

`calibrateHandeye` API consists of five functions:

- `preprocessImages`
- `readRobotPoses`
- `ComputeCamExtrinsics`
- `HandeyeShah`
- `computeErrors`

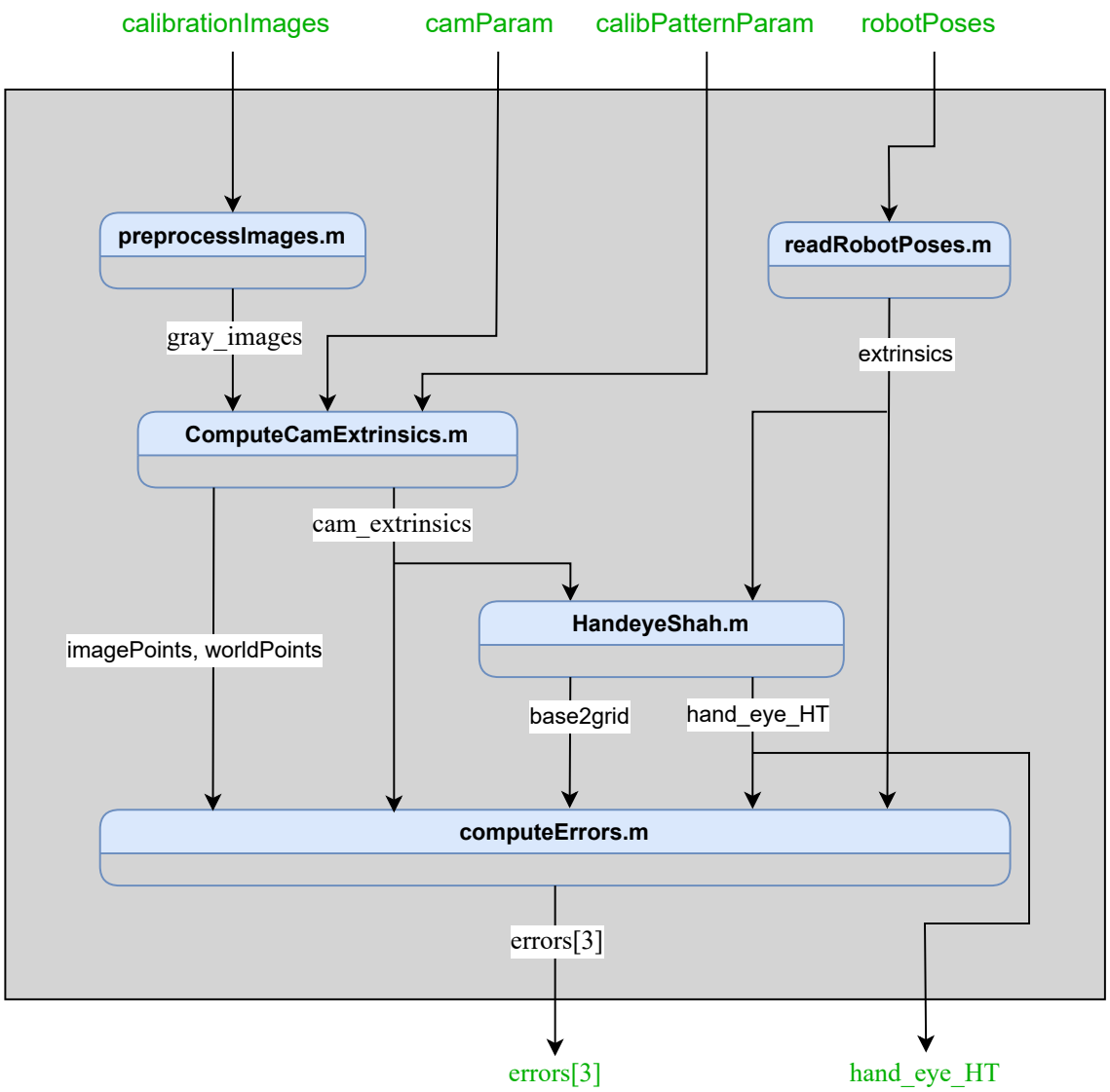

Figure 4.6. *`calibrateHandeye` data flow chart.*

Regarding code generation, it is possible to combine all of the internal functions into one `.m` file and then generate it. By doing that, the APIs are treated as black boxes. However, this approach may cause extreme difficulty organizing and testing the code. Instead, each function should be independently generated. That way, it does not only allows performing tests on each function, but also help in implementing the error handling mechanism.

4.2.2 Design of error handling mechanism and testing

1. **Design of test suite**

   Designing the error handling mechanism and testing can be considered the most challenging throughout the development process. Although all the error cases are well defined and handled by MATLAB code, the C++-generated code may not inherit

that ability. The programmer must adapt all the possible error cases and behaviors from the original MATLAB code to C++-generated one. These errors, however, hardly ever occur during regular use. So, the quickest method to trace all of them is to put the software through all designed test cases, which must represent all possible use cases of software.

By doing that, the testing phase happens before implementation is complete. In other words, for every new feature and error handling case implemented, the code must be put to the regression test until no failed cases are left. In summary, the testing phase and implementation phase are performed cyclically.

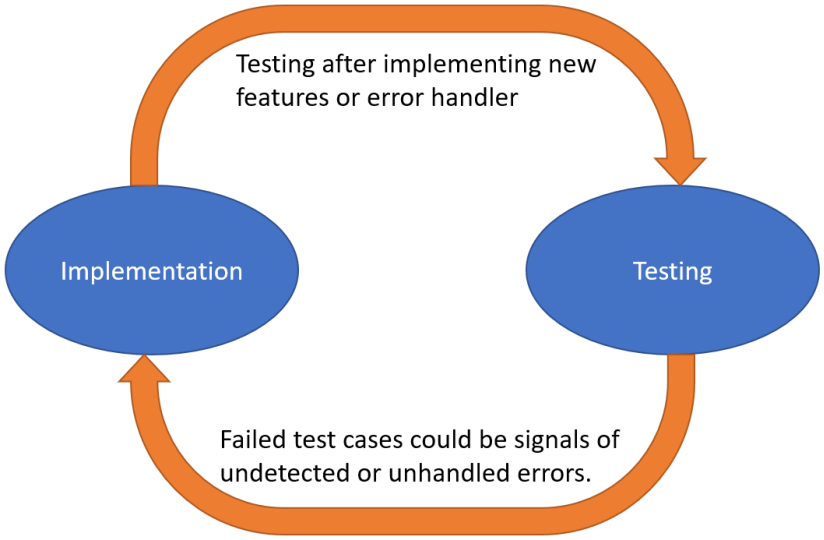

Figure 4.7. *The cycle of the implementation and testing phases*

The testing phase does not just play the role of validating software, but it also involves developing the software. Therefore, the test suite must be designed with extreme thoroughness. For each testing type, there should be test cases as defined below

- Functionality test

  The functionality test is to put the software in a practical working condition. So, there is a need for sufficient data set to validate all implemented features.

- Boundary test

  The boundary test is to put the software in a simulation of severe cases, which may only occur by mistake. When all inputs are defined with a valid range, there must be at least 3 test cases for each software input including input at lower bound, upper bound, and in the middle of that range.

- Error injection test

To effectively design the error injection test cases, programmers must have knowledge on their MATLAB code. There should also be a plan for specific cases corresponding to each input argument. For example, in some cases, such as a poor quality image as the input, the software outcome may be unpredictable. The error injection must cover all those cases to prevent the unexpected behaviors of software.

It is crucial to acknowledge that it is barely possible to have sufficient sets of images to simulate all real-life cases in computer-vision-based software. Furthermore, the performance of computer-vision-based software, in general, depends heavily on the quality of input images. Therefore, the main focus of testing a generated computer-vision-based software is to evaluate the generated code's performance compared to the original code.

After the test cases are decided, the next step is to design an implementation of test code. As in the test plan, there are two testing phases: testing after modification process and testing after code generation. It is also crucial to develop a strategy for fixing the defects.

(a) Testing after MATLAB code modification

Testing in this phase is an online version of model-based testing. The inputs of both MATLAB model code and modified code are set with designed values, and the outputs are compared. If this testing phase detects any issues, they should be solved directly in MATLAB code.

(b) Testing after MATLAB code modification

As opposed to the first stage, the second one is the offline version of model-driven testing. The problem to be solved is to use a test case format that can be read by both MATLAB and C++. Most MATLAB variables are arrays, which may contain thousands of elements. It is unlikely to write down all the data manually. Two possible file types that can be read and written by MATLAB and C++ are MATLAB `.mat` file and `.json` format.

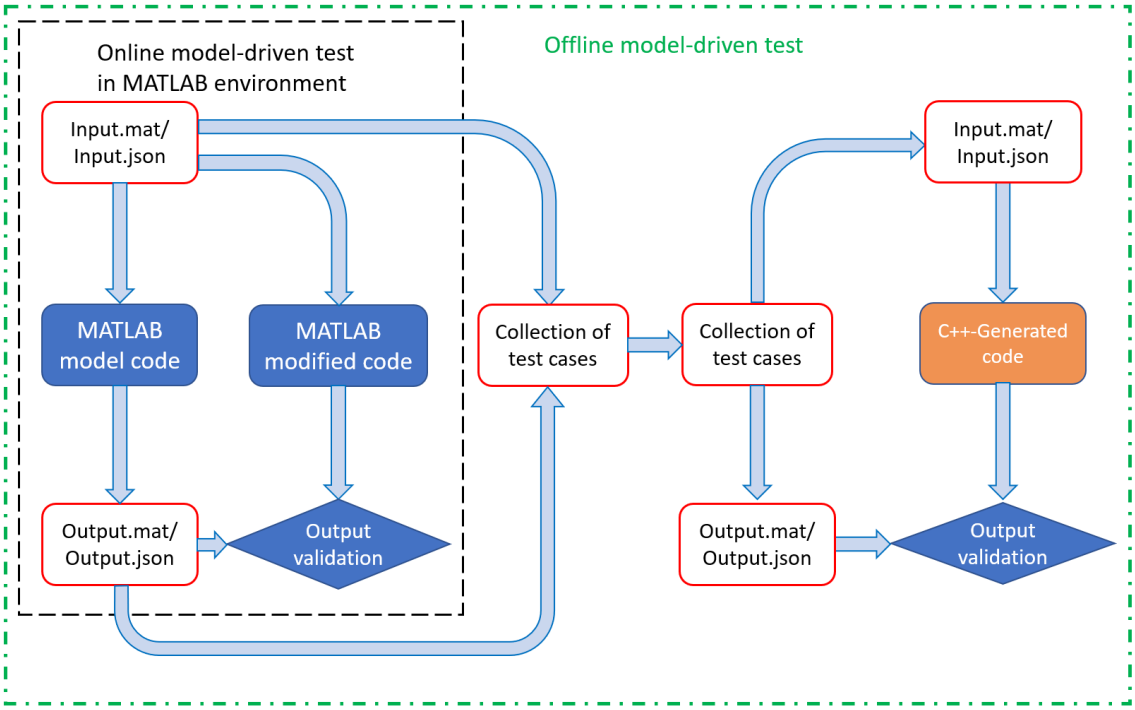

Figure 4.8. *Implementation of testing block diagram.*

Because both testing phases use the same test suite, if an error exists during this phase but passes the first one, it must occur during code generation. In other words, MATLAB code could function properly, but MATLAB Coder fails to understand the code correctly. Regarding feature 3 in section 3.4.3 and the process model in section 4.1.3, it is a waste of effort resolving the problems by fixing generated code, considering the code will be overwritten when being re-generated in the next development phase. As a result, it is most efficient to modify the original MATLAB code so that MATLAB Coder can correctly generate it into C/C++.

2. **Performing static test**

   Static testing is performed by an automated tool. As there are not any test cases in this type of testing, static testing should be done at last, when all the dynamic testing phases are completed.

3. **Design of error handling mechanism**

   After the code is ready for generation, MATLAB Coder will handle most work. The important objective of this section is to implement the code for error handling correctly. Assume that MATLAB code runs into an error when executing the test, especially a simulation of a severe case; it always gets terminated. As a result, there is no output to form a complete test case. It depends on the situation to find a suitable development process.

(a) The condition that triggers the error can be specified.

It is suggested to add an `if` statement in the original MATLAB code to catch each error and re-generate the code. The first reason for this approach is that users and programmers can track what issues occur during operating software. The second reason is the re-generation of MATLAB code in the next development cycle. Regarding the development process in section 4.1.3, when the code is re-generated, the implementation of the `if` blocks for catching the errors are kept in the generated code.

The generated code now has a flag to give information about the status of the API. For example, it can tell if the code operates without any issue or gets terminated because of an error.

(b) The condition that triggers the error is not explicitly exposed to developers.

When MATLAB software runs into an error and gets terminated, occasionally, the error is detected within a built-in MATLAB function, whose code should not be modified. Thus, it is unlikely to know precisely at what point of the code the error is triggered. It is then hardly able to implement an `if` block to catch the error. Because the original MATLAB code and generated code may behave differently, the suggested approach is to keep generating the code and directly apply the test case, which causes the error to generated code.

i. The generated software operates without interruption, but the outputs may contain unidentified objects or unusable values.

The code can be kept the unchanged but a check block can be implemented at the end after executing the generated code. To be specific, looking at the below graphs 4.9 and 4.10, an `if` block in *validateOutputs* can be informative to the users about the error.

ii. The generated software is interrupted by an unknown issue such as `segmentation fault` or getting stuck in an infinite loop

In these cases, the only option is to code `if` block at the beginning of the function to prevent one specific error case to be injected in generated software. This piece of code should be in `validateInputs` .

4.2.3 Data flow chart of C/C++ software

The data flow of C/C++ software is illustrated in the graphs: 4.9 and 4.10. While all the blue boxes are generated codes from MATLAB which has an addition `err` output as the error flag, the yellow boxes are handwritten in C/C++ for converting C/C++ data types into MATLAB data types, validating the function inputs to prevent the error before executing

generated functions, and validating outputs to check if there is any unusable value after executing generated functions.

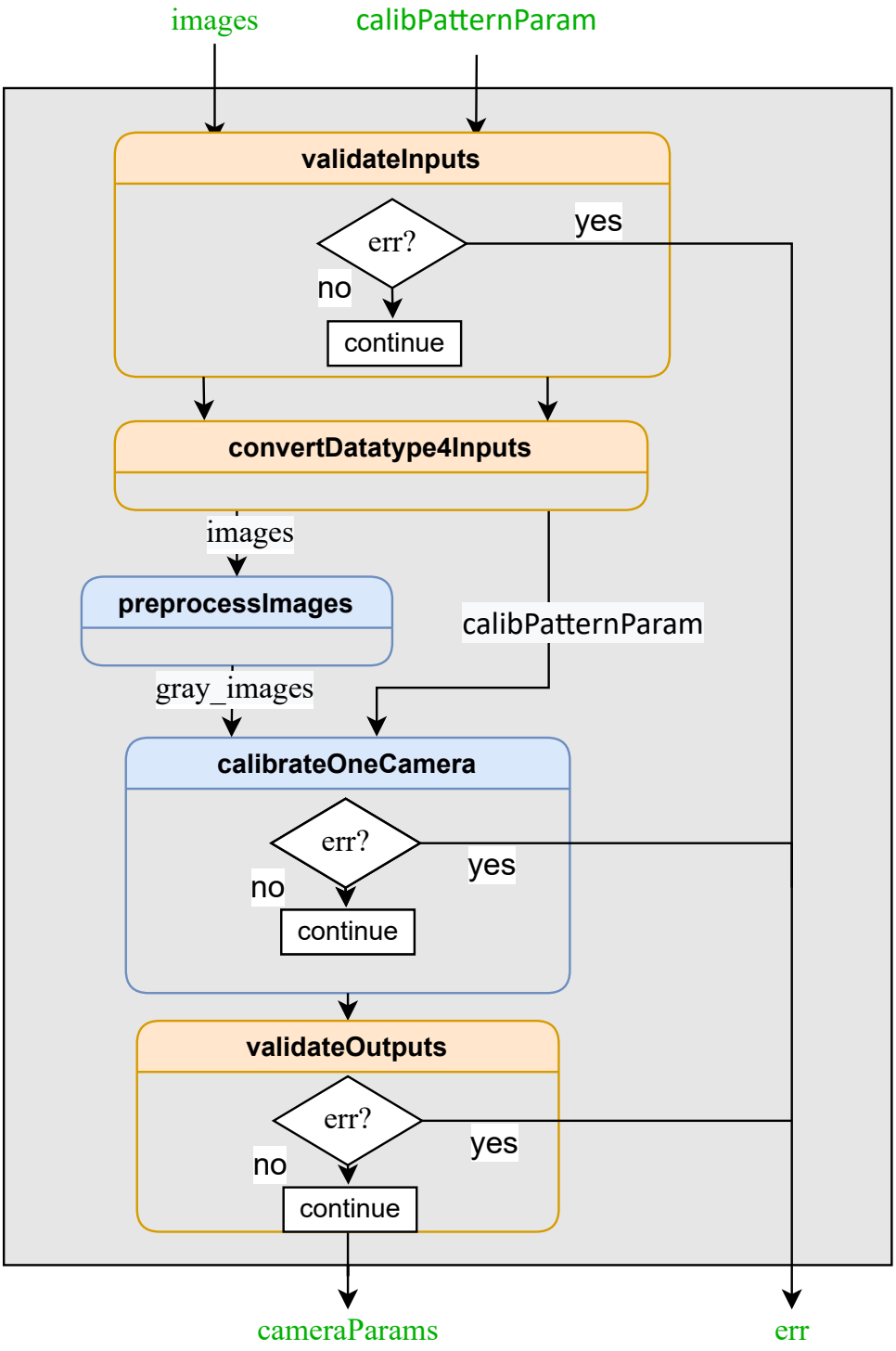

Figure 4.9. *Complete implementation of C/C++* `calibrateCamera`.

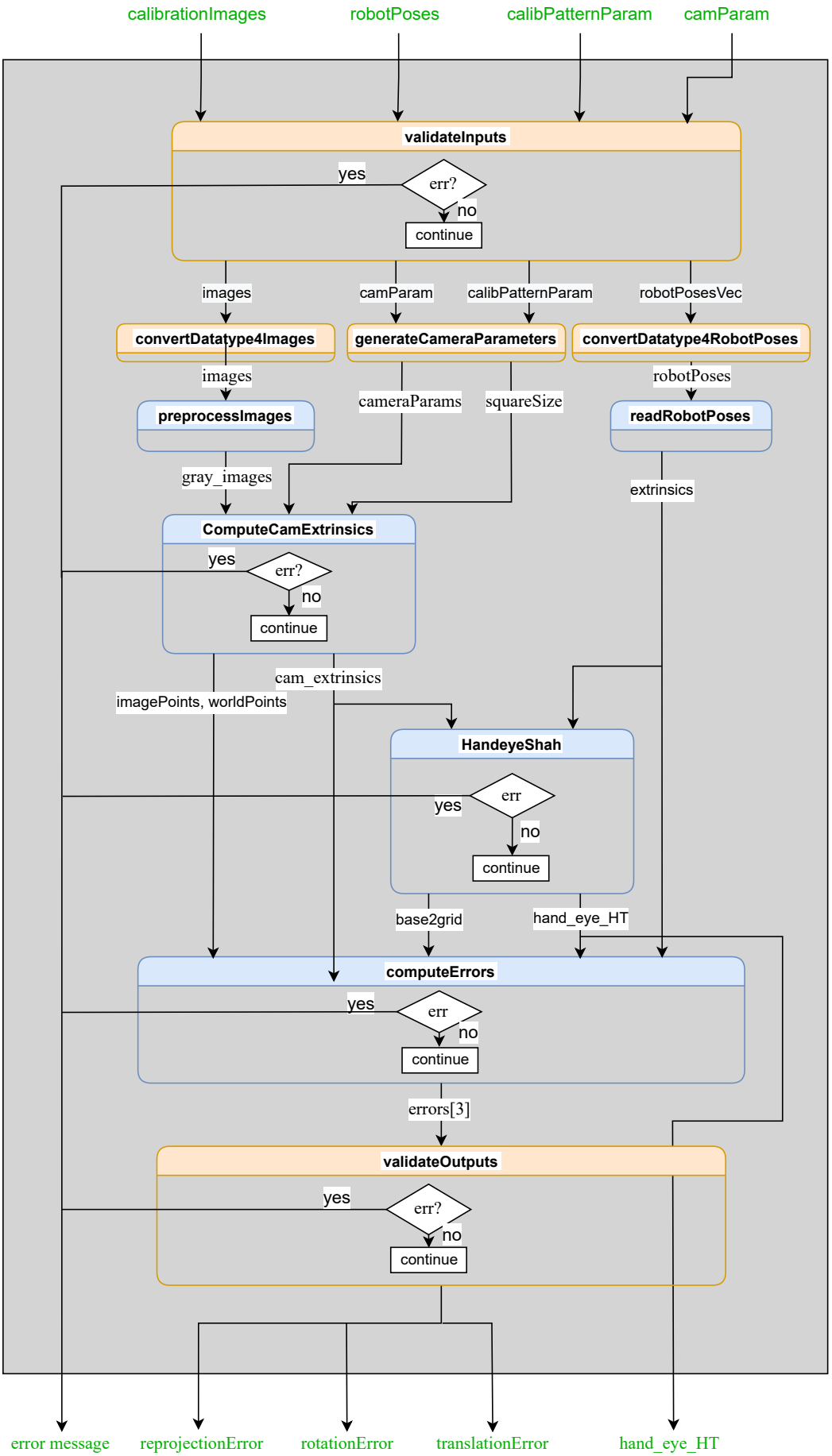

Figure 4.10. *Complete implementation of C/C++* `calibrateHandeye`.

4.2.4 Tools for developing and testing

In this phase, it is also required to document which tools are used during the process.

Table 4.6. *Tools used for development of demo code.*

Tool	Explanation
MATLAB R2021b	MATLAB is the center of the development process.
Eclipse 4.22.0	Eclipse is an IDE for coding that is lightweight and provides good support for C/C++. This IDE will be used to program, compile and test the generated code
CUTE-framework 5.10.0	CUTE-framework is a simple testing framework which provides numerous types of `ASSERTION`. The framework would be implemented into the generated software as a test code.
PC-lint 1.4.1	PC-lint is a static test tool that support MISRA C++. This tool is used for static test of generated code.

4.3 Deployment of generated code

Code generated from MATLAB is fairly easy to use. It can be compiled as ordinary C/C++ code with multiple options. Some suggested flags are listed in table 4.7

Table 4.7. *Some suggest flags that can be helpful during development.*

Tool	Option	Explanation
C++ Compiler	*-c*	Compile the source files without linking
	-m64	Generate code for a 64-bit environment. This option affects `int`-type variables.
	-O3	*-O1*, *-O2* and *-O3* are different levels of optimization.
	-g3	*-g1*, *-g2* and *-g3* request different levels of debugging information. This flag is useful to trace code defects during the development process.
	-fopenmp	Enable OpenMP to multi-threading the software.
C++ Linker	*-static*	Prevents linking with the shared libraries.
	-m64	Generate code for a 64-bit environment. This option affects `int`-type variables.
	-fopenmp	Enable OpenMP to multi-threading the software.

5. EVALUATION OF GENERATED SOFTWARE AND THE DEVELOPMENT PROCESS

After the implementation and testing have been done, this chapter evaluates the software's overall aspects, thus resolving whether the proposed software development process model is a practical design model capable of ensuring output software quality and dependability.

5.1 Evaluation of non-functional characteristics

There are three non-functional features worth discussing:

1. Code writing style and structure

   The structure of generated code can be arguably good. The class methods of built-in libraries are very well restricted to prevent accidental user changes. Generated function has a `void` type with all inputs and outputs as `C++ references`. The input arguments are secured with `const` so that their values remain unchanged after calling the generated functions.

C++ code 5.1. *Example of a generated function from MATLAB.*

```
1  // Function Definitions
2  //
3  // Initialize arrays of poses and extrinsics
4  //
5  // Arguments    :
6  //      const coder::array<double, 3U> &robotPosesVec
7  //      coder::array<double, 3U> &poses
8  //      coder::array<double, 3U> &extrinsics
9  // Return Type  : void
10 //
11 void readRobotPoses(
12         const coder::array<double, 3U> &robotPosesVec,
13         coder::array<double, 3U> &poses,
14         coder::array<double, 3U> &extrinsics)
```

In term of coding style, generated libraries implements some own data types, one of which can be seen in C++ code 5.1 - the `coder::array`. There is no usage of C++ standard template libraries found in the code. In addition, commenting is only at the beginning of a function.

Concerning coverage quality, most generated functions may reach 90 percent, which can be acceptable. On the other hand, the percentage among the library files can be deficient, but this is understandable as some libraries are always generated independently of the primary function and not always taken into use.

In general, code generated from MATLAB is considerably packed because when being used, it can be treated as a black box regardless of what code is inside. However, the code is challenging to read and understand due to a lack of comments between code lines. As a result, generated code is mediocre in maintainability and scalability.

2. Static test result with software standards

   By performing static testing using PC-lint, the report indicates that generated code can be considered to pass the test. Among thousands of lines of code, there are only two issues.

Table 5.1. *MISRA C++ Rules that are violated.*

MISRA C++ Rule	Output	Occurrence
Rule 0-1-7	The value returned by a function having a non-void return type that is not an overloaded operator shall always be used.	*coder_array.h*
Rule 5-0-16	TA pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array.	*BFGSUpdate.cpp*

The violation of Rule 0-1-7 can be simply resolved by adding `(void)` to make the code become similar to `(void) func (para2);`. For the rule 5-0-16, the static check tool detects a potential out-of-bound array access when using a variable to index an array, for example, `a[i] = 0;` with `i` is the array index. The issue may not be resolved in the static test, but the array-access issue can be prevented during run-time by adding a boundary check for the indexing variable.

In general, code generated from MATLAB can be seen as MISRA compliant after some minor issues are resolved.

3. Software performance

   Software performance is measured by operating time, and evaluated in a comparison with the original MATLAB software. With functions that does small calculation, the difference can be minor. Therefore, some sets of high resolution images are used to evaluate the software performance.*Data set 1* comprises simulating photos which are easier to process than ones in *Data set 2* and *Data set 3*

Table 5.2. *Execution time of MATLAB and generated* `calibrateCamera` *code.*

Compiler	Software	CPU Utilization	Data set 1	Data set 2	Data set 3
MATLAB	MATLAB	20% - 30%	18.52s	198.34s	189.44s
GCC	Single-threaded	20% - 30%	68.24s	107.81s	60.01s
	Multi-threaded	40% - 60%	79.34s	1623.10s	1624.34s
MSVC	Single-threaded	10% - 20%	139.41s	286.32s	305.18s
	Multi-threaded	90% - 100%	82.17s	373.65s	454.08s

Table 5.3. *Execution time of MATLAB and generated* `calibrateHandeye` *code.*

Compiler	Software	CPU Utilization	Data set 1	Data set 2
MATLAB	MATLAB	20% - 30%	12.74s	170.23s
GCC	Single-threaded	20% - 30%	71.18s	112.05s
	Multi-threaded	40% - 60%	91.01s	1543.27s
MSVC	Single-threaded	10% - 20%	313.30s	426.57s
	Multi-threaded	80% - 100%	104.07s	398.83s

It is a fact that MATLAB has built-in optimizing libraries, which help to reduce a huge amount of executing time. Looking at *Data set 1*, the C/C++ code always shows 4 to 6 times slower than its original MATLAB code. However, when the data becomes more complex, C/C++ code, a programming language, surpasses the performance of the MATLAB code, a scripting language. The software performance after being generated could increase by 60 percent (in table 5.3) to 200 percent (in *Data set 3* of table 5.2).

With regard to multi-threaded C/C++ software, both table 5.2 and 5.3 indicates a significant drop in performance when multi-threading is activated. Although the software uses more system resources, the execution time is up to 14 times slower. That is a result of the generated code being not well-coded for multi-threading. An-

other aspect that can be observed is the performance consistency when compiling the code with MSVC with minimal difference between run time.

In summary, generated code demonstrates a significant jump in performance compared to MATLAB code. If programmers spend time optimizing the code, it could even improve with multi-threading activated. These factors could prove that the generated code is suitable for time-constrained applications.

5.2 Evaluation of software functionality

The testing code using CUTE-testing framework is simple to be integrated into the main functions. The test report then gives information about the software's functionality. Although MATLAB code and the generated one receive the inputs from the same source, there is a slight but noticeable difference between their outputs. Nevertheless, in the demo project, the error tolerance percentage shown in the table 5.4 can be accepted.

Table 5.4. *Example of error tolerance during testing*

Function	Output	Error tolerance
ComputeCamExtrinsics	`imagePoints`	2.5%
	`worldPoints`	1%
	`cam_extrinsics`	5%

Error handling is a significant drawback of generated code. Compared with the original MATLAB code, the generated one shows misbehaviors when an error occurs. Furthermore, the error, which can be truly simple, may put immense pressure on developing the error handling mechanism. Those simple errors are usually ignored during coding in MATLAB. For example, while concatenating arrays is relatively simple in MATLAB, in C++, concatenated elements must be checked for the same size before the operation is executed.

In summary, generated code demonstrated a remarkably good in ensuring software functionality. However, the testing is hardly sufficient to simulate all working cases of computer-vision software. MATLAB built-in error handling mechanisms are beneficial when they can be adapted to C/C++. Though, programmers must still be extremely careful in testing and developing the capability to handle error cases to ensure software safety.

5.3 Software development process model - Advantages and challenges

The software development process model proposed in this thesis, a type of *the iterative model*, shows some major advantages

1. Time and effort saving

   By using MATLAB Coder, programmers can save a tremendous amount of time developing the software when starting from scratch. Moreover, it takes advantage of the already-written MATLAB code as a model.

2. Suitable for the time-constrained projects

   Because the development process model is based on the iterative model, the products of every *iterator* phase can be taken into use as a complete and stable software without any dependency on the unfinished parts.

3. Software dependability can be ensured

   Because the software uses MATLAB code as its model, the process also take advantage of the MATLAB error handling mechanisms. It means that in lieu of trying to design a set of error cases and how to handle them, programmers can develop them based on MATLAB code.

4. Software quality is also ensured

   As MATLAB Coder provides the ability to make the code comply with software safety standards, it helps to reduce the time and effort of modifying the code to secure compliance. Moreover, the generated code is considerably fast compared to the original MATLAB code, which indicates the ability to apply to software that requires quick system response.

Nonetheless, to successfully use this development process model, there are some challenges or requirements.

1. MATLAB Coder limitations

   It should be noted that MATLAB Coder has limitations. For example, it cannot generate all MATLAB code and built-in functions. Therefore, before deciding to use MATLAB Coder, programmers must check for the ability to generate code of the built-in functions in the original MATLAB code.

2. Knowledge in MATLAB code is required.

   Programmers must have some knowledge of their original MATLAB code. Firstly, that code may require some modifications before it can be generated into C/C++.

Secondly, it would help design test cases and error handling mechanisms if programmers understand the potential issues of the software.

6. CONCLUSION

6.1 Thesis conclusion

It can be agreed that MATLAB is a common platform for software development, thanks to a wide range of applications supported by MATLAB, such as embedded projects, computer vision, and signal processing technologies. Programmers are suggested to use MATLAB Coder to save time and effort re-developing a software whose code is already written in MATLAB. MATLAB Coder is a powerful tool that provides the ability to generate the written MATLAB code into C/C++ code, which can operate in various environments. Unlike the developing process of ordinary C/C++ software, the MATLAB Coder approach possesses some unique features. Therefore, it is important to put this approach under investigation to figure out an effective process to develop quality and dependable software.

This thesis proposed a software development process model based on the iterative model associated with model-driven testing. The process strongly emphasizes developing in-depth software requirements and a complete test suite for each development phase. Because of some exclusive features of the process and heavy dependence on MATLAB Coder, there are a few challenges to consider, including MATLAB Coder's limitation and programmers' experience in MATLAB code and software testing. Despite the difficulties, the process has proven itself to be an efficient method that accomplishes the objective of developing dependable software in a short time.

Regarding the generated software, most testing reports indicated an acceptable precision in the generated software's outputs compared to the original MATLAB code. Thus, it can be said that software generated from MATLAB can ensure functionality correctness. Although the MATLAB code to handle error cases is not generated along with the functionality code, C/C++ software can still be dependable and compliant with MISRA C++ thanks to proper design and testing phases.

In conclusion, using MATLAB Coder gives exceptional advantages in short time development but still ensures software quality. However, there is immense potential for future improvement and development.

6.2 Recommendation for future work

The process for software development proposed in this thesis is applied to software that is already programmed with MATLAB language. It can be considered a standalone process model for future work for developing new software. It means that software can be developed from scratch by coding it in MATLAB and then generate it into C/C++. All developing phases, including MATLAB programming and C/C++ coding, share common requirements. This development model would allow maximum flexibility in coding software behaviors, whether they can be implemented in MATLAB or C/C++ after generation. Furthermore, the final product may greatly benefit from MATLAB and C/C++ strength, including error handling mechanisms and performance efficiency. An extra feature that can also be integrated is *GPU Coder*. It provides code optimization by taking advantage of thousands of cores in the graphic card, which is especially helpful for applications that involve signal processing or deep learning technologies.

REFERENCES

[1] Fabio, A. *KILLED BY A MACHINE: THE THERAC-25*. HACKADAY. Oct. 2015. URL: https://hackaday.com/2015/10/26/killed-by-a-machine-the-therac-25/.

[2] Srivastava, S. *OpenCV vs MATLAB: Which is best for successful computer vision project?* URL: https://www.analyticsinsight.net/opencv-vs-matlab-which-is-best-for-successful-computer-vision-project/. Analytics Insight. Hyderabad, 2020.

[3] Encyclopaedia Britannica, T. E. of. Automatic checkerboard detection for camera calibration using self-correlation. *software* (Jan. 2021). URL: https://www.britannica.com/technology/software.

[4] Humphrey, W. S. *The Software Quality Challenge*. The Software Engineering Institute. US, June 2008.

[5] cnn.com. *Software, hydraulics blamed in Osprey crash*. CNN. Apr. 2001. URL: http://edition.cnn.com/2001/US/04/05/arms.osprey.02/.

[6] Matteson, S. *Report: Software failure caused 1.7 trillion USD in financial losses in 2017*. Tech Republic. Jan. 2018. URL: https://www.techrepublic.com/article/report-software-failure-caused-1-7-trillion-in-financial-losses-in-2017/.

[7] IBISWorld. *Software Testing Services in the US - Market Size 2003–2026*. URL: https://www.ibisworld.com/industry-statistics/market-size/software-testing-services-united-states/. ibisworld.com. Dec. 2020.

[8] Rajesh Agarwal Pinakpani Nayak, M. M. *Virtual Quality Assurance Facilitation Model*. International Conference on Global Software Engineering. Aug. 2007.

[9] Galin, D. *Software Quality Assurance: From theory to implementation*. Vol. 1. 2004.

[10] Deepak Khazanchi, S. S. Assurance Services for Business-to-Business Electronic Commerce: A Framework and Implications. *Journal of the Association for Information Systems* (Jan. 2001).

[11] SoftwaretestingHelp.com. *Difference Between Quality Assurance And Quality Control (QA Vs QC)*. Apr. 2022. URL: https://www.softwaretestinghelp.com/quality-assurance-vs-quality-control/.

[12] JEVTIC, G. *What is SDLC? Phases of Software Development, Models, & Best Practices*. phoenixNAP. May 2019.

[13] Harvey, S. *What is a Secure Software Development Life Cycle*. KirkpatrickPrice. 2020. URL: https://kirkpatrickprice.com/blog/what-is-a-secure-software-development-life-cycle/.

[14] Martyniuk, J. *Software Development Lifecycle: Stages, Methodologies & Tools*. Dec. 2020. URL: https://devoxsoftware.com/blog/software-development-lifecycle/.

[15] SoftwaretestingHelp.com. *What Is Software Quality Assurance (SQA): A Guide For Beginners*. Apr. 2022. URL: https://www.softwaretestinghelp.com/software-quality-assurance/.

[16] Geekforgeeks. *Software Engineering | Software Design Process*. Dec. 2021. URL: https://www.geeksforgeeks.org/software-engineering-software-design-process.

[17] Wenzel, D. *Benefits Of Adopting Software Quality Assurance Early In Product Development*. SIGMADESIGN. Apr. 2019. URL: https://www.sigmadzn.com/2019/04/22/benefits-of-adopting-software-quality-assurance-early-in-product-development/.

[18] Martin, M. *SDLC (Software Development Life Cycle): What is, Phases & Models*. Guru99. Apr. 2022. URL: https://www.guru99.com/software-development-life-cycle-tutorial.html.

[19] Haraty, R. A. and Hu, G. *Software process models: a review and analysis*. 2018.

[20] Royce, W. *Managing The Development of Large Software Systems*. IEEE WESCON. Aug. 1970.

[21] Shiklo, B. *8 Software Development Models: Sliced, Diced and Organized in Charts*. ScienceSoft. Aug. 2019. URL: https://www.scnsoft.com/blog/software-development-models.

[22] Forsberg, K. and Mooz, H. The Relationship of System Engineering to the Project Cycle. *Engineering Management Journal* 4 (Apr. 2015).

[23] Lau, K. K., Taweel, F. M. and Tran, C. M. The W model for component-based software development. (2011).

[24] Gosling, J. and Bollella, G. The Real-Time Specification for Java. *Computer* 36.06 (June 2000), pp. 47–54. ISSN: 1558-0814. DOI: 10.1109/2.846318.

[25] Ibrahim, I. M., Nonyelum, O. F. and Saidu, I. R. Iterative And Incremental Development Analysis Study Of Vocational Career Information Systems. (Sept. 2020).

[26] Boehm, B. W. A spiral model of software development and enhancement. *Computer* 21.5 (1988).

[27] Boehm, B. *The Incremental Commitment Spiral Model*. Vol. 1. 2014.

[28] Beck, K. *Extreme Programming Explained*. Vol. 1. 1999.

[29] Shrivastava, A., Jaggi, I., Katoch, N., Deepali, G. and Gupta, S. A Systematic Review on Extreme Programming. *Journal of Physics: Conference Series* 1969 (July 2021).

[30] Umar, M. A. and Chen, Z. A Comparative Study Of Dynamic Software Testing Techniques. 12 (Jan. 2021).

[31] Nidhra, S. Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications* 2 (June 2012).

[32] Ehmer, M. and Khan, F. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. *International Journal of Advanced Computer Science and Applications* 3 (June 2012).

[33] Jamil, A., Arif, M., Abubakar, N. and Ahmad, A. Software Testing Techniques: A Literature Review. (Nov. 2016).

[34] Fairley, R. Tutorial: Static Analysis and Dynamic Testing of Computer Software. (1978).

[35] Standardization, I. O. for. *ISO 26262: Functional Safety Standard for Modern Road Vehicles*. 2018.

[36] Limited, M. *MISRA C++::2008 - Guidelines for the use of the C++ language in critical systems*. Vol. 1. 2008.

[37] Bose, S. *Coding Standards and Best Practices To Follow*. BrowserStack. Feb. 2021. URL: https://www.browserstack.com/guide/coding-standards-best-practices.

[38] SonarQube. *Code Quality and Code Security*. SonarQube. 2022. URL: https://www.sonarqube.org/.

[39] Software, P. *Static Code Analysis for C and C++*. Perforce Software. 2022. URL: https://www.perforce.com/products/helix-qac.

[40] LLC, G. S. *PC-lint Plus is a comprehensive static analysis solution for C and C++*. Gimpel Software LLC. 2022. URL: https://www.gimpel.com/.

[41] Parasoft. *A Unified, Fully Integrated Testing Solution for C/C++ Software Development*. Parasoft. 2022. URL: https://www.parasoft.com/products/parasoft-c-ctest/.

[42] Sharma, L. *Functional and Non Functional Testing*. Toolsqa. July 2021. URL: https://www.toolsqa.com/software-testing/functional-and-non-functional-testing/.

[43] Hamilton, T. *What is Dynamic Testing? Types, Techniques & Example*. Guru99. Feb. 2022. URL: https://www.guru99.com/dynamic-testing.html.

[44] Haytham Amairah Mike Jones, G. H. *Unit test basics*. Microsoft. Feb. 2022. URL: https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2022.

[45] Google. *GoogleTest*. June 2021. URL: https://google.github.io/googletest/.

[46] Lindholm, J. Model-based testing. (Nov. 2006).

[47] MathWorks. *Model-Based Testing*. MathWorks. URL: https://se.mathworks.com/discovery/model-based-testing.html.

[48] Hamilton, T. *Model Based Testing Tutorial: What is, Tools & Example*. Guru99. Mar. 2022. URL: https://www.guru99.com/model-based-testing-tutorial.html.

[49] Zendel, O., Murschitz, M., Humenberger, M. and Herzner, W. How Good Is My Test Data? Introducing Safety Analysis for Computer Vision. *International Journal of Computer Vision* 125 (Dec. 2017).

[50] Yan, Y., Yang, P. and Yan, L. Automatic checkerboard detection for camera calibration using self-correlation. *Computer* (May 2018).

[51] Moler, C. *A Brief History of MATLAB*. MathWorks. 2018. URL: https://se.mathworks.com/company/newsletters/articles/a-brief-history-of-matlab.html.

[52] Elsayed, A. and Yousef, W. *Matlab vs. OpenCV: A Comparative Study of Different Machine Learning Algorithms*. May 2019.

[53] MathWorks. *Pass Arguments to Shared C Library Functions*. MathWorks. 2021. URL: https://se.mathworks.com/help/matlab/matlab_external/passing-arguments-to-shared-library-functions.html.