Otto Helakorpi

ADVANTAGES OF MODEL-BASED DEVELOPMENT IN TERMS OF BUSINESS BENEFITS

Faculty of Information Technology and Communication Sciences
Bachelor’s Thesis
April 2022

ABSTRACT

Otto Helakorpi: Advantages of model-based development in terms of business benefits
Bachelor’s Thesis
Tampere University
Bachelor’s Programme in Science and Engineering
April 2022

Today’s industries require quick reactions and high-quality solutions. Businesses want to keep costs and resource misuse to a minimum for greater benefits. Automated processes can suit the efficiency needed in businesses. In the software field, this means quick development time and maintainability. Different methods have advantages towards these goals. One of these methods is model-based development.

This thesis analyses model-based development as the method emphasises automation efficiency and high quality. Model-based development is based on capturing required elements into a model. The model is processed with appropriate tools to form a wanted outcome. The tools tend to be highly automatic. The approach is applied in different development fields, more with practical simpler systems than complex software. The simpler ones usually have functional behaviour that can be modelled easier.

This thesis includes multiple advantages of a model-based approach that can answer the needs and requirements of industries, even overcome them. Advantages like reduced development time with high quality and model reusability can result in business-level benefits. To exploit the benefits in today’s state requires costs and effort as they are many times needed to be performed in a specific environment and with specific tools. The environment sets the need for specific knowledge of practises and effort in understanding the method to succeed. The challenges are needed to be overcome by more generally applicable solutions.

The analysing of the advantages in this thesis shows that there is no significant benefit that can be achieved through model-based development. The benefits are due to the overall advantageous nature of the approach. While the commercial usage of the model-based development is still narrow, there is a future for the methodology. Research regarding model-based development is refining the used concepts and providing new ways to exploit the advantages.

Keywords: model-based development, system engineering, embedded systems, software engineering, model-based software development, automation, business benefits

TIIVISTELMÄ

Otto Helakorpi: Mallipohjaisen kehityksen edut liiketoiminnan hyötyjen kannalta
Kandidaatintutkielma
Tampereen yliopisto
Kansainvälinen tekniikan kandidaatin tutkinto-ohjelma
Huhtikuu 2022

Nykypäivän toimialat vaativat nopeaa reagointia ja laadukkaita ratkaisuja. Yritykset haluavat pitää kustannukset ja resurssien väärinkäytön mahdollisimman pienenä saadakseen suuremman hyödyn. Automatisoidut prosessit voivat vastata yritysten tehokkuustarpeisiin. Ohjelmistoalalla tämä tarkoittaa nopeaa kehitysaikaa ja ylläpidettävyyttä. Eri menetelmillä on etuja tarvittavien tavoitteiden saavuttamiseksi. Yksi tällainen menetelmä on mallipohjainen kehitys.

Tässä työssä analysoidaan mallipohjaista kehitystä, koska se korostaa automaation tehokkuutta ja korkeaa laatua. Mallipohjainen kehitys perustuu tarvittavien elementtien mallintamiseen. Mallia käsitellään sopivilla työkaluilla halutun lopputuloksen saavuttamiseksi. Työkalut mallipohjaisessa kehityksessä ovat hyvin automaattisia. Menetelmää sovelletaan eri aloilla, enemmän yksinkertaisissa järjestelmissä kuin monimutkaisissa ohjelmistoissa. Yksinkertaisemman toteutuksen funktionaalisuus on yleensä helpompi mallintaa.

Työ sisältää useita mallipohjaisen lähestymistavan etuja, jotka pystyvät vastaamaan toimialojen tarpeita ja vaatimuksia, jopa ylittämään ne. Edut, kuten lyhyempi kehitysaika, korkea laatu ja mallien uudelleenkäytettävyys voivat johtaa liiketoimintahyötyihin. Etujen hyödyntäminen nykyisessä tilassa vaatii kuitenkin kustannuksia ja vaivaa, koska ne usein vaativat tietyn ympäristön ja tietyt työkalut. Erityistä käytännön tietämystä ja ymmärrystä menetelmästä tarvitaan. Yleisemmin soveltuvat ratkaisut pystyisivät selvittämään nykyisiä haasteita.

Työn analysointi mallipohjaisen kehityksen eduista osoittaa, että mallipohjaisella kehityksellä ei pysty saavuttamaan mitään tiettyä liiketoimintahyötyä. Hyöty mahdollisuudet tulevat mallipohjaisen kehityksen kokonaisvaltaisista etuuksista. Vaikka mallipohjaisen kehityksen kaupallinen käyttö on vielä suppeaa, menetelmällä on tulevaisuus. Mallipohjaisen kehityksen tutkimus parantaa käytettyjä konsepteja ja tarjoaa uusia tapoja hyödyntää etuja.

Avainsanat: mallipohjainen kehitys, järjestelmäsuunnittelu, sulautetut järjestelmät, ohjelmistosuunnittelu, mallipohjainen ohjelmistokehitys, automaatio, liiketoimintaedut

PREFACE

When I started my studies in Tampere University, I had little idea what the studies would include. Through sticks and stones, I still managed to maintain the required level to apply to my wanted master's degree program. In my favour, I got accepted into it. The thesis topic, originally suggested by my supervisor Kari Systä, ended up resembling more my interests than I first thought. It goes along with my ICT major, industrial management minor and future master's.

Learning curve during the whole thesis process was definitely not linear. There were some time-to-time challenges with defining the specification of the topic and its formulating. This kept the structure evolving too. In the end though, the most delighting is to see how your own understanding of the topic has developed to the point, it wants more. My topic regarded just brief analysis and ideas, whereas certainly more research can be done.

The process would not have been the same without the support I received. I would like to specially thank my friends and family who were along the whole time and helped with proofreading. I would also like to thank my supervisor Kari Systä for leading me to the right direction and keeping me in the path of high academic standards.

Tampere, 27 April 2022

Otto Helakorpi

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AADL	Architecture Analysis & Design Language
BOK	Body of Knowledge
EMF	Eclipse Modelling Framework
EPL	Eclipse Public Licence
GLP	General Public Licence
M2M	Model-to-Model
M2T	Model-to-Text
MARTE	Modeling and Analysis of Real-Time Embedded systems
MBD	Model-Based Development
MBE	Model-Based Engineering
MBT	Model-Based Testing
MBSE	Model-Based Software Engineering
SLR	Systematic Literature Review
SVA	System Verilog Assertions
SysML	System Modelling Language
UML	Unified Modeling Language
XMI	Extensive Markup Language Metadata Interface

1. INTRODUCTION

Today’s business environment requires responsiveness to changing situations and to high competition. Companies want to use more automated solutions because automated solutions offer several advantages over manual solutions. The biggest ones are lower time consumption, resource usage and risk for human error. Reliability improves forecasting and is crucial, in addition to security, in a digitalized world.

Demand for fast-paced software development is rising as software are being used more and more. Software development methods have been applied, along with other computer science solutions like cloud computing, machine learning and the Internet of Things, to answer the needs of businesses. Quality assurance and increasing complexity are the most important issues for software development and its supporting actions [1][2]. The data formats transferred and controlled in stages of the software development process can be difficult to understand in themselves. When creating code from them, errors occur easily. Assuring correct implementation can be challenging, affecting quality and maintainability.

One approach for efficient software development is model-based development. It is an activity chain of tools used for engineering purposes. The model-based development process is primarily used for the developing functioning systems or software. Instead of a heavy development process, the outcome is automatically produced from a designed model. It has been more heavily researched throughout the last two decades, while the idea of using models to handle software complexity has been around longer [3]. The main strengths of the model-based approach include fast development time and reliability, which, among other strengths, are analysed in this thesis. These can enhance the development process of software if they are defined in the right domain and complexity.

This thesis considers the advantages of the model-based development gathered from literature sources and research. The advantages are analysed from the perspective of their usage and applicability with tools – whether the usage is the full model-based development process, a part of it, or a specific strength of the model-based approach. The objective of this thesis is to get an overall understanding of the advantages of model-based development and find out where it has been applied on. While the theory is based on a few important model-based development areas, their characteristics and tools are studied to get a general understanding of the methodology. This makes it possible to recognize the broader advantages. The focus is not delved on specific software, tool, or system. The research question of the thesis can thus be formulated as:

What business benefits model-based development can provide and what kind of situations it could fit into?

The main aim of this thesis is to find out the advantages of the model-based approach in terms of business benefits – ways to realize competitive advantage. These advantages are created when the method can be exploited, or at least suitable, for maximum efficiency, thus assuring economic management. The visible commercial usage of model-based development on its own is still narrow. The aim of the thesis is not only to do a versatile analysis of how model-based development could provide economic benefits but also to understand factors limiting the usability of the method. In addition, the thesis examines the tools best suited for commercial use.

The analysis of this thesis is based on a few main literature sources. Database research and internet sources are used to support the content and concepts of the literature sources. Official pages of applications and original documents of organisations are utilized to get the most reliable and recent information. The databases of IEEE, ELSEVIER, ACM, and SPRINGER were sources of remarkable research.

The structure of the thesis aims to present the topic in a logical way. Chapter 2 goes through what the model-based development includes as a process and the theoretical background on how the process is structured within different domains. The main methods are also identified with their concepts and their place in the process structure. Chapter 3 states the main advantages of the model-based approach and considered them in terms of business benefits. Chapter 4 considers what are the issues considering the usage and summarises the factors of usage. Finally, the concepts and key points of the analysis are concluded and summarised in Chapter 5, which also presents the possibilities that model-based approach could provide for companies and indicates direction for future development of Model-based development.

2. MODEL-BASED DEVELOPMENT

Model-based development (MBD) is a development approach that uses modifiable, composable and formal model throughout its life cycle [3]. The goal of the approach is to make a development activity more systematic, repeatable, and success-oriented [4]. It relies on a computer, that is a set of software, to generate the necessary code from abstract artifacts of a model. Most of the decisions for the generation come from the design specifications of the source model [3]. This results in the need of careful consideration and preciseness in the modelling but also in more freedom to the developer. The development process needs to be within the boundaries of the modelled requirements. Specific tools perform the activities related to each task in the process. For most efficiency, the tools need to be compatible with each other and the model. In this chapter, the MBD process is analysed first with its stages and objectives, then specified in domains of a system and software. Domains here are groups of functional areas that have similar or same software requirements.

2.1 From model to development

Information of any system can be stored to a model. The information can be controlled to define a specific perspective, such as functioning or visual representation. The model can be an abstraction or a more concrete model depending on its characteristics and function, meaning it can be created to describe an existing or a non-existing system. Typical types of models are informative models, behaviour models and structure models [4]. The purpose of a model can be to make data more understandable for humans or to present data in a form that can be analysed better mechanically [3].

Model-based means the model and its handling are the fundamentals for the activity. The model should hold all the necessary data to proceed in such an activity. Requirements are captured into the model to suit the wanted implementation. Model-based activities use abstractions to cover the complexities of environments. They produce fast-paced innovation and comprehensibility as the model works as the mutual language for all stakeholders. [5]

For the development of a software, the model-based approach starts in a company by realising the need for the system or software. When MBD is the chosen method, there comes a need to design a model for the software. The model needs to cover the set requirements for the product and the wanted working environment. The engineering process for handing the model can be split into specification of the model and the integration of the designed model. The model life cycle goes closely along a normal product's life cycle V model where testing connects the stages [6].

The figure 1 presents the stages in the model life cycle and connects the corresponding parts of the specification and integration processes by aligning them side by side. Specification of the design starts by defining the wanted outcome – the product. The next stage is defining the operation environment, whether it is a system or a complete software. Different environments or domains have their own purposes and characteristics. The third stage is where the needed tools are defined to suit the decisions made in the earlier stages. Then the definitions are put to action, and finally the elements of the model are integrated to try to satisfy the set requirements. [6] The created model must be analysed to verify the wanted attributes [4]. The attributes include correctness, consistency, traceability, completeness, and interaction analysis.

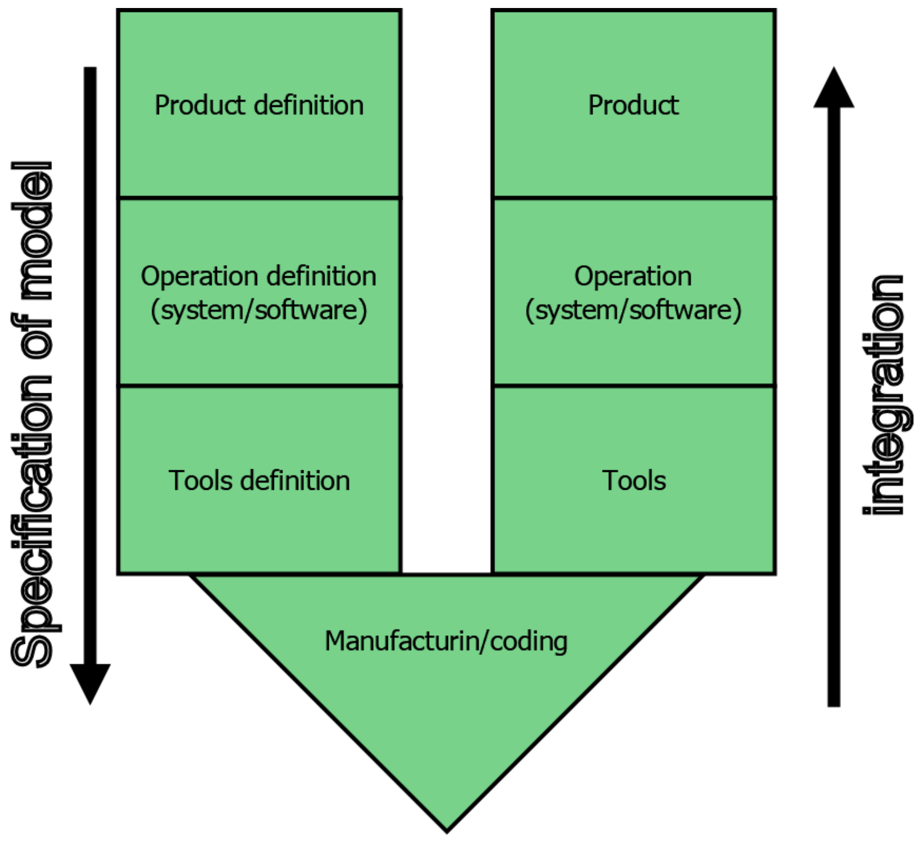

Figure 1. *Product life cycle in MBD. Adapted from [6].*

MBD can be seen to branch to Model-based Engineering (MBE) and Model-based Architecture (MBA) which are approaches with different abstraction levels. In MBE the development process is focused on the domain model creation and exploitation, rather than on computing. The purpose of MDE is to maximise compatibility between systems, simplifying the design process and promoting communication with parties through standardization and practises for increased productivity. In MBA more practical methods are used and only significant executions are conducted. [3]

2.2 MBD for systems

The usage of MBD has been analysed through the years and it has provided significant opportunities and measures for new approaches for developing systems. The systems can be embedded systems or operating systems – components of a larger systems. They are used to monitor or control the larger whole by special hardware devices [7]. System's software is usually made to serve a dedicated function [8].

The MBD process for systems includes three main steps: making the model, transforming the model and simulation [9]. The steps are usual with model-based approaches. In some cases, code generation and verification are seen as their own steps, but many times they are included as necessary functions in the three mentioned main steps. The designed model itself is based on set requirements. The tools for transformation and generation are chosen to best suit those requirements. Other steps of the process are highly dependent of the modelling. Transformation is done to get the source model to the target domain and ready to be executed in a simulation environment [9]. To satisfy all requirements after the transform, the transformed data needs to be verified. Verification is needed to ensure that the wanted information can be obtained from the given model [9]. Many times, the model must go through several modifications before passing the verification step. The verification step creates "a loop" to the process, as can be seen in Figure 2. Each loop has its costs in time and resources. The easy editability and quick testing is essential here. With correctly transformed data, source code can be automatically generated for simulation. Simulation is strongly dependent on the nature of the source code, as it aims to get an understanding of the system's capabilities and validation of its functions [9].

The process of MBD used in systems is represented in Figure 2 to support the data. The figure is simplified to not include unnecessary specifics of the activities to illustrate the flow of the activities more efficiently. A more detailed view of the activities with their common tools is given in the following section.

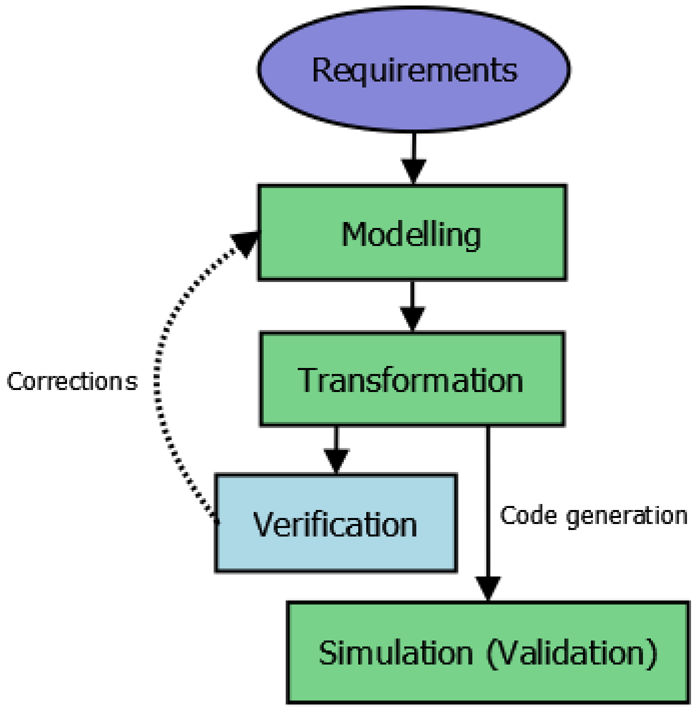

Figure 2. *Flow of the Model-based development process for systems. Adapted from [9].*

While the process is not perfect yet, it is secure and progress has been made through different orchestrations of tools. Problems commonly result from complexities and behavioural aspects of the given model that are challenging to verify and simulate. The aim of MBD system engineering is to provide enhanced efficiency and quicker development time for complex systems. [7]

2.3 Tools and methods in MBD

Different tools are needed to perform specific actions to move forward in the MBD process. In this context, the tools mean the chosen implementing methods in each activity of the development process. Tools are usually third-party applications. Tools are often specific to a working environment. Picking compatible tools for every step may be challenging depending on the specifications of the model.

In software engineering domain there are not so generally applied tools, while there are plenty in embedded systems engineering. For embedded systems the common tools are presented more closely in the following sections. The sections aim to present the main objectives of the MBD activities in the tool chain. While the tools are not necessarily the same for systems and software, the objectives and factors of the different stages are similar.

For systems that are often complex, like embedded systems, the modelling can turn out to be challenging due to the behavioural aspects. The same goes with complex software. Too developed behaviour of functions is hard to verify and validate properly. [9] While the tools are always determined by the methodology used for developing the model, they still affect the requirements and therefore should be considered when planning the approach for modelling, transformation, and simulation. The right tools provide the desired efficiency and reliable outcome.

The tools can be categorised based on characteristic suitability to environments. Compatibility with other tools is an important factor too. Some tools' characteristics can be suitable for a variety of MBD activities. The characteristics of a tool can be seen to be activity-specific or general. Activity-specific characteristics are related to work on a certain step of MBD like modelling, transformation, or simulation, while general characteristics are not. Characteristics that are specific to an activity closely depend on the activity they are included with. [9] The affecting parameters for each activity are listed in Table 1.

Table 1. *Activity specific characteristics of MBD activities [9].*

MBD activity	**Parameters**
Modelling	Support of Extensive Markup Language Metadata Interface (XMI) format, Language profile support
Model transformation	Transformation type
Verification	Verification approach
Simulation	Integration capability

Some general characteristics of tools are its licence type, customizability, and supported platforms. Licence type comes from the developer or the service provider. Windows and Linux can be seen as the most dominant platforms as they are the most supported. [9]

2.3.1 Modelling

One of the most important parts of MBD is to get the requirements modelled the correct way. Methodologies, or languages, to capture requirements into a model include for example the unified modelling language (UML). UML has its own modelling profiles called

System Modelling Language (SysML) and Modelling and Analysis of Real-time Embedded systems (MARTE). UML is a used standard modelling language in software engineering field [7]. Some commonly used tools for modelling are MATLAB, Magic Draw, Modelio Editor, IBM Rhapsody, PapyrusMDT, and Artisan studio [7][9]. In the system domain, the SysML profile is the most frequently used tool for modelling requirements as it has the widest capabilities and can alone provide the most elements in this domain. [9]

The Extensive Markup Language Metadata Interface (XMI) ensures non problematic data exchange through standardised mechanisms between tools. Efficient and secure data exchange is a basic requirement in most cases [10]. UML and its profiles are supported widely, making it possible to use different combinations of the profiles together. Through combinations it’s possible to serve more complex structural and behaviour aspects, which would not be possible with a single profile. There are many tools available, like OBEO SysML Designer and TimeSquare MARTE Tool, to model SysML and MARTE individually for a specific modelling context.

2.3.2 Transformation

Transforming the model is a centric element of MBD. It converts the source model to software artefacts automatically. Model transformation optimizes and refactors to achieve the target implementation of the model. If a model needs to be modified, the transformation can be simply repeated. Important in the transformation activity is to set up the format to work the best in the oncoming steps while forwarding the quality through verification. Customizability of tools is typical in the transformation step as it is the middle part entwining the modelled requirements and the simulation environment. [11]

There are two common types of transformations, model-to-model (M2M) and model-to-text (M2T) transformations. The M2M approach is used slightly more in research as it results in a more accurate model transformation, while M2T results in a highly customizable model transformation. Usage of both together is therefore common to get the most accuracy and flexibility towards the target domain. For example, MediniQVT can be used for M2M, Xpand for M2T and MDworkbench for both types of transformations. [9] Eclipse foundation also has multiple tools regarding model transformation to fit each type. Most of the transformation tools are licenced under Eclipse Public Licence (EPL) which is an open-source licence maintained and updated by Eclipse Foundation making it offer freedom within modern boundaries and be competitive with other licences [12].

2.3.3 Verification

Verification is an activity ensuring correctness. It maintains the development process and tries to increase its preciseness. Verification is closely connected to the other activities. It usually happens through general standardised model verification tools. Verification tools try to find incorrectness, for example, from the transformation of the model. The main tools used in various research are PRISM and Zot tool. These tools are also highly customizable to correspond the verification requirements and are therefore used widely in research and applications [9]. The licence type for PRISM and Zot is a General Public Licence (GLP). With GLP every user can distribute, copy, and use the source code of the tool [13]. This licence type goes highly along with the required usability and flexibility. PRISM is based on probabilistic models so it can be used in a variety of applications and properties, especially to support automation, as it can hold uncertainties [14]. Zot is a model checker that can support multiple logic languages and even different encodings. Therefore, Zot can be customized to suit real-time system specifications. [15] There are other options too, but they lack customizability and thus limit the functionality.

2.3.4 Simulation

Simulation tools are slightly different from tools in the other steps, as they are not easily customizable, and most times have their own model development environment [9]. This makes it hard to integrate other tools and features into the working environment. Commonly utilised available tools for simulation are for example SIMULINK, ActiveHDL, QEMU and OpenModelica. Selecting of the suitable simulation tool is completely dependent on the particular requirements of the simulation and therefore the selection of the tool is the most important activity of simulation. It needs to fit and be compatible with the automatically generated source code language for most accurate validation [9].

2.3.5 Frameworks

Not only tool can be multi-functional, but tools can also be set to form a framework. A framework, or set of tools, is capable of performing multiple MBD actions. This is useful when working with similar environments that can be supported with the same set of tools. For embedded systems common toolsets are Eclipse Modeling Framework (EMF), Topcased, Gaspard2 and TTool. EMF is Eclipse Foundations project to support multiple tools and methods for the development of a variety of applications. It’s highly used, and a lot of tools, like Topcased and Gaspard2, use EMF as a base. Topcased and Gaspard2 are toolsets that are able to develop an embedded system from start to end using the model-based approach. [9]

2.4 Model-based Software engineering

Model-Based Software Engineering (MBSE) is an accepted software engineering methodology in software engineering community, and its industrial use keeps growing [4]. MBSE is specified implementation of MBD activities. It aims to include commonly used software engineering methods to MBD. MBSE can be an alternative to traditional methods with high quality and reduced costs. Although, its more widely agreed upon core concepts and practises still need defining. MBSE follows a process similar to the MBD. The difference between MBSE compared to MBD is the software engineering domain. The model and its transformation are the basic elements for software development. Modelling happens through a domain-specific language. The concepts and requirements are tailored to present specific characteristics of the software domain. [3] Software in this sense is a digital application that has its purpose of usage and can work independently.

As MBSE lacks general set of concepts and practises, compiling of Body of knowledge (BoK) regarding MBSE started at 2018. The Bok is now in process of gathering necessary information from professionals and practitioners regarding the main set of content, tools and practises of MBSE. The aim of the BoK is to get globally accepted, consistent, view of MBSE that can be implemented in professional and educational domain. [4]

The stages of the process are similar to the stages of the MBD process for systems, and many practises work for both. For example, verification and testing are important elements between the stages of development. The objective of a developed software is to work on its own, commonly including multiple dimensions in functionality. The dimensions complicate the requirements of the model. Therefore, specific characteristics of multifunctioning software are needed to be taken into account in the modelling. A software's model usually includes aspects like structure, behaviour, and non-functional properties. These dimensions are needed to describe the complex objective of the software.

Creating software with MBSE can be split to two processes, domain engineering and application. The domain engineering process includes the development and organization of the reusable component and designing the structure to cover the requirements of the domain. It has three primary steps that are domain analysis, domain design and domain implementation. Domain engineering can work as a basis for reusable product quality tracing system. The application process includes the software domains and the management required for software engineering. [3] The model and the product are customized for the aimed development environment and platform of usage. MBSE includes the defining of the requirements for the software overall, analysing them through measuring, verifying, and validating [16].

Preciseness and completeness of the product depend on the same aspects of the model. And as software has to function correctly, preciseness and completeness are in a very important role. This objective requires model-implementation mapping that needs to be efficient. Management of consistency between generated code and model is required. The outcome is usually a simulated to make it able to validate the correctness. When validated to satisfy being complete, it is reasonable to proceed to applying into target environment or intended usage or the software.

MBSE has been applied widely in engineering field. Especially with activities like DevOps and robotics that have similar charasteristics like automaticity, efficient testing, and maintainability [17][18]. MBSE is also applied with cloud computing, Internet of Things (IoT), testing and cyber physical systems [6][19][20][21]. Techniques used with MBSE include domain specificity, metamodeling, iterative transformation and quality assurance [3][22]. To capture and support needed performance and safety, standardized Architecture Analysis and Design Language (AADL) can be used. It provides possibilities for designers to assess non-functional properties and assures partial correctness of functionality through project-specific analysis of structural properties [17].

3. ADVANTAGES OF MBD

The possibilities of MBD and its aim to create efficiency in development have been acknowledged for a while now. Still, the notable usage is low, making it uncertain if the advantages of the method are recognised enough in the industry. [7] The frequently occurring research suggests that there still is usage and interest in the method. Under the “curtain” it is more accepted, and different field companies’ workers have had good experiences with it [7]. Today’s usage of MBD activities relies on supporting other activities of a company. This chapter names the advantages that are acknowledged in research and tries to provide solutions in the sense of business benefits through them.

3.1 Advantages of MBD

Research has found positive effects of MBD from motivations to use it. These include the reusability, development time, quality, reliability, maintainability, cost, and traceability. In addition, variable positive effect has been seen in activities where automation is seen important for efficiency like code generation, simulation and documentation. [7] The effects, positive and negative, are also dependant on the way of usage and implementation. MBD can be tailored for a customer, so it possibly answers the needs and requirements accordingly [7]. The positive effects are advantages and beneficial in business sense when they can be exploited to overcome existing methodology. Competitive advantage in business can be reduction in costs or increase in benefits [23].

One noticeable advantage of MBD is the methods support on efficient testing. Efficient testing plays a key part when wanting to keep costs minimum while driving for high quality. Through careful testing meaningful savings can be made in the long run, while the testing process itself might be costly, prompting the need for efficiency. Due to these reasons efficient testing can be seen as a business benefit. Testing based on model-based approach has been applied on multiple research [20][24]. Testing allows the quality of the software to improve [16].

Automating software code generations can be developed with help of modelling tools through UML and other standards. For example, a profile of UML called UML 2.0 Testing Profile brings designers and testers closer by using multiple functionalities of UML like system modelling and test specification. Testing is possible to start in the early stage of development with reusable UML-designs [16]. The usage of the testing strengths has formed its own methodology, namely model-based testing (MBT). Aim of MBT is to create automatically derived and efficient tests. The failure-detecting has been developed to a point where it is seen highly positive in terms of many research [20][24]. The strength of MBT has been applied to create a tool called MobiGuitar [24]. MobiGuitar automates Android app testing. Results with MobiGuitar show that it can detect relevant and serious bugs.

MBD suits fast paced development, where results can be obtained quicker. Problems like changed requirements which require redesign can be reacted on quicker due to improved understandability of the model. Same goes with responding to obsolescence and correcting latent design defects. Reacting quick can reduce costs or increase benefits in the product life cycle, making it an attribute for business benefits too. [23]

Research of MBD aims to create more efficient and secure mechanisms to provide MBD solutions [19][22]. The solutions are wanted to work better in commercial use, as in work their way as a possible way to get business advantages. The essential advantages of MBD are found from research. In 2016 Grischa Leibel's group [7] conducted research about the usage of model-based approach practices in relative industries and more deeply about tools used in modelling. The study had 113 participants in a survey, mainly from the automotive industry. Notable amounts were also from avionics, health care, defence, and rail industries. 48 of the participants stated that they have over 3 years of experience in MBD, 40 participants had moderate experience and 25 were new users of the method. The research stated that the majority of the participants had a positive attitude towards MBD. Within the industries, participants were mainly from software implementation, architecture responsibility, testing, designing, requirement specification and managerial fields. Most used tools for modelling were answered to be MATLAB and Eclipse-based tools. In the majority of answers UML and SysML were used as modelling languages. The main reason for model-based approach usage was answered to be the need to improve reusability and need for quicker development time. Needs for improved quality, reliability and maintainability with cost savings and traceability were significant motives too. They plan to use the gathered information as basis for their future research. The questions in the survey are needed to be fixed to gather even more relevant information in embedded system domain and to validate the now identified results. [7] Difference in domain, company size and position in company's value chain had little effect to the results, although they are normally a meaningful part when evaluating the business perspective. The fact that workers were still familiar with the approach is the attribute lowering the boundary to exploit the advantages of MBD.

Complete toolsets for applying MBD for embedded systems have been developed [9]. The toolsets are stated in chapter 2. These toolsets can work as an effective way to develop systems. The toolsets still hold some limitations, mainly as they are sourced from EMF. To pass these limits, open-source tools have been introduced in form of a toolset called PolarSys. Open source provides possibility to tailoring project-specifically and high freedom to modifying the development into suitable form. This has affected in transition to more model-based solutions from document-based practises in system engineering field [25].

3.2 Advantages of MBSE

MBSE can be used to determine software requirements in comprehensible, structured, and formal models [16]. Trending with MBSE has been solving problems with implementation-stage models to generate executable code. With MBSE it is possible to provide applications to the business industry in forms of applications suitable for business process modelling, widely distributed repositories, and architectural models [3].

An advantage with software nature is that it is in digital form for its whole product lifecycle. Digital form provides access to easy editability in each stage of the development. This is crucial especially after providing the software to the customer or on its target usage. The adaptation with software to new situations can arise to save in costs. And as the reusable model is used, the traceability of issues is more convenient helping on the maintainability. This also benefits the reliability as the reliability is considered and integrated into design on a real-time basis. [23]

A model of a software is easier to understand than its code. This makes the validation of the correctness of a model easier too. It is important to verify correctness in every stage to mitigate the risk of a major problem occurring, like a bug or such, in a later stage, which could cause an unwanted amount of effort to a company. The code is generated by tools, which ensures that the code is not altered after it’s obtained. [16] This creates security, transparency, and trustworthiness to the process that are important factors for companies.

The possibility for early testing help reduce costs in MBSE too. Time needed for the software development process is reduced by systematic designs and reusable models. With successful MBSE, manual back-end coding and developing might be completely unnecessary. This lowers the threshold of efficient testing and modification straight from the start of the development process. The possibility for human error is also reduced. MBSE is therefore able to provide practical, efficient, and functional solutions for software design and testing [16]. MBSE provides wider possibilities for models and the requirements can be managed more easily compared to the more traditional approach for embedded systems that is aimed for more practical solutions. These all are factors for gaining a competitive advantage in business.

The most limiting factor for software is complexity. There has been a success with simpler functionalities. For example, opportunity with MBSE is utilized with blockchain smart contracts. Blockchain technology is a modern, safe, and transparent technology for global companies to conduct information. Although in the development of blockchain, errors are highly unwanted as they leave a mark on the blockchain forever. Blockchain is based on encryption like cryptocurrencies, and the technology is trending. Smart contracts are software and as a concept enable, without intermediaries, efficient execution of agreements [16]. These kinds of agreements are important to create trust between the contract parties, as the blockchain data is immutable. Creating the required data is heavy and consists of a systematic mechanism to collect specifications and requirements with verifying and validating before deployment to the blockchain platform. MBSE could answer the demand of the trend while traditional software development cannot. With the MBSE the errors in design and code can be found as there is verification and validation from the start. More focus can be put on identifying the absences and inadequacies of requirements. MBSE can create the required reliability efficiently and through automatic code creation the desired quality. [16] So MBSE can be beneficial to companies if the need is for the right type of software.

3.3 Benefits in economy strategies

Strategic choices at the business level define practices of the development process. These include cost leadership, product differentiation, collaborations, and flexibility. Business strategies are ways to achieve competitive advantage [26]. A new method, that is not used widely in an industry, can be a way to enter the industry. It can also provide benefits by having efficiency over others. For the same outcome the new method can work as a substitute for an existing method. In substituting the customer needs are satisfied, but in a different way. If substitution is superior to previous means, it can ultimately replace an industry’s products and services. Imitation occurs also in form of direct duplication where the same functionality is produced with lower price creating cost leadership position [26]. MBSE has the characteristic in right environment to do both, substitute and provide a product with lower cost. If the reusable model can be imitated, it can also work as an opportunity to implement it further with lower cost.

Another way to achieve business benefits is vertical integration. Vertical integration is a corporate strategy where company owns major parts of its production process. [26] Here vertical integration could be developing and maintaining a framework internally in a company. Therefore, the whole process can be controlled and managed efficiently, and opponents cannot imitate the development easily. Reusability of models is also essential in this case as with broad and precise models, great amount of efficiency can be exploited. Having these valuable resources that are costly to imitate are important in competitive situation and work as foundation for economic benefits [26].

Appropriate cooperation can be seen as a valuable asset in a company and can result in benefits in itself. Cooperation with other companies can work as a way to achieve competitive advantage. [26] For example, if one company has some set of tools but still lack in some parts of the MBD process. Some other company then has the missing tools, and the tools are not outsourced. They can then work in cooperative manner to create benefit to each other by complementing the creation of a complete activity chain.

3.4 Evaluation of the benefit analysis

Compiled overview of the advantages can be found from Table 2. In the table the advantages are stated as attributes and their ways to acquire possible business benefits are set from the analysis. While there are more minor advantages and supports, only the necessary advantages are presented in the Table 2.

Table 2. *Overview of the advantages with their benefits.*

Advantage	**Advantage type (O=overall S=system SW=software)**	**Ways to acquire business benefits**				
		Increasing development benefits	Lowering development costs	Business-level strategy (cost leadership, product differentiation and etc.)	Corporate-level strategy (Vertical integration)	Increasing value of the product (e.g. efficiency, customer perception)
Reusability	O	+	+	+	+	+
Development time	O	+	+	+		

Resource usage	O		+	+		
Quality	O	+				+
Reliability	O		+			
Maintainability	SW		+		+	
Editability	O/SW	+	+			+
Traceability	SW		+			
Efficient testing	O/SW	+	+			
New method	O/SW			+	+	+
Complete Toolset	S		+			+

As seen in Table 2 the advantages are broad and include multiple possibilities to get a competitive advantage. The distribution is fragmented meaning there isn’t a significant way gaining benefits over another, which goes along with research results [25]. This analysis doesn’t still take the challenges and shortcomings of MBD into account, and thus the actual distribution can differ from this. This still gives direction on the capabilities of the model-based approach.

4. CHALLENGES OF MBD

Usually the challenges in software development arise from shortcomings and limitations. With MBD the most noted are the interoperability problems between tools, usability and the training of developers requires effort. Realizing the advantages requires consumptive activities. [7] Even with the little usage of MBD in industries, some companies have not used the method long, meaning the effects of advantages and disadvantages can still be unmeasurable or unreliable [7]. So, there is no empirical evidence supporting the benefits of MBD, while there are numerous claims of the benefits in literature and research. There also isn't any significant one benefit claimed. This doesn't still imply that MBD is disadvantageous. [25]

To make MBD more beneficial, interoperability between tools should be improved for commercial tools. Commercial tools are licenced, which when used creates licence costs. This means that they can be non-beneficial to the business if the licence costs are higher than the benefits that arise from the tools. As tools are commonly provided by a third party, they might cause support and maintenance issues as well. [7] The nature of the situation motivates the usage of open-source options and companies' own internally developed tools. But many times, the company's own tools might lack generally applicable functionality. Also, non-functional requirements are still an open issue with MBD. These requirements include security, usability, and performance [20].

4.1 MBD challenges with systems

Most common shortcomings with systems arise from the applicability of tools and their issues [9]. Tools cannot always handle the functional behaviour of a complex system. This usually arises from the interoperability between the tools and tool inflexibility. A poor modelling method or language can also be a problem in handling complexity.

With some UML models, different combinations of XMI and UML versions can be used sourcing challenges in the modelling of systems. This with other reported issues regarding XMI can result in combinations which are not supported by every tool and therefore can also cause in some troubles with the data exchange [9]. An affecting factor can also be that XMI has not been updated since 2015 [10].

Sometimes the process of testing can be costly for embedded systems meaning its either time consuming or heavy as a process. Such process should be kept in the necessary amount of testing to avoid extra costs. Minimising the loops going through the verification stage is important. To avoid unnecessary testing and reproduction, simulation environment can be utilized. With simulation tools the challenges with inflexibility and boundaries create difficulties to the selection process. That's why sometimes when it fits the situation better, a simulation tool is specifically developed for the occasion [9].

With more complex product the connection with the relevant stages in the development process is clear but many times the specifications can only be verified in the end when the product is ready. This happens due to the engineering activities in the complex process being connected by time series and requiring sufficient conditions for the life cycle. [6]

4.2 Challenges of MBSE

Implementation of MBSE is a fairly new methodology. A lot of aspects and possibilities are needed to be researched more properly. Wide business needs and usage of software require more and more combinations with requirements and orchestration of tools. Tools are also continuously developed to suit a wider range in usability and combability. Trends in business and research affect the usage a lot. The challenge with software then compared to the system approach is that in software development the stages of the process are not so connected. Every stage most of the time requires its own procedures, boundaries, and relevant concepts in a suitable level of abstraction. [3][25]

Research has been about whether MBSE can survive in the software engineering field as if the implementing of the process exceeds the cost that would come from manually developing the code. MBSE requires multi-aspect modelling and therefore can create challenges in it, making it costly in effort. As with software too, the challenge comes with the behaviour aspect as it can be hard to compile with a premade model and remarkable success with it has not been achieved. Just few tools perform well enough for software development, but none can completely provide exact the form for MBSE. Another challenge occurs with non-functional properties as they need to stay simpler compared to the application itself and they require the software model to contain enough details for satisfactory completeness. In cases with limitations of tools it can be more efficient to directly write the code. [3]

Many existing modelling techniques assume that the software models are artifacts of documentation that are peripheral to code development [3]. Although, this many times isn't enough for complexity of a multi-aspect software. Modelling of behaviour include methods that are based on formal notations and methods that are more informal but practical. The formal methods contain for example the use of process algebras. They are tools for automatic analysis but have limited expressiveness making them more rarely used. The informal methods can include usage of UML profiles. Structure wise UML profiles fall under the assumption and are overall more suitable for informal approaches. In informal method UML can be in a form of state, activity, or interaction diagram. They cannot model the behavioural aspect alone due to their incompleteness of properties. Diagrams are commonly used in system comprehension and communication. Depending on the nature of the development operation and on significance of executions, these informal methods can be extended to work. [3]

MBSE tries to supply appropriate interconnection between the stages of the development by models that are concise and expressive throughout the process [3]. This unites the relevant concepts and enhances transparency by making it able to use the concepts in other areas too. The benefits of the model-based approach for software engineering are therefore only usable when the complexity is considered properly. That defines the requirements for the environment to be suitable and in the right domain for the development process. If not considered enough it can affect the process negatively and make it inefficient. It can lead even to its failure, meaning unnecessary costs in resources. Companies many times have a strict budget which is needed to be obeyed.

4.3 Summary

To summarise, the challenges with the model-based approach come from tools used and their incompatibility. Important in implementing MBD is to use suitable modelling language. The language should be able to establish fundamental relationship between design, physical and logical for the designer [23]. The effectiveness of model-based approach is also dependant on its conditions. Conditions like organisational structure and qualification of the working team [20]. These are factors for defining the nature of requirements that the whole development process is dependent on. Well put implementation can itself increase the effectiveness of the approach. As in theory the model-based approach can provide a highly automated process and still have high quality outcome. It ultimately aims to get design right the first time resulting in reduced costs and timescales. If considering the method working correctly, the only affecting factors come from the environment. But the lack in the core set of concepts and procedures makes it difficult to have enough understanding on implementation of the approach, which might be the result in the low commercial usage. There are not enough commercially viable tools to support the methodology.

The research trend and papers utilising the model-driven solutions has been rising throughout the years. The research considered is from IEEE, Springer, Elsevier and ACM which are significant professional science and technical organisations and societies. For example, in 2012 there were three times more relevant research regarding MBD comparing to the last year. Every year the change has not been that radical or might have been even less than last year – the growth happens in phases [9]. Research is commonly executed through systematic literature review (SLR). One SLR method is the Kitchenham's method which is widely used in software engineering field [16]. It includes three phases planning, conducting, and reporting. Other data gathering methods included survey and conducting a body of a knowledge [4][7]. These make it possible to find lower boundary understanding of MBD usage.

The aim of such studies is to have high quality and trustworthy outcome through compiling wide range of expert knowledge [16]. With versatile analysis it's possible to understand the advantages in the situations MBD is convenient to implement. From the advantages, the MBD can be seen to provide a wanted outcome and therefore it has wide possibilities to provide business benefits. The open issues still limit the usage, and the economic benefits can only be analysed through the advantages and shortcomings.

5. CONCLUSIONS

MBD is process emphasizing efficiency and quality. It is a highly automated process yielding advantages in the methodology of the development. The advantages and issues can be seen to come from the attributes of the method and from the operating domain. Still, low usability limits the knowledge regarding the advantages and shortcomings too [7]. Domains inspected in this thesis included system engineering and software engineering.

There has been done more research regarding MBD done in system domain. This implies that the method is now at least more suitable for system engineering. This is due their straighter forward functionality requiring less dimension in the model. Embedded and collaborative systems still have behavioural aspects that create challenges to the requirement defining and modelling.

Like other software development methods, MBSE is constantly developing toward more generally applicable solutions. Still, general framework to handle the process completely is missing. And as it seems from the studies, there would be a need for that. Such framework requires a lot of effort and defining to work properly. The construction of the solutions is hard too due the demand for more complexity also evolves and modern world is in constant change. A lot of the research consider on their own specific topic, and the MBSE is just on side role and its activities and usability are considered and applicated to fit the needs of the specific study. This makes the connection between the research challenging to compile. Overcoming of this challenge became in 2018 when a conduction of BoK started. It aims to provide globally accepted set of concepts and procedures with appropriate tools.

Through today’s state there are good arguments to see that model-based approach has more positive attributes and advantages than shortcomings. The advantageous nature can be seen as the factor why the industrial usage of MBD concepts and practises is growing [4]. But at the moment, the usage can be seen to have large costs to implement as it requires special environment. Visible commercial usage benefits of MBD are still missing making it hard to fully convince economical management [25]. Issues with the method can work as opportunities and show direction for future research.

The advantages of model-based approach that this thesis analysed can be factors in benefitting with business-level strategies. These strategies are the measures of how a company can define its mission and achieve benefits in industries [26]. Some advantages and disadvantages seem to have same effect independent of economy of scale. Different company size, domain and value chain seem to have no effect [7]. Interdisciplinary designing is the future of industries, with its possibility to provide solutions to modern problems [27]. MBD suits, and emphasizes, interdisciplinary solutions. It requires knowledge on human perception, different engineering fields, designing, manufacturing, and management. The approach values cooperation that can have its benefits.

While this thesis was able to find business benefits from the advantages of MBD, there are still a lot of limiting factors. Research regarding the model-based approach is actively seeking new ways to exploit the advantages of the method and conducting more generally applicable solutions. The results of this thesis should be specified and updated when research develops the method and the usage is broader.

REFERENCES

[1] Felderer, M. & Ramler, R. 2021. *Quality Assurance for AI-Based Systems: Overview and Challenges (Introduction to Interactive Session).* Lecture Notes in Business Information Processing. Vol.404, pp. 33–42.

[2] Vallon, R. et al. 2018. *Systematic literature review on agile practices in global software development*. Information and software technology. Vol.96, pp. 161–180.

[3] Md, N., Basha, J., Moiz, S. and Rizwanullah, M. (n.d.*). Model Based Software Development: Issues & Challenges.* [online] Available at: https://arxiv.org/pdf/1203.1314.pdf [Accessed 2 Apr. 2022].

[4] Burgueño, L. et al. 2019 *Contents for a Model-Based Software Engineering Body of Knowledge*. MOMENTUM: analysis of models towards compilation to predictable embedded real-time and safety-critical applications. Vol. 18 (6), pp. 3193–3205.

[5] ICT Group. (n.d.). *What is the difference between Model Based Testing and Model Driven Engineering?* [online] Available at: https://www.ict.eu/en/projects/what-difference-between-model-based-testing-and-model-driven-engineering [Accessed 3 May 2022].

[6] YU, C. et al. 2021*. Industrial Design and Development Software System Architecture Based on Model-Based Systems Engineering and Cloud Computing*. Annual reviews in control. Vol. 51, pp. 401–423.

[7] Liebel, G. et al. 2016. *Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice.* Software and systems modeling. Vol. 17 (1), pp. 91–113.

[8] www.heavy.ai. (n.d.). *What is an Embedded System? Definition and FAQs | HEAVY.AI.* [online] Available at: https://www.heavy.ai/technical-glossary/embedded-systems [Accessed 3 Apr. 2022].

[9] Muhammad Rashid, Muhammad Waseem Anwar, Aamir M. Khan. 2015. *Toward the tools selection in model based system engineering for embedded systems – A systematic literature review.* Journal of Systems and Software. Vol. 29, pp. 150-163.

[10] XML Metadata Interchange (XMI) Specification. 2015. [online] Available at: https://www.omg.org/spec/XMI/2.5.1/PDF [Accessed 17 Mar. 2022].

[11] www.sciencedirect.com. 2012. Model Transformation - an overview | ScienceDirect Topics. [online] Available at: https://www.sciencedirect.com/topics/computer-science/model-transformation [Accessed 3 May 2022].

[12] The Eclipse Foundation. 2017. *Eclipse Public License 2.0* [online] www.eclipse.org. Available at: https://www.eclipse.org/legal/epl-2.0/ [Accessed 24 Mar. 2022].

[13] Gnu.org. 2016. *The GNU General Public License v3.0 - GNU Project - Free Software Foundation*. [online] Available at: https://www.gnu.org/licenses/gpl-3.0.html [Accessed 23 Mar. 2022].

[14] www.prismmodelchecker.org. 2011*. PRISM - Probabilistic Symbolic Model Checker*. [online] Available at: https://www.prismmodelchecker.org/ [Accessed 24 Mar. 2022].

[15] Pradella, M., Morzenti, A. and San Pietro, P. 2008. *Refining Real-Time System Specifications through Bounded Model- and Satisfiability-Checking*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/4639315 [Accessed 24 Mar. 2022].

[16] N. Sánchez-Gómez, J. Torres-Valderrama, J. A. García-García, J. J. Gutiérrez and M. J. Escalona. 2020*. Model-Based Software Design and Testing in Blockchain Smart Contracts: A Systematic Literature Review*. IEEE Access. Vol. 8, pp. 164556-164569.

[17] Hugues, J., Yankel, J. 2021. SEI Blog. *From Model-Based Systems and Software Engineering to ModDevOps.* [online] Available at: https://insights.sei.cmu.edu/blog/from-model-based-systems-and-software-engineering-to-moddevops/ [Accessed 20 Apr. 2022].

[18] Aldrich, J. et al. 2019. *Model-Based Adaptation for Robotics Software*. IEEE software. Vol. 36 (2), pp. 83–90.

[19] Morin, B. et al. 2017. *Model-Based Software Engineering to Tame the IoT Jungle*. IEEE software. Vol. 34 (1), pp. 30–36.

[20] Utting, M. et al. 2012. A taxonomy of model-based testing approaches. Software testing, verification & reliability. Vol 22 (5), pp. 297–312.

[21] Chamberlain, R. et al. 2019. *Cyber Physical Systems. Model-Based Design: 8th International Workshop, Cyphy 2018, and 14th International Workshop, WESE 2018, Turin, Italy, October 4-5, 2018, Revised Selected Papers*. Vol. 11615. Springer International Publishing AG.

[22] Russo, A. & Schürr, A. 2019. *Model-based software quality assurance tools and techniques presented at FASE 2018.* International journal on software tools for technology transfer. Vol. 22 (1), pp. 1–2.

[23] Halligan, R. www.linkedin.com. 2019. *Benefits of Model-Based Systems Engineering*. [online] Available at: https://www.linkedin.com/pulse/benefits-model-based-system-engineering-robert-halligan/ [Accessed 22 Apr. 2022].

[24] Amalfitano, D. et al. 2015. *MobiGUITAR: Automated Model-Based Testing of Mobile Apps*. IEEE software. Vol. 32 (5), pp. 53–59.

[25] Henderson, K. & Salado, A. 2021. *Value and benefits of model‐based systems engineering (MBSE): Evidence from the literature*. Systems engineering.Vol. 24 (1), pp. 51–66.

[26] Barney, J. and Hesterly, W. 2018. *Strategic Management and Competitive Advantage: Concepts and Cases*, Global Edition. 6th Edition. Pearson International Content.

[27] Du, Y. www.linkedin.com. 2016*. What is Interdisciplinary Design?* [online] Available at: https://www.linkedin.com/pulse/what-interdisciplinary-design-yue-du/ [Accessed 26 Apr. 2022].