

Shining light on software modelling

A Creative Technology workshop

Femke Jansen

S1951262

Creative Technology | EEMCS | University of Twente

17-08-2020

Supervisors:

Dr. A. Fehnker

Dr. C.M. Epa ranasinghe

Abstract

Communication between software engineers and non-software engineers is often rough. But this communication is getting more and more important in today's world. To improve this communication there should be made more awareness towards solutions. A solution for this problem could be software modelling. Therefore, the research question is *How do we make students more aware of modelling and software design?* To answer this question a prototype was designed. This designing process was done by following the Creative Technology Design Process. The prototype consists of a workshop on modelling and software design. This workshop consists of lectures based around examples, card exercises, and feedback sessions. The card exercise is the main focus of the workshop. The card exercise is about selecting different themed cards which combined should help generate an idea. This idea then will be used to create a model as explained in the lectures. When evaluating this prototype with the help of interviews, surveys and analysing the requirements, it became clear that the prototype was a success. However, the prototype was not yet fully tested. This research was a good starting on finding a way to improve the communication between software engineers and non-software engineers, however, there is still a lot of future research to be done. For instance, further developing and testing this implementation.

Acknowledgement

I would like to thank my supervisors Ansgar Fehnker and Champika Epa Ranasinghe for their time, patience and help throughout the graduation semester and working on the thesis. Additionally, I would like to thank Sander Koomen for the emotional support, help and keeping up with my struggles. Finally, I would like to thank all the respondents to the survey, Marcus Gerhold and Joke van Staalduinen for their insightful feedback and willingness to think along.

Table of Contents

Abstract	2
Acknowledgement	3
Table of Contents	4
List of Figures	8
Chapter 1 -Introduction	9
1.1 Approach to software engineering	10
1.2 Software design	10
1.3 Model-based engineering	10
1.4 Model-based engineering in education	11
1.5 Learning resource and workshop	11
Chapter 2 – State of the art	12
2.1 Unified language	12
2.1.1 Unified Modelling Language	12
2.2 Software design	13
2.3 Model-based engineering	14
2.3.1 Pros of model-based engineering	14
2.3.2 Divisiveness in Model-based engineering	15
2.3.3 Cons of Model-based engineering	15
2.3.4 Model-based engineering in practice	16
2.4 Conclusion	17
Chapter 3 – Methods and Techniques	18
3.1 Creative Technology Design process	18
3.2 Ideation	19
3.3 Specification	20
3.4 Realisation	20
3.5 Evaluation	20
Chapter 4 – Ideation	22
4.1 Stakeholders	22
4.1.1 Power-Interest classification	22
4.1.2 Assumptions and risks	24
4.2 Brainstorm end product	25
4.3 Requirements	28
4.4 First design	28
Chapter 5 – Specification	31
5.1 Continued development	31

5.1.1 Models -----	31
5.1.2 Exercise cards-----	32
5.1.3 Time table -----	32
5.2 Workshop details-----	34
5.3 Interviews -----	35
5.3.1 Interview Marcus Gerhold -----	35
5.3.1.1 Cards-----	35
5.3.1.2 Layout workshop-----	35
5.3.1.3 Content-----	36
5.3.1.4 Overall tips and added feedback-----	36
5.3.2 Interview Joke van Staalduinen -----	37
5.3.2.1 Cards-----	37
5.3.2.2 Layout workshop-----	37
5.3.2.3 Content-----	37
5.3.2.4 Overall tips and added feedback-----	38
5.4 Finalisations -----	38
5.4.1 Models -----	38
5.4.2 Exercise cards-----	39
5.4.3 Lecture content -----	39
5.4.4 Time table -----	39
5.5 Final requirements-----	40
Chapter 6 – Realisation -----	41
6.1 Workshop design-----	41
6.1.1 Cards -----	41
6.1.2 Design lectures-----	43
6.2 Content workshop -----	43
6.2.1 Introduction-----	43
6.2.1.1 What is the workshop about-----	43
6.2.1.2 Information on modelling -----	45
6.2.1.3 Examples and real-life experiences -----	45
6.2.1.4 The Example that will be used to introduce the models-----	47
6.2.2 Block 1: Activity Diagram & State machine -----	47
6.2.2.1 Activity diagram-----	47
6.2.2.2 State machine -----	48
6.2.3 Block 2: Requirements list & Use-case diagram -----	49
6.2.3.1 Requirements list -----	49
6.2.3.2 Use-case Model -----	50

6.2.4 Block 3: Sequence Diagram	52
6.2.4.1 Sequence diagram	52
6.2.5 Conclusion	53
Chapter 7 – Evaluation	54
7.1 Survey	54
7.1.1 The workshop	54
7.1.2 The cards	54
7.1.3 The structure of the workshop	55
7.1.4 Blocks	55
7.1.5 Conclusion	56
7.1.6 Final remarks	56
7.2 Requirements	57
Chapter 8 – Conclusion	58
Chapter 9 – Future work	59
Appendix A – Interview guide	61
Questions to ask	61
Appendix B – Survey	62
Evaluation of Workshop on Modelling	62
Introduction	62
What is the workshop about?	62
Model Cards	63
Domain Cards	63
Target Group Cards	63
Technology Cards	64
The Workshop	64
The Cards	64
Structure of the workshop	65
Blocks	66
Activity diagram	66
State machine diagram	67
Requirements list	68
Use case diagram	68
Sequence diagram	69
Conclusion	70
Final remarks	71
Appendix C – Responses survey	72

The Workshop -----	72
The Cards-----	72
Structure of the workshop-----	75
Blocks -----	75
Conclusion -----	78
Final remarks-----	79
References -----	81

List of Figures

Figure 1: A Creative Technology Design Process – CTD [13].....	18
Figure 2: Baseline stakeholder table.....	22
Figure 3: Power-Interest diagram.....	23
Figure 4: Interest-influence classification.....	25
Figure 5: First Word-map.....	26
Figure 6: Overview list Word-map.....	27
Figure 7: Second brainstorm.....	28
Figure 8: Workshop schedule.....	29
Figure 9: Structure blocks.....	29
Figure 10: Cards.....	30
Figure 11: Example of card layout.....	30
Figure 12: Initial workshop schedule.....	33
Figure 13: Initial block session schedule.....	34
Figure 14: Requirements with the MoSCoW method.....	40
Figure 15: Overview model cards.....	41
Figure 16: Overview domain cards.....	42
Figure 17: Overview of target group cards.....	42
Figure 18: Overview of technology cards.....	43
Figure 19: Workshop scadual.....	44
Figure 20: Scadual of block session.....	44
Figure 21: Canvas system.....	46
Figure 22: Horus App.....	47
Figure 23: Activity diagram.....	48
Figure 24: State machine diagram.....	49
Figure 25: Requirements list.....	50
Figure 26: Use-case diagram.....	51
Figure 27: Sequence diagram.....	52
Figure 28: MoSCoW requirements achievement.....	57

Chapter 1 -Introduction

Graduates of engineering degrees have to contribute to software development projects.

Within these teams, they will need to be able to participate in discussions on Software Design even though they are not software engineers themselves. This project focuses on the design phase of a project where people from different disciplines work together. In this phase, IT is common to include both whiteboard discussions and written proposals where the designing is drawn out. These will be shown within software groups but also outside those groups to other engineers or even non-engineers to show the design. Not only is software design used beforehand in the creation process but it is often only used for later in the documentation phase.

From the current situation rise different challenges. One of the biggest challenges is the lack of integration of models and extensive software design in the work environment. While students at university usually learn about models, software design, and software architecture, companies do not use these methods in the same way, and it often depends on their development process. Most companies still use in practice more text-based documentation approaches and are more development focused. Students entering these work environments will not be able to use the things they learn on the subject, let alone change the companies way into using what they have learned.

Another occurring challenge is having a process that is understood by all members of the team. Because of the multidisciplinary teams, software projects have, not every documentation manner will be understandable for everyone. Often software engineers lead the way when documenting, while non-software engineers should also be able to understand the documentation.

Additionally, most of the engineers are not familiar with the design software programs. Student engineers often find the purpose of design models unclear or have difficulty understanding the abstraction and refinement. Lastly, software teams are not able to effectively visualize the software architecture of their system. Engineers can often visualize their process but not their software. There are too many ways but they are all not good enough to fully stick. Meaning there is no ubiquitous language for software, everyone has their variations on it.

The end product of the research will either be a workshop with online learning tools or a mini degree. To create this there has to be looked into ways of learning and teaching abstract topics. This then has to be combined with the findings on a workable way for designing software. This however might not easily be found, since every engineer thinks differently and every field is different. To find this, there first will be done a literature research. This literature research will be on the problems with unified language within software engineering and finding the most common approach, software architecture and

design, model-based engineering, software architecture and design in education, and learning resources and workshops.

1.1 Approach to software engineering

Within software engineering, there is no universal language. For as long as the existence of software engineering, engineers used different variations to document and design their software. S.Brown [1] talks about software teams that aren't able to effectively visualize the software architecture of their systems. He is searching for a way to visualize systems looking at existing ways in other fields of engineering. Brown often mentions how it should look like a map. The more details wanted the further has to be zoomed in. Sadly, there is still no unified language for software engineering. Therefore the project will look into the most used approaches of software engineering, one of which will be model-based engineering.

1.2 Software design

Software design is about the activities involved to create a computer program for a particular purpose. The interest in Software design has been rising within the software literature, mainly because of the great advantages it shows for software development. One of the main advantages of software design is the improvement in quality work and the decrease of errors while developing. However, the growing interest within literature seems to lack in the work field of software development. Developers often find the software design process too long and daunting.

1.3 Model-based engineering

Systems are getting increasingly complex. It is needed that these systems are understood throughout multiple disciplines, not only to improve existing systems, but also to design and create new ones. To design, communicate, and document these systems, model-based engineering is increasingly used. Model-based engineering, also called model-driven engineering or model-driven development, is a methodology to develop and design systems in a visual way by creating models. It is used to design, communicate, and document complex systems by offering a clear view of domains within the model of the system. Models are made with a specific goal in mind since no model can be made for different kinds of views and domains.

Even though model-based engineering is rising in popularity, it is not yet often used in practice. Researchers see great benefits in using model-based engineering. So much so, that students learn about Model-based engineering and model-based software design in universities. Nonetheless, it does not change the fact that most companies still have a text-based documentation approach to development. Over the years, a handful of companies did

switch to a model-based approach to development but this trend seems to evolve very slowly. Thus, there needs to be explored why adopting Model-based engineering into practice is challenging, regardless of the benefits shown by research. Therefore, the benefits and obstacles of model-based engineering need to be understood. This will be achieved by comparing papers on model-based engineering, development, and design.

1.4 Model-based engineering in education

Despite model-based engineering not being used in practice, model-based engineering is widely taught to many students. Since model-based engineering is one of the most used approaches of software engineering it is interesting to look into the approaches of teaching model-based engineering. To do so, this project will focus on the following aspects: the models used for model-based engineering in the classes, as well as tools and technologies used to do so, how those models will be used, the aspects predominantly covered, and the most effective ways to teach the students model-based engineering.

Model-based engineering is a different way of looking at code. Coding or writing software is often abstract and structural whereas model-based engineering is less abstract and structural and more about design and creating an overview to fulfil the requirements. This makes it for some students hard to grasp, since some are more structural oriented. Is there a way to teach model-based engineering so it is easily understandable for everyone.

1.5 Learning resource and workshop

The end goal of the project is to have a product to be able to teach non-software engineering students about software design and architecture to teach them to communicate and develop together with software engineers. First, the project looks into the ways of teaching in general. How do you easily convey a subject in general? When and how do students learn best? Afterwards, should there be more focus on how to teach software design and architecture? How are not only students learning this subject but also how do employees learn it?

The final product needs to be a way to convey the subject. There are multiple ways to do so. For starters, a popular way of learning new skills currently is via online classes and courses. This exists in different ways. Some online classes are video-based others are a combination of explanation videos and exercises accordingly. Examples of online classes and courses are Udemy, Skillshare and OER Commons. Additionally, a subject could be taught via a workshop or learning resources like workbooks.

Chapter 2 – State of the art

In this chapter, a literature study is conducted to get insight into the currently available information on unified language, software design and modelling.

2.1 Unified language

One of the main problems since the start of software development is a unified language or rather good communication within a development team. Brown [1] states that software teams aren't able to effectively visualize the software architecture of their systems which is the base of communication. Most people do have a certain way of visualizing but everyone is doing so differently making even the less effective visualizations incomprehensible. This lack of common vocabulary causes bad communication resulting in miscommunication between teams which could lead to errors in the project which do not have to be there. Brown [1] points out the importance of a common set of abstractions. According to Brown, a common set of abstractions is more important than a common notion. Brown focusses on diagrams. Diagrams are maps that help a team navigate a complex codebase, each level of the diagram is for other people. These levels show different details of the project according to what is needed for the person working on the project.

2.1.1 Unified Modelling Language

One well known language promoting unified communication is Unified Modelling Language (UML). UML is a standardized modelling language existing of 13 different diagrams categorised by structure diagrams, behaviour diagrams, and interaction diagrams [2]. The creators of UML saw the importance of models being used in software projects. According to the UML website [2] surveys show that for large software projects it is more likely that a large software application will fail to meet all of its requirements on time and budget than that it will succeed. To increase the chances of success UML uses modelling. Modelling is the designing of software applications, it is to visualize the design and check it against requirements before starting to code the application.

UML shows the importance of modelling by being able to work from different levels of abstraction. With UML someone can zoom in and out from different levels showing detailed views and a more general environment of execution. Having these different levels UML also shows the connection between them and thus between applications. Other advantages with the different levels used with UML is the choice to focus on different aspects of the application, such as business or engineering. A few other aspects of the UML platform is the middleware- and methodology-independents. UML can be platform-independent or platform-specific.

2.2 Software design

Software design and Software engineering are often closely used and depending on who is asked there are given different definitions for either. J.A. Robinson defines software design as everything to do with creating a computer program for a particular purpose in his book *Software design for engineers and scientists* [3]. P. Freeman and D. Hart define software design as all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems [4]. In 1997 Rumbaugh said design is describing how a system will be constructed without actually building it. Software design is not a logic science only about functionality. Software design is also about factors as innovation, style and requirements. Which makes software design a different way of thinking than the current way within computer system engineering.

Software design does not exist of rules to follow. It is about practice and learning to use the tools. There are a lot of tools out there and they all use different elements and diagrams, a few well-known diagrams are activity diagrams, use case models, class diagrams, and sequence diagrams mostly used with UML. It is not possible to use all of the different tools in every project so it is up to the designer how the end product will look. Having a good end product, in the end, depends on the natural skills of the designer and amount of practice.

Software design is often talked about in literature. It has been an up-and-coming way of looking at software and system development. According to Joke van Staalduinen [] using software design will be of great benefits for system development. Van Staalduinen mentions the importance of design comparing it to building a house. Before building a house, the house will be designed since it is not doable to change a lot of thing when already building. Computer systems are much more complicated which makes designing even more important. Another reason software design is important is the time efficiency. By creating a clear overview of the system, the development is easier and more quickly done. Additionally, designing the software beforehand lowers the chances of mistakes later when developing, since everything is well thought out beforehand. Considering different situations and mistakes they might give and adapting the design on them prevents making those mistakes when coming into that situation when developing. Lastly, software design takes all the different stakeholders into account when designing the system, ending most likely up with a system easily useful for everyone.

Despite the stated advantages of software design it is not mainly used and taught. One of the main reasons is mentioned by Freeman and Hart: "Creating a science of design requires computer scientists and engineers in academic institutions and industry to think about software differently and make fundamental changes in the way they work" [4]. Van Staalduinen confirms the lack of usages. Van Staalduinen has seen it in her environment as well on universities as in the industry. She mentioned promoting software design within the

universities but often getting slammed for it. Students do not want to do long design processes, they almost immediately want to start developing and the universities seem to think the same way. Learning the students to program and develop is more important than teaching them how to design what they want to develop. However, not only the students have problems with the long designing processes, but in the industry people also tend to leave out a solid design process and go almost straight to the development. Not only forgetting to think through the process but also forgetting to talk to the user and keep them in mind while developing.

2.3 Model-based engineering

2.3.1 Pros of model-based engineering

The advantage mentioned most often of using model-based engineering is the possibility to show stakeholders specific views. Modelling is nothing new in the engineering world however, it was often used domain-specific. System engineers use models as a development tool but often these models are not understandable or applicable to the software engineers of the same system. With the model-based approach “the state of the system can be automatically generated for the different stakeholders”[5]. The advantage of the multiple views is mentioned in the majority of articles. Piaszczyk for example states “model-based system engineering brings the system model to the centre stage for use by all stakeholder together” [6]. Partly because of the stakeholders specific views, the gap between design and development decreases. This gap decreases because with Model-based engineering the development starts after the design phase. Often, this design phase is cut short since developing is often found to be more important. However, when doing a model-based approach, the focus is on the design process. This design process is meant to make the development phase go smoother. According to Piaszczyk, with model-based engineering, the entire system and surrounding environment is visualized, creating the overview to use when developing.

Mandi and Sievers mention model-based engineering is a means to optimize speed, cost, and quality according to some people [5]. Model-based engineering optimizes the speed by making a detailed model at the start of the development to prevent mistakes and defects further on in the process. This means at the beginning of the process, model-based approaches will take longer but in the long run, they speed up the process. A faster process also means lower costs. The cost optimization is not only because of the reduced development time but also because of the reduction in errors made in the development process. Similarly, the quality of the development is optimized by the reduction in errors in the development process. Mandi and Sievers state “early studies indicate that up to 40%

fewer requirement defects were found in systems development with model-based engineering” [5].

2.3.2 Divisiveness in Model-based engineering

Two articles state that model-based engineering makes it easier to make changes or reuse the model later on [6] [7]. However, Ahmed and Ashraf state “model-based methods lack the flexibility of reusing knowledge in building and transforming models” [8]. All three articles do not state more specifics on why Model-based engineering is or is not flexible to reuse or change. Additionally, in the previous section, it was mentioned that model-based approaches optimize speed, but multiple articles talk about the time-consuming work of Model-based engineering [9] [6] [8]. With the pressure for companies of competition and cost efficiency they often choose for the fastest way. A document-centric approach to development is a faster way to go through design-wise. Next to the quick documentation of the process, the document-centric approach to development is best known by people making the process even quicker than using a model-based approach which people are still trying to understand.

2.3.3 Cons of Model-based engineering

Model-based engineering has no set of clear rules which shows multiple obstacles. Patou et al, mention a steep learning curve as one of the obstacles of model-based engineering [10]. Since there are no clear rules, people have a hard time learning model-based approaches since it is more a way of working then something with a clear step by step guide. Additionally in some situations, the meaning or the working of the model may not be clear since the modeller can shape it to their liking. Ramos et al. talk in their paper about the modeller shaping a model according to their favourite modelling approaches [11], making models often incompatible and inconsistent. Madni and Sievers also mention in their paper the “incompatible and inconsistent semantics” [5] making no set of clear rules one of the biggest obstacles in model-based engineering.

The main focus of Model-based engineering is the integration of multiple practices in one model. However, making models for every aspect of a system is hard and not always possible. Patou et al. argue that the integration of multiple practices in a model while managing quality and a set of rules is not possible for now. Additionally Schulz et al. mention “no one can build models that account for all behaviour” [7]. Schulz et al. talk about making subsets of the overall system, which together makes the whole system. Creating subsets might make some argue that model-based engineering does not create a clear overview of a system, but simply an overview of parts of the system where people still need to find out the connection between.

2.3.4 Model-based engineering in practice

Model-based engineering is rising in popularity but in practice, document-centric approaches are often still used. Model-based engineering has a lot of advantages as shown before, how is it then that in practice it is not used? A few articles point towards this problem of having a good alternative but it not being used in the work environment. However, the articles softly touch on the subject. Both Madni and Sievers [5], and Cameron and Adsit [12] talk about the not readily fit of model-based engineering within the traditional way of documentation. Madni and Sievers say “the move to Model-based engineering requires a cultural change during system development” [5]. According to them, there is progress but the gap between document-centric and model-based approach is still big today. According to Cameron and Adsit, the acceptance within a lot of fields is still missing and needed to change from document-centric to model-based development. Additionally, Madni and Sievers mention the tendency to rush the architectural phase even though the importance of strict architecture is well known.

Madni and Sievers state that training can be used to reduce the shift the use of document-centric approaches towards the use of model-based approaches. However, there will always be reliance on parts of the document-centric approaches. Ahmed and Ashraf approach a solution for the reusability problem that might cause some to stay with the document-centric approach. According to them, the reusability problem could be overcome by patterns. “Patterns are usually presented as a vehicle to capture the best practices and facilitate their dissemination”[8]. According to Ahmed and Ashraf patterns can be changed according to the current state and customized.

Lastly, a few articles talk about the resistance of transitioning from document-centric to model-based approach coming from the doubt of the improvement when using model-based engineering. Working with the document-centric approaches have been good so far and making that transition is a lot of effort. If this effort is made, there needs to be certain it will improve the development, reduces time and reduces mistakes. Since people are not sure if that is the case, they prefer staying with what they know works. To shift this mentality, there should be more attention towards successful examples of model-based engineering used in work environments to make people see the improvements model-based engineering makes.

Concluding, the biggest advantage model-based engineering has is the possibility to show different levels of the modelled system, giving the possibility for different stakeholders to understand the system. As a result, model-based engineering improves the quality, time and costs of development. The biggest disadvantage of model-based engineering is the lack of strictness and rules. model-based engineering is used to fit the specific development

making, this results in different models and workings per development. These different models and workings make model-based engineering seem chaotic. Two arguments were used as advantage and disadvantage. The ability for reusability and change to models was said to be the reason to use model-based engineering. However, in another article, it was not doable to reuse and change models. Additionally, the amount of time it takes to go through the process of model-based engineering was according to some articles an improvement thus an advantage. However, one article said the process takes too long. Nevertheless, the process indeed is longer in the beginning but it should reduce the time spend later on reinventing, changes and mistakes.

2.4 Conclusion

With the knowledge gathered from the state of the art, an implementation can now be made of that information. When designing this implementation the pitfalls and tips found in the state of the art will be used for improvement. For example, it is important to show the importance of modelling however, it can get complicated since the many different opinions and lack of clear guidelines on usage.

After collecting insight on the current information on unified language, software design, and model-based engineering the following research question was set up: How do we make students more aware of modelling and software design?

Chapter 3 – Methods and Techniques

In order to design a prototype, several techniques will be used. In this chapter, there will be explained what methods and techniques will be used for the rest of this thesis. The creative technology design process, stakeholder analysis, brainstorming, interviews, surveys and requirement analysis will be explained.

3.1 Creative Technology Design process

This report follows the Creative Technology Design Process (CTDP) proposed by Mader and Eggink [13]. This design process is specially made for the bachelor Creative Technology.

When creating this process the goal of Creative Technology was kept in mind: “To design products and applications that improve the quality of daily life in its manifold aspects, building on Information and Communication Technology” [13]. Key elements to creating this design were existing approaches. One described by Jones starting with a divergence phase followed by a convergence phase. The second approach is Spiral models based on Wieringa’s problem-solving. The CTDP consists of four main phases; Ideation, Specification, Realisation, and Evaluation. The process with these phases are shown in Figure 1.

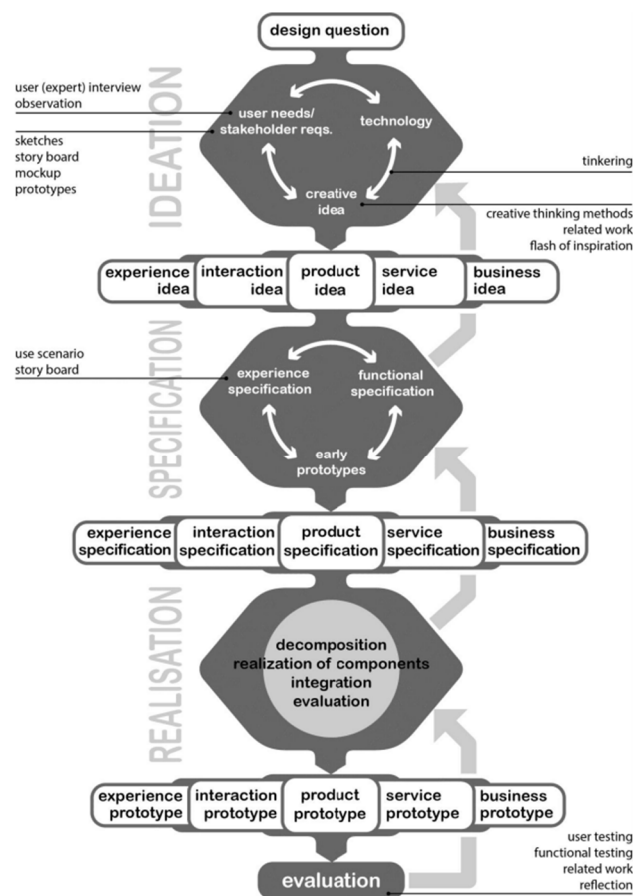


Figure 1: A Creative Technology Design Process – CTDP [13]

CTDP start with the ideation phase. The ideation phase is about idea generation and evaluating those ideas with users and stakeholders. The goal of the ideation is to have “a more elaborated project idea, together with the problem requirements” [13].

The ideation is followed by the specification phase. In this phase, the design space is explored by using multiple prototypes, followed by a short evaluation. This then is applied to a feedback loop. “Prototypes are subsequently discarded, improved or (partly) merged into new prototypes. The evaluation may also lead to a new functional specification, in its turn leading to a new prototype” [13].

Next, the realisation phase starts. The goal of the realisation phase is to realise the final prototype. This is done by combining all the knowledge of the previous phases. Within this phase, small evaluations can be done to see if the prototype fits the requirements.

Finally, the fourth phase of the CTDP is the evaluation phase. The evaluation phase is mostly focused on seeing if the original requirements are met. However, functional testing could also be done to test earlier functional requirements. This phase should be a reflection to contribute to personal and academic progress. This means “a personal attempt to make implicit decisions explicit and to reconsider one’s own (implicit) standards” [13].

3.2 Ideation

In the ideation two techniques are used. The ideation begins with a stakeholder analysis. According to J.M. Bryson: “Stakeholder analyses are now arguably more important than ever because of the increasingly interconnected nature of the world”[14]. By using a stakeholder analysis, a clearer view of the needs of the people involved in the project is achieved. Knowing these needs and keeping them in mind while creating improves the product. The stakeholder analysis of this thesis is done in a few different steps. First, the stakeholders are categorised into baseline stakeholders to get an understanding of their involvement in the project [15]. Additionally, a power-interest classification is made to see how to manage which stakeholders[16]. Lastly, assumptions and risks are gathered. When working with stakeholders conflicts could occur between them. The risks and assumptions are used to get a better understanding of what might happen and how to prevent that from happening.

The second technique used in the ideation is a brainstorm. The brainstorm is split up into two parts. The first part is a Word-map. In this Word-map everything to do with a workshop of learning tool is thought about. This is put onto a paper into keywords. This Word-map is then made into a clear overview. The second brainstorm was done with the goal in mind to further work out the workshop idea. This was done by sketching designs and writing keywords to support those sketches.

3.3 Specification

In the specification two techniques are used: the MoSCoW method for gathering requirements and interviews for gathering feedback. The MoSCoW method was created by Dai Clegg [17], [18]. It started as a tool to prioritize within projects. Currently, it is used more widely. In this thesis, MoSCoW is used to create a requirements overview. Using MoSCoW shows which requirements are a higher priority. MoSCoW stands for the categorisation of the prioritize: Must have, Should have, Could have and either won't have or would like to have, depending on the source and thus preference[17]–[19].

The second technique used in the specification phase is interviews. To receive feedback on the design made in the specification phase interviews are conducted with teachers in the modelling and software design field. The interviews are based on the content of the workshop, the design of the workshops and personal experience of the interviewees. The interviews will be an unstructured interview [20]. This means that the interview will be held with an interview guide. The participant will be asked a question to which the participant can respond how they see fit. The interview will follow the response from the participant. When the conversation starts to slow a new question from the guide will be asked. The interview guide used in the specification is can be found in *Appendix A – Interview guide*.

3.4 Realisation

In the realisation, there are no specific techniques used, since the realisation is about working of the work from the phases before. However, there are some tools used to create the design and content. To create the design of the cards Adobe Photoshop [21] is used. This was used since it has a lot of options to create strong designs. There are a lot of tools within Adobe Photoshop to give effects and change the design easily. Besides, with Adobe Photoshop it is possible to change pictures but also add your designs. Additionally, Lucidchart [22] was used to create the example models for the lectures. Lucidchart was chosen to use since it is a freely accessible site with an easily usable interface to create charts, diagrams and tables.

3.5 Evaluation

The evaluation will be done in the form of a survey. A survey was chosen for the evaluation since the evaluation was done when most of the university was already finished and on vacation. By using a survey people could better be contacted and answer the questions in their own time. This survey will be about quality, not quantity since specific feedback is needed on knowledge of modelling and teaching. For that reason, the target group of the survey are teachers of Computer Science, teachers of Creative Technology with a

background in Computer Science, TA's of Creative Technology focused on programming subjects and a Technical Computer Science student. The survey consists of open-ended and closed-ended questions. The close-ended questions are order response [23], meaning they have 5 options going from agree to disagree.

Chapter 4 – Ideation

In this chapter, the first requirements and design will be made. This will be done by first conducting a stakeholder analysis. Followed by a brainstorm on the final product idea.

4.1 Stakeholders

The identified stakeholders for this project are students, teachers, the program staff of the modules, developers and management in the work field. To get a better understanding of the stakeholders a stakeholder analysis as described in Chapter 3 – Methods and Techniques will be done. In this section, there will be looked closer to these stakeholders. The above-mentioned stakeholders are generalised. In Figure 2 all stakeholders are shown in an overview of baseline stakeholders. These roles are assigned to the stakeholders based on their involvement in the project.

Stakeholder	Baseline stakeholder
Creative technology teachers	Users
Computer science teachers	Users
Other teachers	Users
University	Decision makers
Developers in workfield	Users
Management in workfield	Users/Decision maker
Creative technology students	Users
Computer science students	Users
Other students	Users
Ansgar Fehnker	Developers/Decision maker
Programme staff	Decision makers
Femke Jansen	Developers

Figure 2: Baseline stakeholder table

4.1.1 Power-Interest classification

When doing a stakeholder analyses it is of interest to start by creating an overview of the power and interest of the found stakeholders to get a better understanding of the influence the stakeholders have on the project. Figure 3 shows the influence of the stakeholders of this project in a Power-Interest diagram. The diagram is separated into four different fields. The upper left field has high power and low interest, the stakeholders in this field should be kept satisfied. The upper right field has high power and high interest, the stakeholders in this field should be managed closely. The lower left field has low power and low interest, the stakeholders in this field should be monitored and require minimum effort. The last field in the lower right has low power and high interest, the stakeholders in this field should be kept informed. In Figure 3 the six general stakeholders are shown.

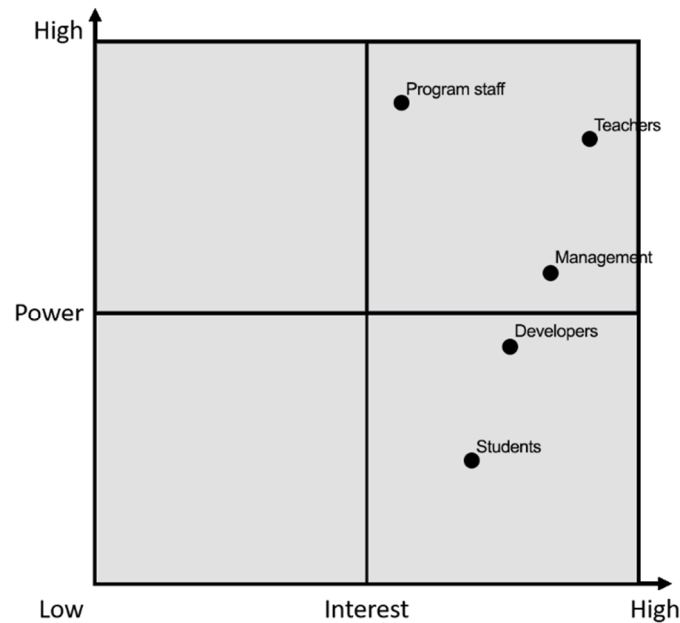


Figure 3: Power-Interest diagram

Program staff

The program staff creates the modules and thus decides partly what is thought in those modules. They have a lot of power over the content that is taught. However, they do not go into detail on what should be taught and they leave room for the teacher to teach as they wish. Showing that the program staff has high power but less interest, meaning that they are in the upper right corner. So program staff should be managed closely.

Teachers

Teachers are high in power and interest. Teachers mostly create the courses and thus choose the subjects told within a course. They can change the course as they see fit and have the direct possibility to do so. They may have to pass it by the program staff but for the most part, they are in charge of the formation of the course. Therefore the teachers fall into the upper right field meaning they have to be managed closely.

Students

The main focus for the end-users will be on students. So automatically students are one of the stakeholders. However, students do not have a lot of power over the workshop or learning tool. They might be able to share feedback but students are not the ones deciding if the topic or the workshop is given. Students have a higher interest value, although this may differ per student. Some students will be interested in the topic while other students less. However, the average of students will find it somewhat interesting since they either choose

the subject or it is part of their studies. The students fall under the lower right field meaning they have to be kept informed.

Management in the work field

Management in the work field is comparable with the program staff, they decide what is important and how the company is showcased. They will look closely to what is important and if what they do fits the companies vision. So the content of the workshop and the way the workshop is taught will be considered by management before they decide to get the workshop. Thus, management has a relatively high power, however not higher than teachers and program staff since they are way closer to the material and thus have more influence. The interest, however, is for management high placing management in just in the upper right field. Management thus has to be managed closely.

Developers

Developers have less power than management but have a great influence on what is done within a company. If they see something fit for improvement in skills or efficiency the management will most likely go along with their ideas. The interest of the developers is around the same as management however since this workshop promotes with more efficiency and lower risks management will most likely be more interested. Especially since developers are known for working in their way. Alongside that, software design and developers work very differently which often makes them not likely to match. The developers, therefore, are placed in the lower right field, meaning they should be kept informed.

The developers and management in the work field, however, will not be included in the design process for the rest of the research. This is because this research will mainly focus on creating a workshop for students. These stakeholders will be kept in mind for the stakeholder analysis since they could be of great importance in the future.

4.1.2 Assumptions and risks

Interests of stakeholders might not be in line. To shine a light on those risks an assumptions and risks table is made as shown in Figure 4. Of every stakeholder, the project influence and importance is noted, followed by assumptions and risks. When deciding on the requirements these assumptions and risks should be kept in mind.

Stake holder	estimated project influence	estimated project importance	assumptions and risks
Creative technology teachers	High	High	Have the curriculum already fully packed, find other topics more important, can not decide on which topics discussed in the curriculum
Computer science teachers	High	High	Think programming skills are more important than design skills
Other teachers	High	High	Do not care for computer science based content, do not understand the importance of modelling
University	Low	High	detailed content of lectures, wants the students to be excellent in what they learned when leaving the university
Developers in workfield	Medium	Low	Want to do it their own way, do what they know and do not want to change that
Management in workfield	Medium	Medium	Do not understand programming, so could make a bad judgement
Creative technology students	Medium	Low	To all rounded to understand, but likes to explore if it is interesting
Computer science students	Medium	Low	Often dislike design, rather do programming subjects
Other students	Medium	Low	Do not have enough knowledge on programming

Figure 4: Interest-influence classification

As shown in Figure 4 there are a lot of assumptions and risks to keep in mind when designing. One of the concerns is to keep all the students and teacher satisfied. One way of doing so is by listening to their interest and assumptions. For example, to make the final product interesting for all kinds of students an engaging exercise could be used to make it more fun for students who have more pre-knowledge and make it easier understandable for students with no or low pre-knowledge.

4.2 Brainstorm end product

After analysing the stakeholders and finding the requirements a brainstorm on the end product is done. There are a few things that need to be looked at in this brainstorm. First, the way of conveying information needs to be thought out. Secondly, the content of the workshop has to be worked out. Lastly, the structure of the end product has to be established. The brainstorm was done in different stages in different ways.

The first brainstorm was done with a Word-map followed by an overview list as shown in Figure 5 and Figure 6. In the Word-map, there were different focusses. First of the way of conveying the information. As shown in Figure 5 the options thought up were video tutorials, online workshop, live tutorials, a booklet, an activity sheet, or activity cards. Also, the goal of the project was Word-mapped. The results of that Word-map were to give students a tool and techniques, make sure the students understand the usages of models, and shine a light on models and software design. Then a Word-map was made to see who the target group would be. As a result of that brainstorm, students would be the stakeholders. However, there could be different options on what kind of students. Do they have specific prior knowledge or do they have different prior knowledge? Are they participating voluntarily or is it a mandatory part of the curriculum?

Lastly, the structure of the product is brainstormed. The options for duration are a one-day event or multiple sessions. Then there is also thought about if there should be a diversity of activities or diversity of techniques. In the overview in Figure 6 these ideas are put into more structured lists. They focus on the definition of a session, the audience and the aim of the product.

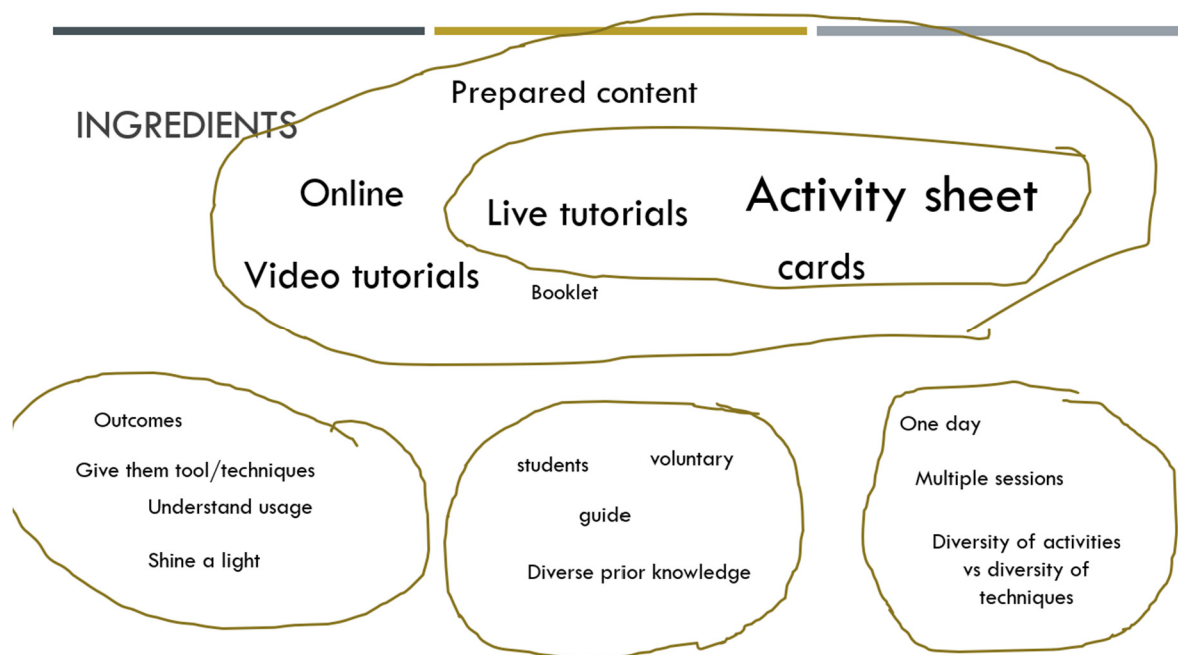


Figure 5: First Word-map

CHOICES

- How to define a sessions
 - By technique (class diagram vs activity diagram vs block diagram)
 - By source (English description, requirements document, source code)
 - By abstraction level (system architecture, software architecture, class level, within a class)
- Within a session what is the audience
 - Business
 - Lecture/talk
 - Other developers in the team
 - Other developer outside of a team
- What is the aim?
 - Inform
 - Convince
 - Document
 - Design
 - Ideation

Figure 6: Overview list Word-map

Later there was a second brainstorm done to specify the ideas more. Beforehand the decision was made to create an offline workshop. Since this would be more engaging and thus students would be more likely to learn better. This brainstorm was more freely done by making small sketches and writing catchwords as shown in Figure 7. In this brainstorm, a rough idea was sketched of the layout of the workshop. The idea sketched was to split the workshop up in block starting with an introduction and finishing with a conclusion. In between the introduction and conclusion would be 4 blocks focused on a model each. These blocks would consist of a lecture, a group example and an exercise on their own. The group example an own exercise were based on the paper *Let's Get Physical: Promoting Data Physicalization in Workshop Formats* from S. Huron, et al. [24]. The exercises would consist of combining cards to create a model. Other ideas were to combine a few models per block instead of one per block and letting the students teach each other about the models. Since explaining a topic makes the students understand that topic better. They then would have the time to do research, prepare and present. The first found card topics were domain and technology later models were added and for a longer time, there were multiple options for additional cards: stakeholders, intention or usages of the model. Lastly, the idea was found to let the students create a model and give feedback to each other.

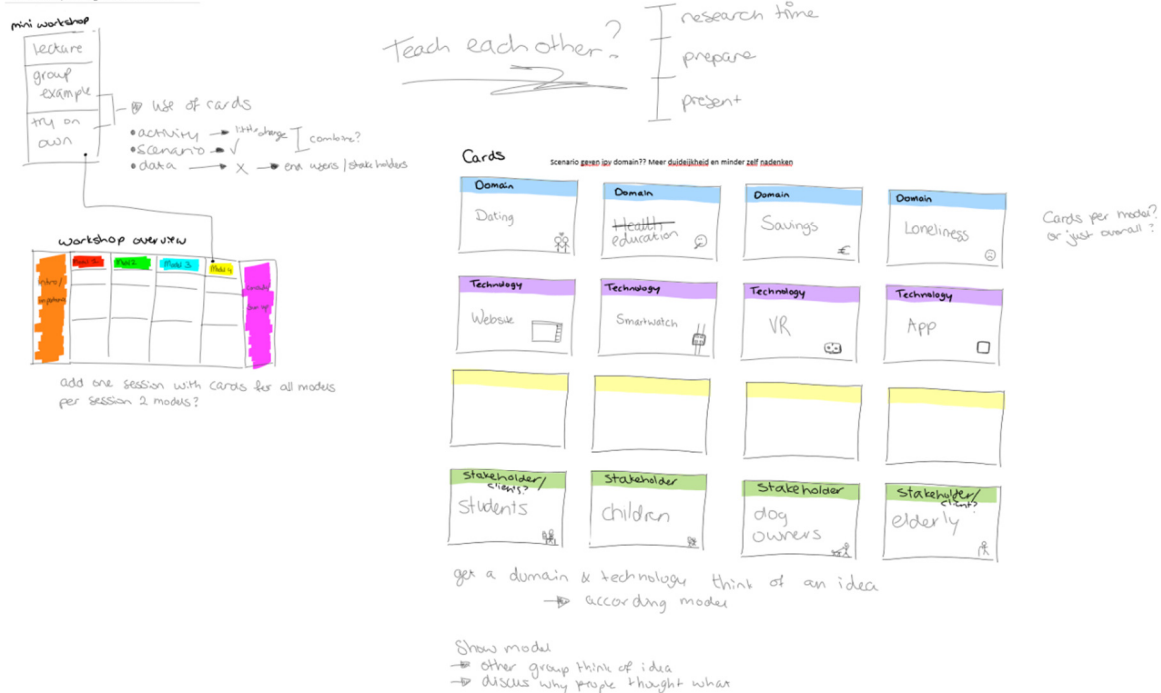


Figure 7: Second brainstorm

4.3 Requirements

The requirements set up for this project are:

- The result must convey knowledge on models.
- The result must motivate to use modelling and software design.
- The result must show the importance of modelling and software design.
- The result must consist of understandable content.
- The result must show improvement in the skills of the participants.
- The result should make clear how to use modelling and software design for programming.
- The result should be enjoyable.

4.4 First design

The idea of the first design is a workshop consisting of 6 blocks as shown in Figure 8. Of those 6 blocks, one is for the introduction and one for the conclusion of the workshop. The other 4 blocks are meant to introduce and exercise models. These blocks are structured as follows: starting with a lecture on the models followed by an in front of the class example card exercise. Lastly, the students themselves will in groups do their card exercise.

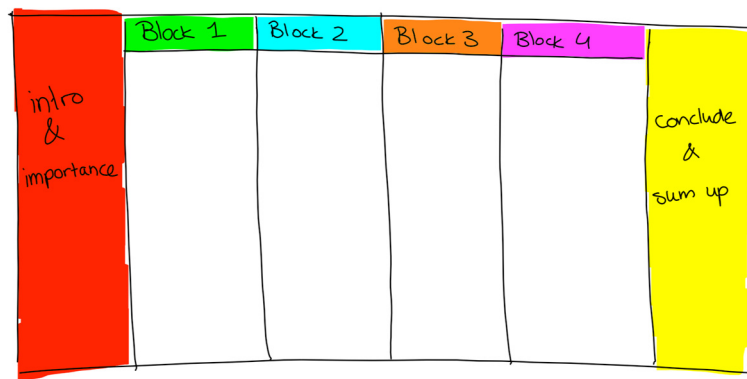


Figure 8: Workshop schedule

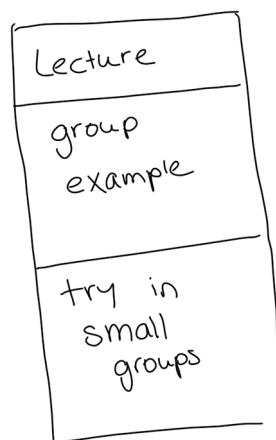


Figure 9: Structure blocks

For the card exercises, there are 4 categories of cards designed. To start the exercise one card has to be drawn from each category. With those categories, the students have to come up with an idea that fit those categories and create a model accordingly. As seen in Figure 10, two of the categories are domain and technology. For each of these categories, a few examples are given. For instance, in the category domain, there is a card “savings” meaning the idea the students come up with should have to do something with savings. If they combine it with one of the technology cards for example website, the students have to think of a website that has to do with savings. When the students thought out the idea they have to create the according model that has been taught in that block. A rough design of the card is shown in Figure 11, every category will have its distinctive colour. Additionally, every card will have a picture alongside the description to emphasise on the meaning.

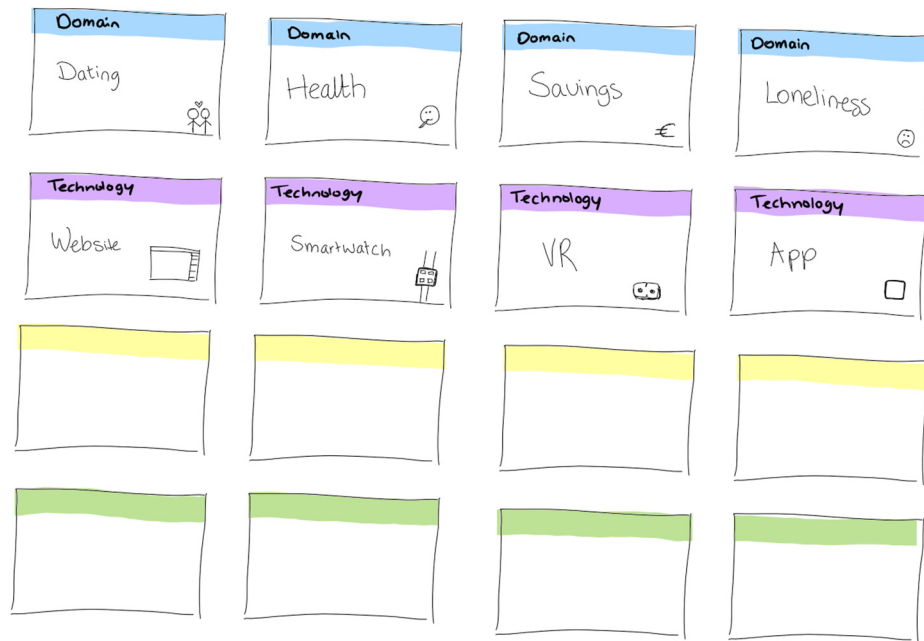


Figure 10: Cards

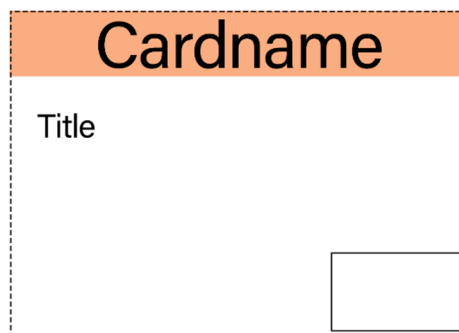


Figure 11: Example of card layout

Chapter 5 – Specification

In order to further specify the requirements found in the ideation phase, the first design is further elaborated in this chapter. Adding to the card exercise, the models and the time table. In this chapter, the workshop details will also be added to the design. This design will then be evaluated via interviews. Resulting in the final changes to the design and the final requirements.

5.1 Continued development

To further develop the workshop the design and content of the workshop should be further elaborated.

5.1.1 Models

To develop the workshop there should be established which models will be taught in the workshop. The first chosen models were: *activity diagram*, *use-case diagram*, *class diagram*, *data flow diagram*, *requirements list*, and *sequence diagram*. These models were chosen since they are commonly used and diverse diagrams. When selecting the diagrams there was chosen to not go for only UML diagrams. Since they do not have to be the way to go. The point of the workshop is that models can be used in different ways and that there are no strict rules. Using only UML diagrams would go against this idea.

Later the *data flow diagram* was removed from the workshop as well. This was done since after some research it was clear that this diagram was used less and less. Alongside that, the diagram was more difficult than the other diagrams since it had way more components. Instead of this diagram, no other diagram was added. Since finding a fitting diagram to add to the workshop was hard to find. Often diagrams are too high level or are too vague.

The combinations and order of the models within the workshop blocks were also decided. However, this changed a lot throughout the process. For one the models changed a few times. Secondly, there should be a flow in the workshop by combining comparable models but also have the participants start with the lower level simpler once and ending with the higher level harder once.

In the end, the models are gone through within the workshop as follows: *activity diagram* and *class diagram*, *requirements list* and *use-case diagram* and ending with *sequence diagram*.

Finally, for the models part of the development, an example had to be thought of to use in the lectures to explain the models. Alongside this example, the example diagrams of that example should be designed. When deciding on the example to use for the lectures a small

brainstorm was done. Before starting the brainstorm some requirements for the example were made:

- The example should fit all the diagrams.
- The example should be understandable.
- The example should be clear for everyone, step by step.
- The example should be a task a lot of people are familiar with.

When combining these requirements a few ideas were thought of:

- Booking a flight seat.
- Buying something online.
- Booking a meeting room.
- Sending an Email
- Online voting
- Finding an article
- Making a money transfer online
- Posting a photo

5.1.2 Exercise cards

The first development in the process of the exercise cards is adding two extra themes to the cards. In the ideation, the themes *domain* and *technology* were chosen. However, to give enough possibilities to create an idea one to two themes should be added to the exercise. The two themes that should be added have to compliment the already existing themes. With these cards, an idea has to be generated so the participants need enough information to create an idea in a set time. However, they still need to have some space for creativity. After a small brainstorm, the theme *stakeholders* was thought of. Besides the *stakeholder* theme, it was decided that the models would also be an exercise card since in every block two models would be taught the teams had to be able to pick one. By adding the models to the cards, the teams can pick one of the models just as picking a card from the other themed cards.

Secondly, the design of the cards was worked on. Starting with the colours of the theme cards. To make a clear distinction between the themes colours were added. For the participants to quickly see what theme card they picked, the theme colours selected should be very distinctive from each other. To make the cards attractive for the participants to use they should have a cheerful design as the picture to emphasize the meaning of the card. Possibly, making the design the same colour pallet as the colour of the theme to create coherency.

5.1.3 Time table

When further developing the time table of the workshop there were two big changes made to the original idea. First off, the time slots and breaks are added, changing the lecture blocks

from 4 to 3 as shown in Figure 12. The workshop will start at 9 o'clock in the morning and ends at 5 o'clock in the afternoon. After every session there is a 15-minute break except for once around lunchtime then there will be an hour break. This break will be in between block 1 and block 2. The introduction and the conclusion will both be 30 minutes long whereas the block sessions will be 105 minutes long.

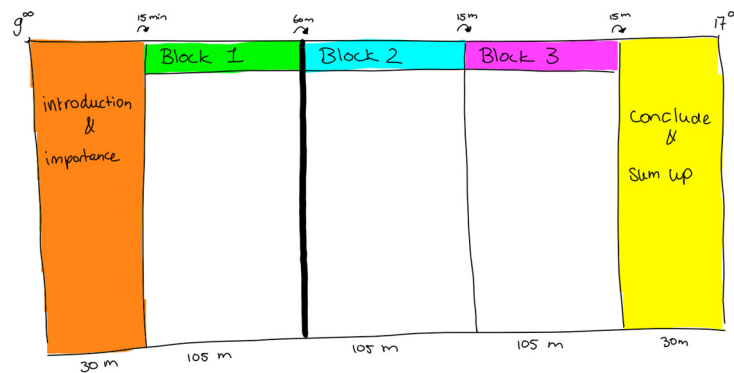


Figure 12: Initial workshop schedule

The second change when further developing the time table is the structure of the block sessions. In the initial design, the block sessions were split up into a short lecture followed by a group example of the card exercise and ended with participant trying the card exercise for themselves in small groups. However, after evaluating this structure it was founded that the participant learns more from explaining what they did and giving feedback to other participants. Because of that, the group example was removed from the structure and a feedback session was added to the structure. This feedback session will be about reviewing and discussing the models made with the card exercise in groups.

The time table for the block session is as follows. The block session will start with the information session on two models, this will be 35 minutes. Followed with a 30-minute card exercise session and a 45-minute feedback session as shown in Figure 13. Making it a total session of 105 minutes.

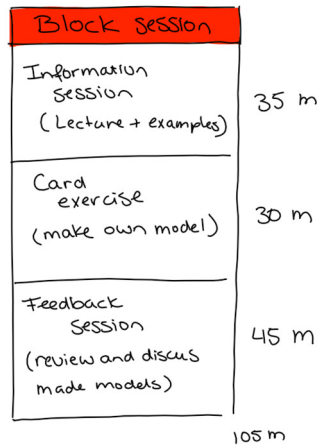


Figure 13: Initial block session schedule

5.2 Workshop details

Alongside the content and design of the workshop, other workshop details were developed as well. Starting with details on the participants. The workshop would be able to handle 20 to 30 participants. This, however, could be more but would be a great start. For this design, there is focused on participants voluntarily following the workshop.

Secondly, the exercise groups and the number of cards needed to be decided. The number of cards and the size of the exercise groups are based on the number of participants. When wanting 20 to 30 participant 10 cards per theme should be made to create small groups of 2 to 3 people for the exercise.

Additionally, a decision had to be made whether or not the participants have to have certain pre-knowledge on the subject. The most obvious pre-knowledge would be some experience with programming. However, this thesis is focused on making people with different backgrounds communicate better. So it was decided that no pre-required knowledge was needed. However, this meant that the content of the lectures and the according models should be understandable for participants in different kind of backgrounds. Since the final workshop is focused on Creative Technology students there can be a little bit more in-depth information because of the basic knowledge on programming those students have around year two. Nonetheless, Creative Technology students differ in knowledge because of the broadly taught subjects, meaning there still needs to be a focus on having understandable content for different kinds of backgrounds. The lecturer(s) however, does need pre-required knowledge for giving the workshop. They at least need to know the basics of modelling. It would also be helpful if they knew programming however this is not mandatory.

Additionally, the idea of adding (student) assistants to the workshop to support the lecturer came up. The assistants would mainly help with the card exercise and the feedback session to support the participants with any questions or struggles. Nonetheless, this is not necessary, meaning for now the workshop is only focused on having a lecturer and no assistants.

5.3 Interviews

5.3.1 Interview Marcus Gerhold

Shown and explained the interview and concept idea.

5.3.1.1 Cards

Overall positive on the content and design of the activity cards. According to Marcus the cards make sense and feel intuitive. He, however, had some notes on the stakeholder cards (changed to focus group/audience). He mentioned that not all models and diagrams use stakeholders or audience and even if they do, it differs how they call it per model/diagram. Marcus agreed with the combination of the themes of the cards, he could not think of changing something at the time of the interview.

5.3.1.2 Layout workshop

For the introduction of the workshop, Marcus mentioned that it is indeed important to have one. In this introduction, there could be told a few things that might be of importance. For starters, an introduction of what the workshop is about should be given. Besides that, Marcus makes clear that it is important to let the participants know that there is no right way of modelling, it is not an exact science. The modelling should visualize the idea thought of to make a clear overview. In the introduction, the lecturer should mention that with this workshop the mindset of modelling will be given not the perfect way of modelling.

For the concluding lecture, it depends on the audience what it should look like according to Marcus. It is important to feel the room after a packed day of information. Based on that, the concluding lecture should be able to be shortened. However, Marcus does mention that beforehand it should be planned for the full extent of time.

The first reaction of Marcus on the schedule was “quite intense”. He proposed to change the work and feedback session in duration. So 45 minutes for the card exercise and 30 minutes for a feedback round. However, this very much differs as both options have their pros and cons. The mini-lecture at the start of each block was a good estimate according to Marcus.

5.3.1.3 Content

Interesting models/diagrams to add to the cards according to Marcus are the *sequence diagram*, the *state machine diagram* and the *requirement list*. The advantage of the *requirements list* is that it is a list everyone can write and not a model to be designed. This might make it approachable for people who dislike drawing out ideas. As regards to the combination of models per block Marcus agreed with the combination of *dataflow diagram* and *class diagram*. For the other combination, Marcus suggested *activity diagram* and *state machine diagram*, because they model the behaviour of different levels. Other combinations are harder. Marcus questioned however how closely related the models should be per block. Maybe the learning experience is better when they are very different from each other.

A few important things for in the lectures were mentioned by Marcus. Starting with keep repeating that modelling is not an exact science and that working with them differs per model. Secondly, it is important to explain the level of extraction of the model being introduced and what the importance of the models is, make clear what is modelled.

5.3.1.4 Overall tips and added feedback

When asked about examples of real-life experience or in the workplace, Marcus had a few examples. He started with a tool for in classrooms to answer and help students called Horusapp. According to Marcus, there is a clear use of models in the design process when looked at the Horusapp. It looks clear and is user friendly. It would at least have used a use case. A less good example is Canvas. According to Marcus (and a lot of responses), Canvas is very unintuitive. If you would have used models this would not be the case. Marcus also mentioned that for the majority of software programs models aren't used and that is visible when using the programs.

A problem occurring when teaching models and software design to computer science students is that the students are there to learn how to program. This means that often they do not show interest in software design. However, as Marcus states once more it is also worth knowing what to program. A lot of the students do not want to think before they start to program, they just want to start programming. However, the more you program the more you will start to see the importance of modelling.

Lastly, Marcus mentioned that it might be interesting to look into making the content lively. He noticed with colleagues presenting models and software design it could get a little old and boring. An example he made was using role-play, student assistances act as the clients which the students have to interview to find out the requirements needed for the system. Because as often said; your client never understands what they want, you have to tell them.

5.3.2 Interview Joke van Staalduin

Shown and explained the interview and concept idea.

5.3.2.1 Cards

Joke was enthusiastic about the cards. According to her, they looked nice. She, however, did not have a lot of other feedback. Something she mentioned later on in the interview was that there might be some weird combination in cards. For example, dating and the elderly. However, she did agree that this could make some creative concepts.

5.3.2.2 Layout workshop

According to Joke the layout of the workshop is doable depending on how deep the workshop goes into the models. When staying at a higher level the 35 minutes should be enough. The 35 minutes is fast to explain the models so it should mostly be the highlights, to do so a good example that fits should be chosen. The *class diagram* needs some more information so might take a little longer than others.

Joke also had some feedback on the blocks within the workshop. According to her, it might be better to about the same time scheduled for the exercise and feedback rounds. Maybe even modify while teaching to see fit. If students are still working hard on their exercise, give them some more time, are they as good as done? Then go to the feedback round.

5.3.2.3 Content

Joke proposes to create one example that can be used for all the models and to use that to introduce the models instead of using a different example for every model. This will save time explaining the examples instead of explaining the models. This example can be presented in the introduction so that when the lectures for the models start the lecturer can immediately start explaining the model with that example.

Joke was wondering if the students would be able to design the models without knowing the other models. This since a lot of them are built of one another. She mentioned that use case models, data flow models and *activity diagrams* are beginner models but that the *class diagram* and *sequence diagram* are more advanced, especially the *class diagram*. Later Joke advised to not use the *class diagram* since it is too technical and it is used after some of the other diagrams have been used. When telling that the *class diagram* might not be the best option to use, Joke wondered if 6 models were needed or that 5 would be enough.

As for examples to use for the explanation of the models, Joke later sent a few suggestions: online shopping, booking a theatre seat, finding a recipe online, online meeting, tracking a delivery, booking a flight, and conducting an interactive online lesson.

5.3.2.4 Overall tips and added feedback

Joke did not have any specific real-life examples of her own this quickly but did tell about a student she taught modelling to. The student had sent her an email later in that year telling Joke about the benefits she had found using software design and modelling. When first taught the student was not really into designing and modelling, but the more she used it with other projects she realised how important it was. It was clear how much easier it was first designing than programming.

Additionally, Joke talked about how she once created a *sequence diagram* on a deeper level and showing how to program for that. This was to show the students how the models all tie together into the programming. Afterwards, some students got value out of it, others hated it. With this example, Joke told about how important it is to give understandable easy examples. However, the pitfall with easy examples is that those are easily programmed without any model. While modelling gives value to complicated systems. So, it is important to find a way of conveying the model clearly without showing that it is not needed to model.

Lastly, Joke had some commands to think about when giving the lectures and further designing the workshop. How do you let the students know software design and modelling is worthwhile in a short time? Models are used to get information out of people to give a clear overview of what is wanted from the system.

5.4 Finalisations

After evaluation and the interviews, some more changes were made to the workshop.

5.4.1 Models

Starting with changes with the models. As mentioned in the interview with Joke van Staalduinen *class diagrams* could get very complicated and are higher level diagrams. The remark of Joke van Staalduinen in combination with evaluating on the model the choice was made to remove the *class diagram* from the workshop. The first reason being that *class diagram* is too deep of a level of modelling to easily use within the card exercise. When using *class diagrams* a clear idea of the system is needed, since the model is about structuring the classes used in the program. Since the card exercises are meant to create a general idea this would not work. Second, *class diagram* might be too hard to understand when starting of with models. Often *class diagrams* are made after making more lower-level models. So without knowing or creating those, creating *class diagrams* might be too hard. Besides that, to create *class diagrams* basic programming knowledge is needed to know how to structure classes.

Instead of the *class diagram*, the *state machine diagram* was added to the workshop. The choice for the *state machine diagram* came from the interview with Marcus Gerhold. He mentioned adding the *state machine diagram* and combining it with the *activity diagram* since they both model behaviour. Adding the *state machine diagram* would indeed be a perfect fit for the workshop since the difficulty level is around the same as the already existing diagrams in the workshop and are the same higher level diagrams.

5.4.2 Exercise cards

When evaluating, it came to light that the card theme *stakeholders* made some confusion and was not the correct naming for the goal of this themes card. After rethinking what these cards represented the name of the theme was changed to *Target group*. This was because the cards were not representing all the stakeholders but a specific group for which the design would be meant.

5.4.3 Lecture content

The final chosen example for the lectures came from the brainstorm and the interviews done. The example that will be used in the lectures to explain the models is: shopping for a laptop online. When decided on the example, the according models had to be made. The example models were carefully designed with the shopping example in mind. Every model was made step by step according to the guidelines of the model.

5.4.4 Time table

To changes are made to the time table. Starting with the change in time of the block session structure. In the initial design in the specification phase, the structure existed of a 35-minute information session followed by a 30-minute card exercise session and a 45-minute feedback session as shown in Figure 13. The first thing to change was the information session of 35 minutes. In the initial design, it was stated that the block session would be 105 minutes. However, summing up the information session, the exercise session and the feedback session make 110 minutes. To get this back to 105 minutes the information session was changed from 35 minutes to 30 minutes.

After the interviews, it was decided that it was better to have the same amount of time for the card exercise session and the feedback session. This meant that the card exercise would get more time, 35 to 40 minutes, and the feedback session would lose time, also 35 to 40 minutes. To have a clear overview the card exercise session will be scheduled for 40 minutes and the feedback session for 35 minutes. However, it is possible to make some small changes if participants need more time for the exercise session.

Secondly, the end time of the workshop could be set a little earlier. After removing the *data flow diagram* from the workshop one model will be taught on its own in a block session. This means that less time is needed for the information session, meaning the final block session could be finished earlier. However, since this would only make it 15 minutes shorter for now the time table is not changed. Nonetheless, it is something to keep in mind when starting on the conclusion of the workshop.

5.5 Final requirements

Finally, the final requirements are decided. In Figure 14 the chosen requirements are shown. The requirements are prioritised with the MoSCoW method as explained in 3.3 Specification.





	Priority	Requirement
	Must	have diagrams with an accurate presentation of that particular diagram.
		have diagrams with an accurate presentation of the example.
		have understandable but detailed example models.
		be a workshop that can be held in a day.
	Should	have enough breaks.
		have enough time for every part of the workshop.
		have distinctive theme colours.
		have a cheerful designs on the activity cards.
		have an engaging and challenging card exercise.
	Could	have opportunity to give and receive feedback for the participants.
		have the same colour pallet for the theme colour and the design on the exercise cards.
		have teacher assistants for the workshop.
	Won't	
		take into consideration the workingfield as stakeholders.
		have more than 6 models explained.
		have non-UML models.

Figure 14: Requirements with the MoSCoW method

Chapter 6 – Realisation

In this chapter, the final design is shown with additional information. The chapter will walk through the workshop explaining every step and the according design. This will be split up into the workshop design and the content of the workshop.

6.1 Workshop design

6.1.1 Cards

The workshop design is mainly focused on the cards of the card exercises. In the workshop, there will be 4 types of cards: model cards, domain cards, target group cards and technology cards. There are 5 model cards and 10 domain, target group and technology cards. These combinations of cards were chosen because they give the basics to create an idea. With these cards, there is enough information to think of an idea but also enough room for creativity.

To create a fun and engaging exercise the cards should be engaging as well. To do so every type of card has a different colour and fun supporting designs are added to the cards. To make the cards all coherent they have the same style and outline. This makes the cards clear and easy to use. To continue the type colours the design on the cards have corresponding colours to the type of card it is for.

In Figure 15 an overview of the model cards is shown. The model cards consist of the activity diagram, the requirements list, the sequence diagram, the state diagram and the use-case model. There will be 10 groups which pick one of the two models explained per block session. So, there will be 5 cards of every model card.

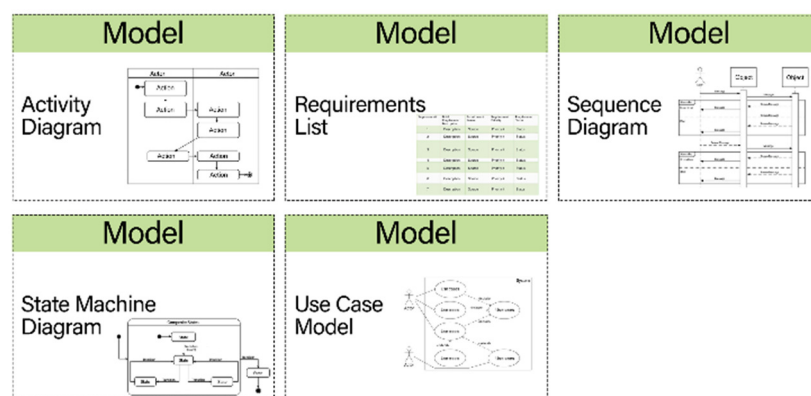


Figure 15: Overview model cards

In Figure 16 an overview of the domain cards is shown. The domain cards consist of dating, education, physical exercise, loneliness, savings, transportation, travel, nutrition, motivation and relaxation. To choose the domains a small brainstorm was done in the form of a brain dump. The idea was to think of everyday things but also specific things. Looking around and thinking about daily routines to create domains. Another way of thinking of

domains was to think of domains where the help of technology is already used. The goal was to find diverse domains which could create creative ideas.

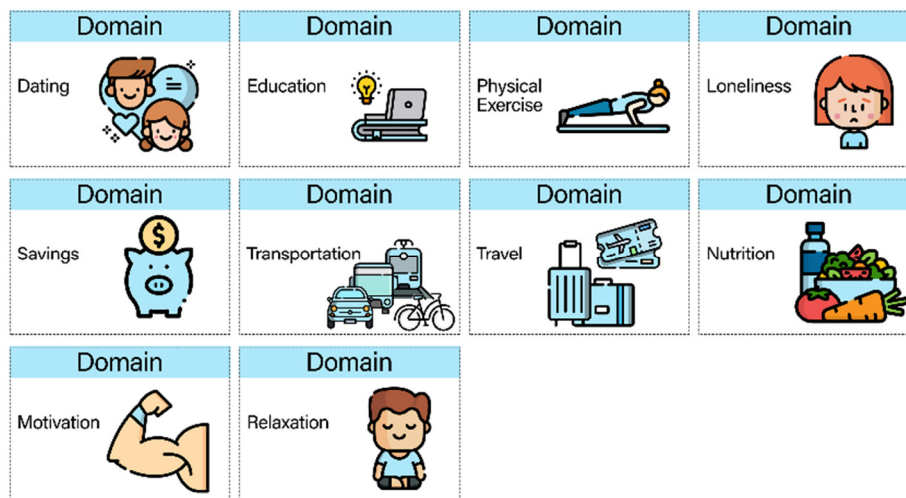


Figure 16: Overview domain cards

In Figure 17 an overview of the target group cards is shown. The target groups cards consists of business owners, children, dog owners, elderly, gamers, health care workers, parents, students, volunteers and athletes. To choose these target groups another small brainstorm was done. Going into the brainstorm thinking about target groups seen in everyday life. Alongside target groups most often talked about.

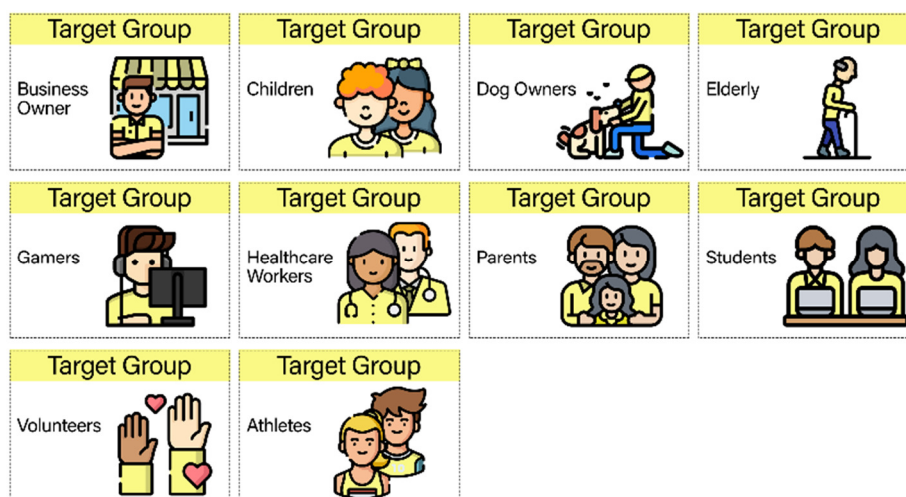


Figure 17: Overview of target group cards

In Figure 18 an overview of the technology cards are shown. The technology cards consists of twice application, Augmented Reality, drone, Internet of Things, smart TV, smartwatch, Virtual Reality and twice website. To choose the technology card was less of a brainstorm. It was looking around at what technology is mostly used by people and what are technologies rising in popularity. However, finding enough equipt technologies was harder

than expected. Together with the fact that applications and websites are widely used, there was chosen to use the application and website twice in the technology card deck.

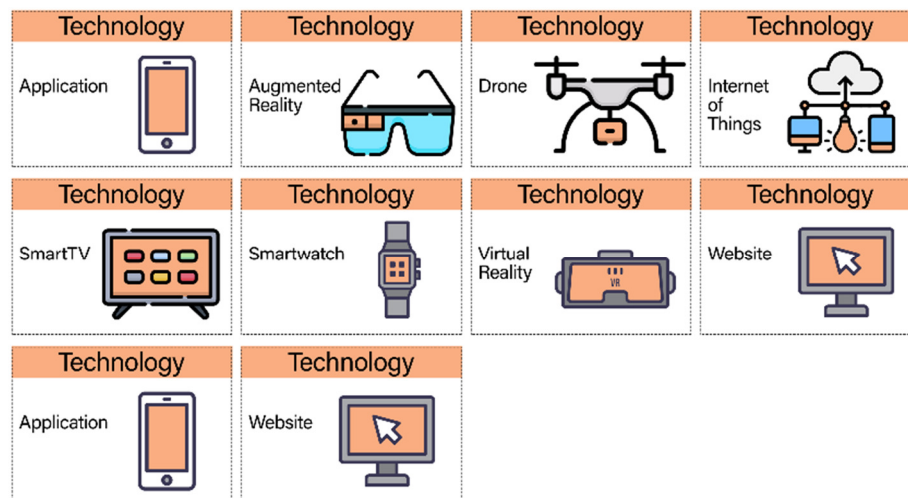


Figure 18: Overview of technology cards

6.1.2 Design lectures

For the design of the lectures given within the workshop were a few options. The lectures could be given accompanied by slides or handouts. Another option for the lectures was using some kind of interactive quiz. However, since the workshop is already using the card exercises it is important to keep the lectures clear and quick. To do so there was chosen to give the lectures without any supporting materials. The lectures will show the models and the lecturer teaches the according content. The models are created according to the example and fit the colour theme of the model cards to make it more cohesive.

6.2 Content workshop

6.2.1 Introduction

The introduction of the workshop will take around thirty minutes. In these thirty minutes, the workshop will be introduced and the goal of the workshop will be explained. Below is the information that should be told in this introduction.

6.2.1.1 What is the workshop about

This workshop is meant to introduce modelling. This workshop is split up into an introduction, 3 workshop blocks, a long break and a conclusion as shown in Figure 19: Workshop schedule. Between every block is a 15-minute break. The workshop blocks will be as follows. Every block starts with a lecture of 30 minutes introducing two kinds of models. Then an exercise will be done for 40 minutes with afterwards a feedback round of 35 minutes.

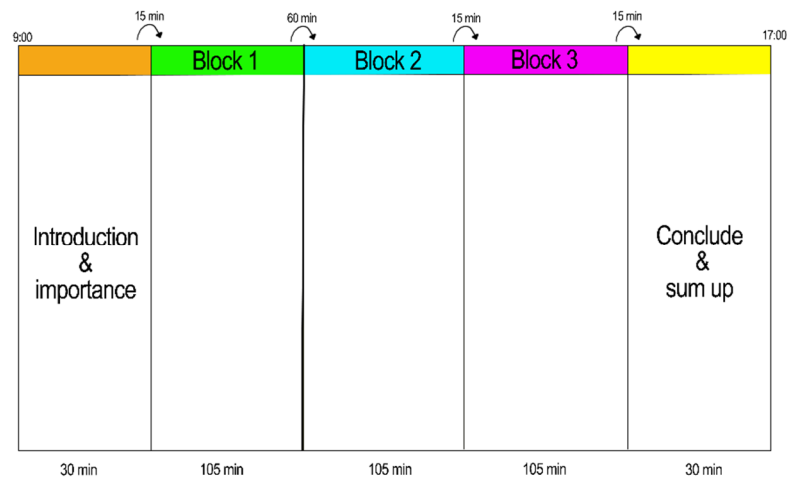


Figure 19: Workshop scadual

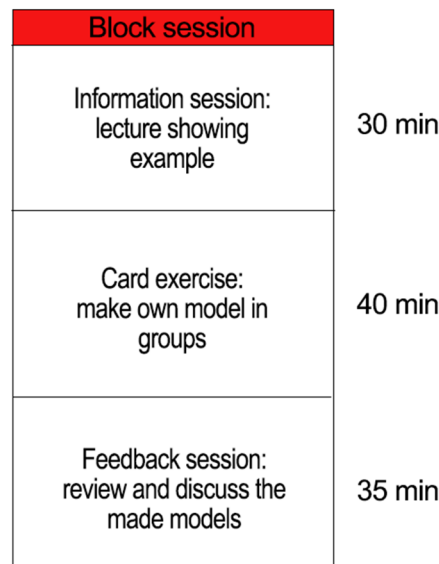


Figure 20: Scadual of block session

The lectures of each block will differ per topic but are mostly the same in structure. Every model in the workshop will be explained via an example diagram. This diagram will show a simple introduction to the diagram to give the students a basic understanding of the model structure. Before introducing the example model the goal of that specific model will be explained. Then, the way of modelling the model will be explained via the example. This could be done in two ways. Either by presenting the diagram and explaining it or building the diagram while explaining. Both have their advantages and depend on the teaching style. The advantage of presenting the diagram and explaining it is that the participants start by seeing what the diagram should look like in the end. This could create a clear goal. The advantage of building the diagram while explaining is that the participant see the process of

how to create a diagram from start to finish. However, some teaching style might fit a lecturer better than the other because of the way they explain or construct their content.

The exercise goes as follows: There will be teams of 2 to 3 people. Every group has to pick a card from every category (Model, Domain, Focus group & Technology). For the model cards, there is an option for one of the two models talked about in that block. With these cards, the teams have to come up with an idea. Then they have to model with the chosen model card. When the teams have modelled their idea it is time to discuss and give feedback to each other. To do so, two groups are going to join each other to explain what they did and how they came to the model they have now. This then will be discussed with some possible feedback or questions from the other team and vice versa.

6.2.1.2 Information on modelling

Modelling is meant to give an overview of a system or software program. It should address the entire system or program. This means the algorithms at the backend as well as the interfaces and interactions with other systems and programs. In other words, a model is a simplified representation of part of a system from a particular view. The best-known modelling language is UML (unified modelling language).

It is important to know that modelling is no exact science. There are no specific ways of doing it right. When modelling there could be different outcomes. The model should visualize the idea thought of to make a clear overview of that idea. This overview can then be used to explain, discuss and execute the idea. When modelling certain models there are guidelines to these models however how exactly the model should turn out is up to the person modelling the idea.

Therefore, in this workshop, you will learn the mindset of modelling and the guidelines instead of the perfect way of modelling. You'll get the basics of modelling. Models can get very detailed and complicated but for today the goal is to get a basic understanding.

6.2.1.3 Examples and real-life experiences

To motivate the participants to see the importance of modelling a few real-life experiences are told. Below a few experiences are shown. However, if the lecturer has other examples or examples of their own they can add those to the introduction instead.

- I. In software development, proper use of models is a must if you want to get your point across. Modelling makes sure that you cover full workflows. This person has worked at 3 IT companies and while none cared about proper UML, it was always appreciated if you could clearly communicate your ideas through models.
- II. A Technical Computer Science student had a course in design in her second module of the first year. This course was not her favourite course since she thought it was very boring and not efficient in the programming world (it takes a lot of time). However, in module four of

that year, it became more and more clear how efficient models could be. She and her project group used models leading up to everyone in their project group knowing what had to be done, what the program had to do, and how the database was designed. In the end, their project was of good quality without a lot of unnecessary lines of codes. Every object in their code was used efficiently, and because of the use case models they used, they could make the interface of their project much better. All by all, the models helped them on making a product that satisfies the clients and they all knew how the program was going to work. And this is time-saving in the end.

- III. Another good example is not so much an experience from someone but the design differences of two educational systems. These two systems are Canvas (Figure 21) and Horus app (Figure 22). The Horus app looks clear and it user friendly. As a user, it is clear where you need to go and what everything means. When looking at this app it is clear that the user was kept in mind when designing. It shows everything a user needs of the system. When looking at the Canvas page, on the other hand, it looks less user friendly. It is way less clear where the user should go and what everything means. When going through all the options it is unclear and the essentials for the users are not there or somewhere hidden. Canvas is very unintuitive, where there models to be used when designing this system this would not have been the case.

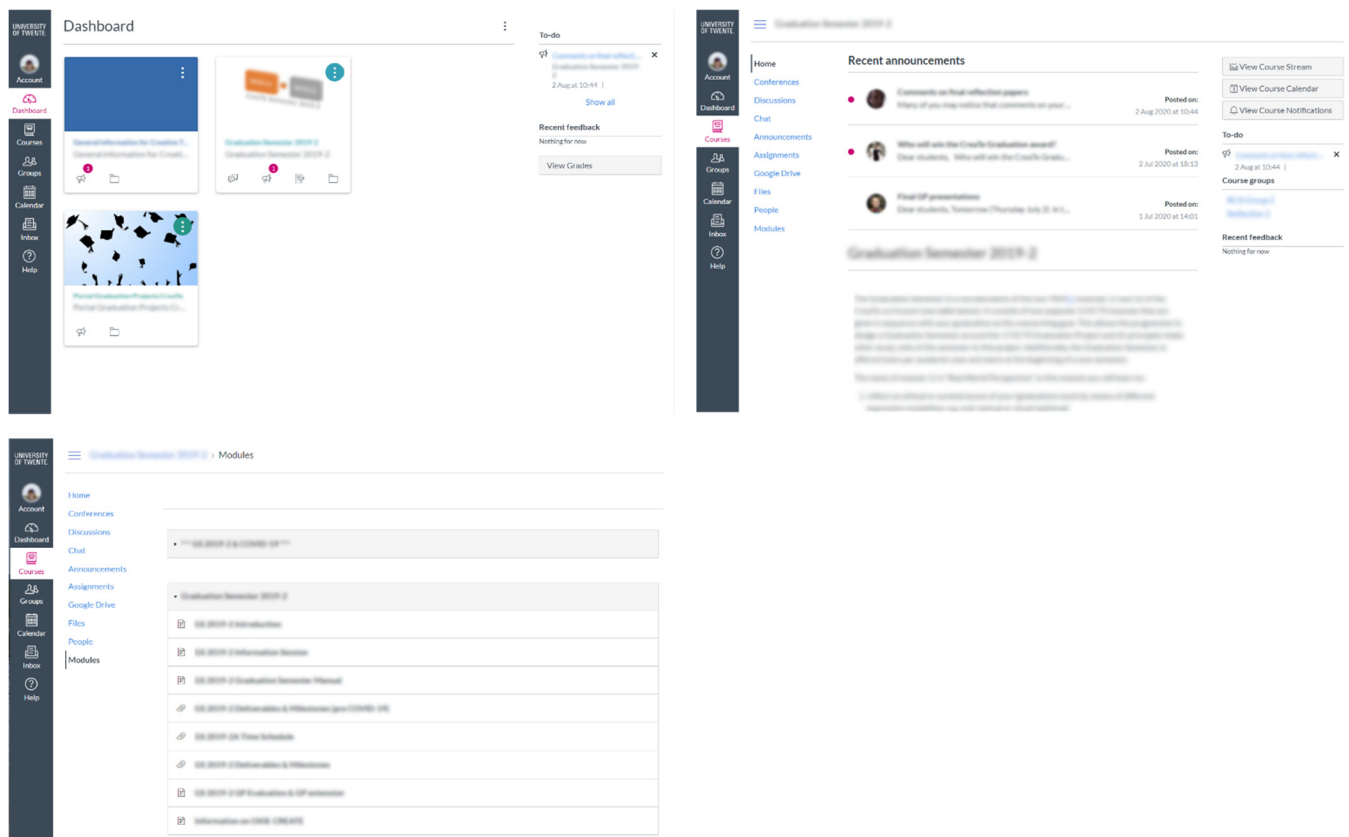


Figure 21: Canvas system

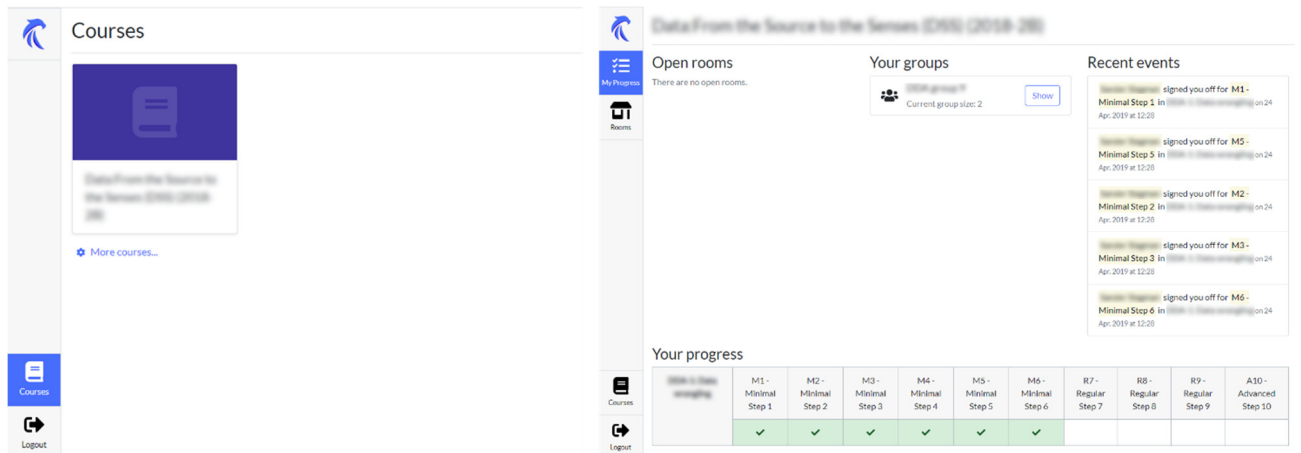


Figure 22: Horus App

6.2.1.4 The Example that will be used to introduce the models

The example that will be used for this workshop to explain the models is shopping for a laptop online. The idea is that the examples given of the models show the system and process of shopping online for a laptop.

6.2.2 Block 1: Activity Diagram & State machine

6.2.2.1 Activity diagram

The example that will be used to explain the activity diagram is shown in Figure 23.

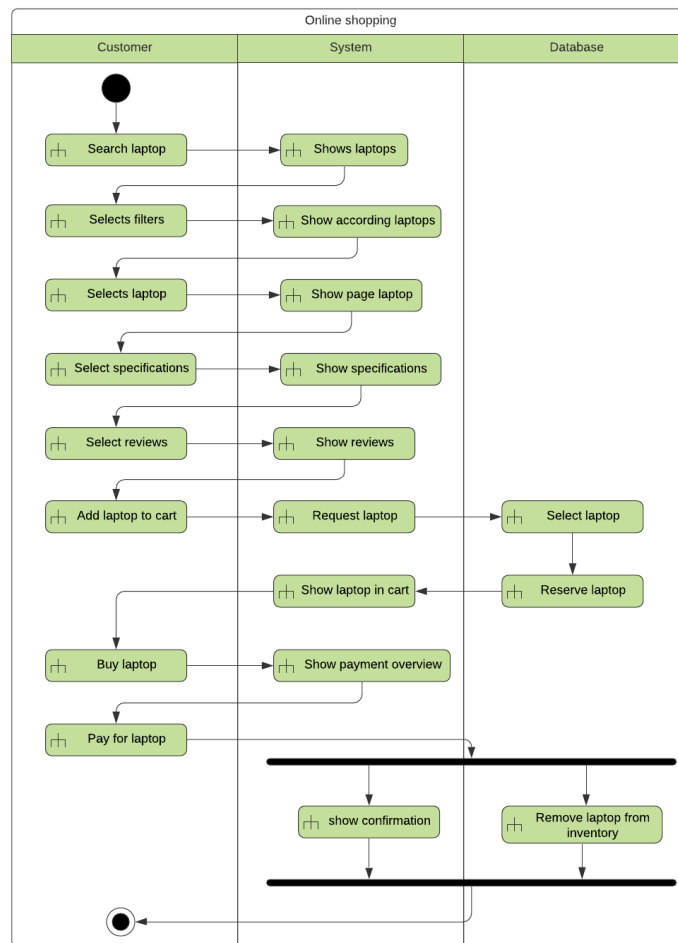


Figure 23: Activity diagram

An activity diagram shows the main flow of activities from a system and its actors. The series of activities are carried out by their actors. Actors can be persons or systems. In this case, the actors will be the customer, the online shopping system and the database. The diagram starts with a start symbol from there on out the activities in the process are stated. The arrows show the control flow. In this example the customer starts with searching for a laptop, then the system will show laptops. Later when the customer is about to buy the laptop the database shows activities. The last two activities are shown differently; this is to show that these activities are down simultaneously, not one and then the other. In the end, there is a stop symbol to show that the activities are done. It is possible to create various separate activity diagrams to construct a model.

6.2.2.2 State machine

The example that will be used to explain the state machine diagram is shown in Figure 24.

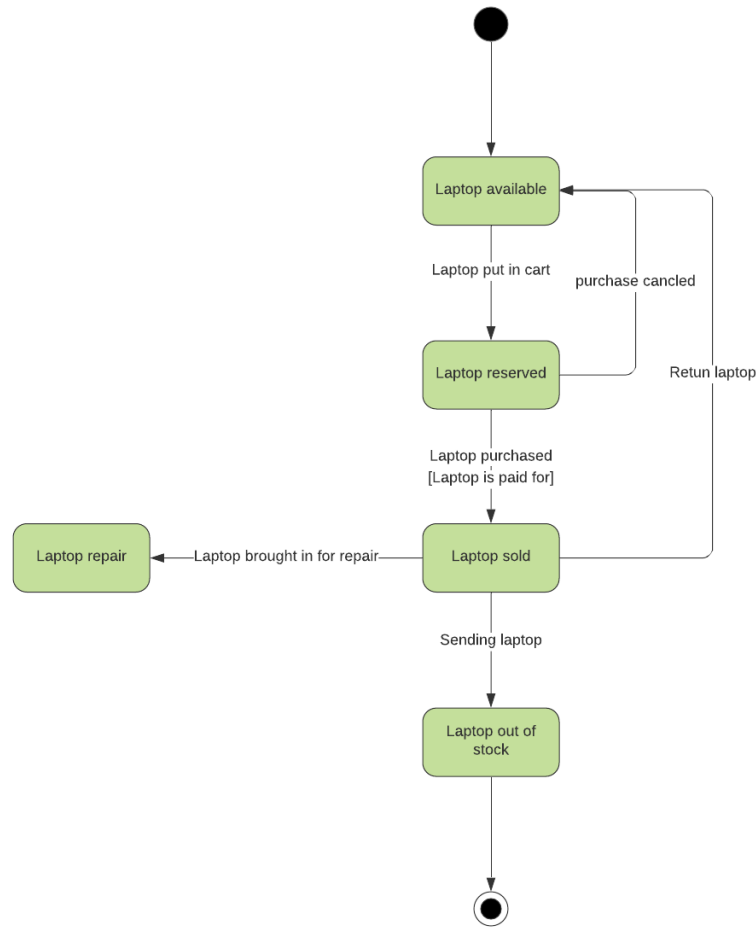


Figure 24: State machine diagram

A state machine diagram denotes possible states and transitions between states for an object of a particular class. A transition is triggered by an event which causes the object to move from one state to another. An object cannot be in-between states, an object always has to be in some state. In the example, this is shown since the laptop is either available, reserved or sold. A transition can be constrained by a guard. In the example, this is in the transition from laptop reserved to laptop sold. The guard in this case is the condition "laptop is paid for". Since you have to pay to buy a laptop. A state machine diagram has always on start symbol, denoted as a black dot in the example. A stop symbol does not always have to be there since it could be a circular diagram. In case of the example, there is an end symbol denoted as the black circle with a black dot in the middle. There is also the possibility of combining states into one block which could be exited. It is also possible for two things to happen at the same time, this is called parallelism.

6.2.3 Block 2: Requirements list & Use-case diagram

6.2.3.1 Requirements list

The example that will be used to explain the requirements list is shown in Figure 25.

Number	Requirement description	Requirement source	Priority
1	Can search for specific products	Customer	1
2	Can compare products	Customer	3
3	Can filter on specifics	Customer	2
4	Can save products	Customer	3
5	Secure payment	Cusotmer	1
6	Safely store personal details	Customer	1
7	Add and remove within inventory	Administrator	1
8	Notify of shipment	Customer	2

Figure 25: Requirements list

The requirements list is used to create a clear overview of what users want from the system. This is simply done by putting those requirements into a clear table. Often there will be added extra information to this table. In case of the example requirement source and priority of the requirement are added. Often useful information to give more details on the requirements are added. Requirement source is the user the requirement is coming from. The priority is the importance of achieving the requirement.

6.2.3.2 Use-case Model

The example that will be used to explain the use-case diagram is shown in Figure 26.

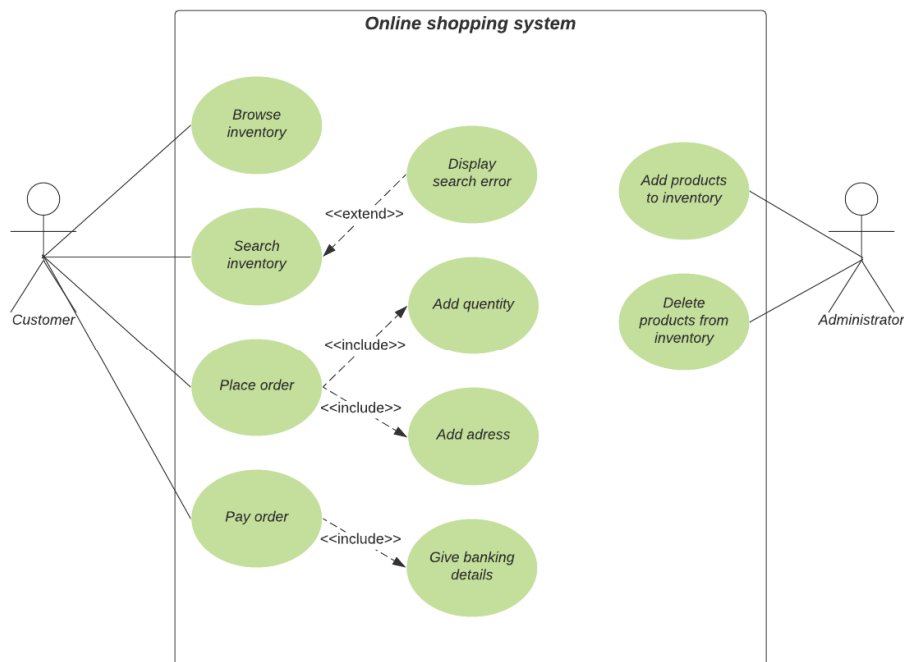


Figure 26: Use-case diagram

A use-case diagram is a top-level description of a system's functionality. These diagrams are focused on users since they are easy enough for non-IT staff to understand. They are meant to talk to the user and understand the user and their actions. A use-case diagram consists of actors and use-cases. Use-cases are circular with a described imperative sentence and are often linked to a single actor. In this example there are two actors, one is the customer the other the administrator. There are shown three included use-cases denoted as a dotted arrow towards the included use-case. These are use-cases that are executed at the same time as the base use-case. In case of this example when placing the order the quantity and address should be added at the same time. Additionally, an extend use-case is shown. The extend use-case is a use-case that sometimes will be executed but not all the time. In this example, there might be shown a search error after searching for something specific. As shown in the example multiple actors can interact with the system. In the example, alongside the customer, the administrator can add and delete inventory within the system. Often in combination with the use-case diagrams, there are other tools used to give more details on the system and the working with the users. A few of those are a glossary, requirements list, actor list and use-case descriptions. Use-cases can be made as detailed as needed, however the more details the less users will understand.

6.2.4 Block 3: Sequence Diagram

6.2.4.1 Sequence diagram

The example that will be used to explain the sequence diagram is shown in Figure 27.

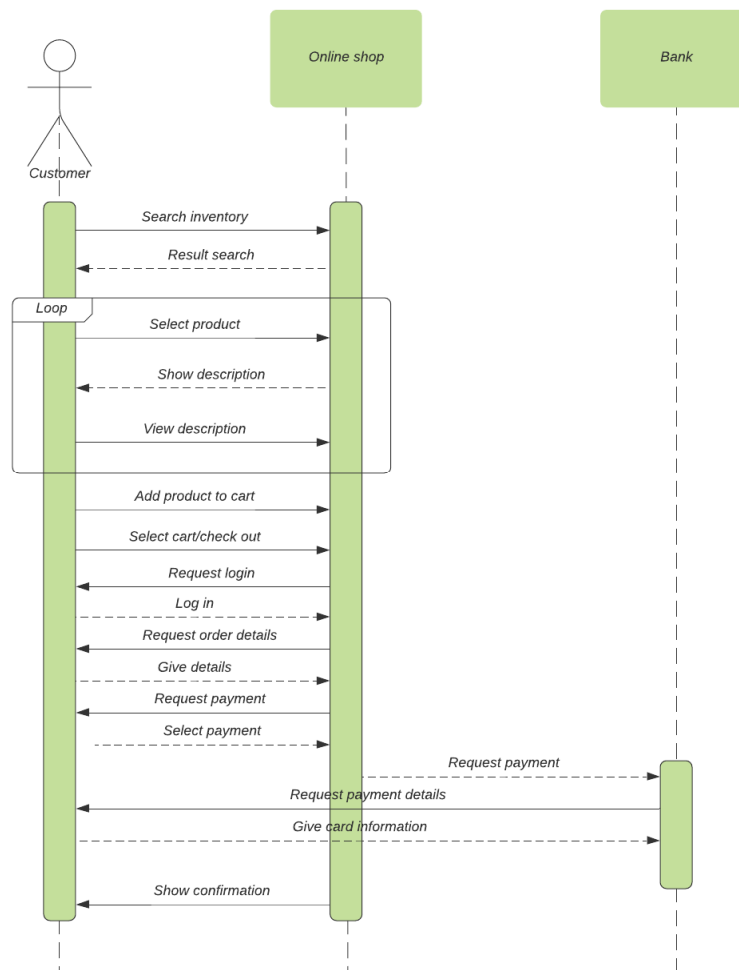


Figure 27: Sequence diagram

A sequence diagram describes how a use-case is executed. They describe the interactions between objects and actors. These interactions are called messages and are denoted by arrows. The lifeline of an object is denoted as a dashed vertical line and an activation is denoted as a vertical bar. The activation means when an object is activated so when it is making interactions. Normal messages from one object to the other are denoted as solid arrows while return and create messages are denoted by dashed arrows. In this case, the actor will be the customer and the objects are the online shop and the bank. The first message is from the actor to the online shop searching the inventory where the online shop gives a response by showing the results of the search. Then a loop is shown meaning these interactions could be done multiple times. In this case, it is selecting a product and looking at the description of that product. These kinds of blocks can be used in different ways, not only for loops but also for alternatives. Later in the diagram, there are interactions between the

bank and the online shop as well as between the bank and the customer to manage the payment. Afterwards, all the activations stop and thus the system interactions are done.

6.2.5 Conclusion

Of importance when giving the conclusion of the workshop is to take into account the hours the students have spent so far on the workshop. The workshop is fun and engaging but knowledge heavy. This could mean that the students are tired in the end or no longer interested. So, when giving this workshop there should be anticipated how the students are feeling at this point in the workshop. If it is obvious they are tired and/or no longer interested the conclusion should be kept short. If the students still seem to have some energy left, go into more depth with the conclusion.

The conclusion is about reminding the students about the importance of modelling and that modelling/software design is not an exact science. Alongside that, a quick overview will be given on what the students learned in this workshop: recapping the models thought and the often heard feedback in the sessions. Additionally, there should be told that there are way more models than the five explained in this workshop. For example: storyboards, Data flow, network diagram, timeline, or tree diagrams. In this workshop mostly UML diagrams are shown since those are the most common diagrams however some non-UML models can be used. With modelling, the goal is to create a clear overview of the system that everyone within a team can understand. Lastly, trophies will be given out for the best model, the most original design, and the best idea. Alongside these trophies, every student will get a small certificate.

Chapter 7 – Evaluation

In this chapter, the final prototype of Chapter 6 – Realisation is evaluated. This will be done by evaluating the results of the survey conducted and by analysing the final requirements.

When analysing the final requirements there will be decided whether or not those requirements were met.

7.1 Survey

The survey had 4 respondents from the 6 initial send out survey links. The respondents were categorised into the initial stakeholders. The respondents were student, Teacher and Teaching Assistant. The survey was split up into subsections as follows: the introduction, the workshop, the cards, the structure of the workshop, the blocks, the conclusion and the final remarks. The whole survey can be seen in *Appendix B – Survey* and the exact responses in *Appendix C – Responses survey*. In the survey, the workshop was explained as explained in *Chapter 6 – Realisation* along with questions concerning the content and design of the workshop.

7.1.1 The workshop

In the first part of the survey, the workshop and the introduction of the workshop is explained. The feedback on the introduction and the explanation of the workshop was very positive. The description was clear and had the right information. There was however still some confusion. One command said “I don’t completely follow how the example case (laptop shopping) ties into the actual exercises (the card ‘game’)” another did not yet see how to get from an idea to a model.

7.1.2 The cards

The initial reaction on the card exercise and the cards were positive as well. According to the respondents, the card exercise was clear however, they were a little less positive about how engaging and enjoyable the card exercise would be. Nonetheless, the participant seemed excited about the exercise idea. There were a few suggestions on adding other diagrams. One person mentioned adding the class diagram and another would like to see more free-style diagrams.

Additionally, some questions were asked about the separate theme cards. The respondents all found the cards clear however, when asked if things were missing there were a few commands. For the *domain cards*, it was suggested to add things like culture, music, inclusion or entertainment. For the *target group cards*, it was suggested to add vulnerable groups like addicts, visual impaired and refugees. Other suggestions for the *target group cards* was a cultural person group like an artist or an actor. Suggestions for the

technology cards were adding smart speakers like Alexa or Google assistant, Domotica (home automation) or Arduino.

One of the respondents was concerned that the third time of doing the card exercise will become tedious. Especially because they get to see the different model types from the other groups in a rather long feedback round. If the point of this workshop isn't to learn the exact rules of UML (which is fairly difficult and needs much more exercise), then 2 rounds is enough to both keep it engaging and to get the point across. Another respondent voiced to make sure to motivate the participants to do the exercises since design can be very boring and confusing. A compliment was also received by a respondent: "A really clear way of doing it, rather than getting lost in the details. I like it!". Lastly, a participant mentioned that the cards over-specify the exercise. They expected that most of the ideas coming from the exercise would be boring and do not result in more than the expected. They propose to remove one of the theme to leave more room for creativity or add some very unexpected domains or target groups.

7.1.3 The structure of the workshop

A few questions were asked about the structure of the workshop. The first question asked if the structure of the workshop was clear to the respondents. They responded that the structure was clear. However, they were more divided on whether or not there were enough breaks within the time table. Nonetheless, none of the respondents was very disapproving on the amount of time for the breaks. Additionally, the respondents were divided on whether or not there is enough time for every part of the workshop in the schedule. Nonetheless, they still are very positive that it is. One of the respondents was concerned about the time cut out for the exercise. They said that there should be looked at how long every group is working on the exercise. If the groups need to make a model from scratch, then 40 minutes are enough to make a decent model. But as the respondent stated earlier, design can be confusing and boring if people don't understand it very well or don't see the point of the design, then 40 minutes is a bit to much time. If there are enough breaks? The participants think so, but they encourage to do a longer break after 2/3 blocks. Since the outcome of several resources give that in a longer break the brain takes the time to process all the information that you covered in the workshop.

7.1.4 Blocks

The feedback on the content of the block session was very positive. According to the respondents, the content of the lectures were clear, just as the use of the example diagrams. The response on the accuracy of the diagrams was very diverse. Nonetheless, the response was mostly positive, sometimes neutral.

Additional feedback was sent in separately from one of the respondents. The respondent had seen a mistake in the *state machine diagram*. In the current example diagram of the *state machine diagram*, there is a state 'laptop repair' which never goes to another state. The respondent noted that this is not valid and it might need another look to make it correct.

When asked for suggestions for a 6th diagram two respondents had respond. One mentioned setting up an extended use case description. The other mentioned that adding a structural diagram, like a component diagram, would be necessary.

7.1.5 Conclusion

The content of the conclusion was clear according to the respondents. They, however, were not as positive about giving out trophies and certificates. Someone suggested replacing the trophies with something small students care about like candy, random trophies are overrated and not quite as motivating as we think according to this person. This person also wondered what the purpose of the certificates are. Since it is only useful if the workshop is good enough of an exercise to for example add to LinkedIn certificates.

7.1.6 Final remarks

Sadly there was a mistake made when making the survey. Because of this, the question: "Do you have any experiences with modelling/software design that have shown the importance or influence of models? If so, please explain what that experience was." was not possible to answer by the respondents. However, a few of them did with the later question: "Do you have any final remarks on the workshop or the thesis?". A few of the answers on that question were about giving compliment on the workshop and wishing good luck on the thesis.

The experience that was mentioned is: "In software development, proper use of models is a must if you want to get your point across. Modelling makes sure that you cover full workflows. I have worked at 3 IT companies and while none cared about proper UML, it was always appreciated if you could clearly communicate your ideas through models. " Most of the feedback, however, was on the models. One of the respondents was missing a use case within the *use-case model* that interacted with multiple agents. The same respondent preferred a MoSCoW method for requirements instead of the numbers used in the requirements list.

Lastly, one of the participants whos research for several years was focused on modelling for formal methods advised to not stick too much to the existing UML versions of the models. Instead take the UML models as an inspiration and allow freestyle models.

Since the point is that people start to reflect, identify the structure, and make that explicit, and not to struggle with definitions of different categories of models.

7.2 Requirements

To further evaluate the prototype there is looked at the requirements set in 5.5 Final requirements and whether or not those are achieved. In Figure 28 the requirements are again shown by using the MoSCoW method. However, this time with additional checks. Those checks show if the requirements are achieved, not achieved or not yet fully evaluated. The latter is the case for the following requirements: must have understandable but detailed example models, should have enough breaks, and should have enough time for every part of the workshop as shown in Figure 28. For these requirements, it is unsure if they are achieved since they were not tested. Looking at the research and the reasoning for the design choices it could be said that they were achieved. However, it is not possible if the real-life experience of the workshop would have shown the same results as envisioned. For that reason, those requirements are set to unsure.

Most of the higher priorities are met, except for having understandable but detailed example models, which is unsure. Most of the Should requirements are also met or unsure, concluding that the prototype had done very well. The only requirement missing in the prototype is adding teacher assistants to the workshop.

	Priority	Requirement	Achieved
M	Must	have diagrams with an accurate presentation of that particular diagram.	✓
		have diagrams with an accurate presentation of the example.	✓
		have understandable but detailed example models.	~
		be a workshop that can be held in a day.	✓
S	Should	have enough breaks.	~
		have enough time for every part of the workshop.	~
		have distinctive theme colours.	✓
		have a cheerful designs on the activity cards.	✓
		have an engaging and challenging card exercise.	✓
C	Could	have opportunity to give and receive feedback for the participants.	✓
		have the same colour pallet for the theme colour and the design on the exercise cards.	✓
		have teacher assistants for the workshop.	✗
W	Won't		
		take into consideration the workingfield as stakeholders.	—
		have more than 6 models explained.	—
		have non-UML models.	—





 Achieved
  Has not been evaluated fully
  Not Achieved
  Does Not Apply

Figure 28: MoSCoW requirements achievement

Chapter 8 – Conclusion

This chapter will summarize the research and answer the research question mentioned in Chapter 2 – State of the art.

When starting this research the goal was to look into better communication between software engineers and non-software engineers. From gathering background information the research question was conducted which became: *How do we make students more aware of modelling and software design?* To answer this question a stakeholder analysis and brainstorm was done in Chapter 4 – Ideation. This resulted in the initial requirements which were elaborated in Chapter 5 – Specification into the final requirements using the MoSCoW methods for prioritising the requirements. With these requirements, the final prototype was made in Chapter 6 – Realisation, which then was evaluated.

Based on the results of the evaluation in Chapter 7 – Evaluation we can say that the prototype was a success. However, this is only based on a survey conducted not on actually testing the workshop with participants. Since the research question was: *How do we make students more aware of modelling and software design?*, we can say the prototype could be the start of making students aware of modelling and software design. But to fully answer the research question additional research is required.

Chapter 9 – Future work

In this chapter, the possibilities of future work expanding on this thesis will be discussed.

The first thing that would be important to future work would be the testing of the workshop. In this thesis, the workshop was only evaluated by a survey. When giving the workshop more specific feedback could be gathered on the design and content of the workshop.

Recommendation for this evaluation would be to hold the workshop as stated in *Chapter 6 – Realisation*. When giving the workshop the evaluation should be done in different ways. One have evaluation forms for the participants and the lecturer(s) to here about what they thought regarding the content and the design. Additionally, have people observe the workshop. When giving the workshop they can look at expressions of people and how the participants react to aspects of the workshop. Another option would be to hold interviews afterwards or in between blocks.

Another possibility for in the future is to look into giving the workshop to other students (non-Creative technology students). Or even expanding it towards the working field, using the stakeholders mentioned in *Chapter 4 – Ideation*: developers and management. To do so most of the structure and content of the workshop could probably stay. However, it might be wise to check the stakeholders interest and see if changes should be made accordingly.

Another suggestion for future work is to look into having the workshop as a mandatory subject for students, instead of the voluntary workshop created in this thesis. The advice would be to split the workshop up into separate lectures spread through a quartile instead of everything in one day. When doing so more content could be put into the workshop and there would be more time to focus on the topics. This could easily be done since the current design is split up into blocks. These blocks could make separate lectures given over time.

Additionally, the workshop could also be done for bigger groups. This could be done by having the different block sessions given simultaneously. This would mean, having 60 to 90 participants per workshop. However, to do so a big room is needed since the introduction and the conclusion would still be held for everyone at the same time. Thinks to keep in mind when holding this workshop for bigger groups is to check if the order of the models still work if some participants will have a different order than others. Additionally, more lectures and possible assistants would be needed to give a workshop that much bigger. A workshop with this amount of participants could be done when the workshop is part of a curriculum but will not be split into separate sessions.

Lastly, the cards and content of the workshop could be worked out even further. There are a lot of possibilities like adding more cards and models. But also expanding the

content. This could mean adding non-UML diagrams to the workshop or giving more detailed information on the models in the content.

Appendix A – Interview guide

Explain and show the workshop idea, set up and design.

Questions to ask

Do you have feedback or input on the workshop layout?

Are the themes of the cards a good combination?

Do you have feedback or input on the workshop layout?

What are the important things for in the lecture?

What would be important/interesting models to add to the workshop?

Would you change models that are now in the workshop design?

Do you have tips on giving a mini-lecture?

Do you think the schedule is accurate or is there too little or too much time?

Do you know examples of the positive influence of modelling from experience or in the work field?

Is the content of the introduction clear?

What would you add or remove from the introduction?

Is the content of the concluding lecture clear?

What would you add or remove from the concluding lecture?

Appendix B – Survey

Evaluation of Workshop on Modelling

This form is for the evaluation of a workshop on modelling. This workshop is designed by Femke Jansen for the graduation thesis of Creative Technology. The goal of the graduation project was to create more awareness and knowledge towards modelling and software design. To do so a workshop was created. In this evaluation, the workshop will be explained and partly shown. For now, the workshop is focused on Creative Technology students.

The evaluation will follow the workshop design. Starting with an introduction, followed by the learning blocks and finishing off with a conclusion. Each section will show the workshop design, the content and a few questions on the design and content.

Introduction

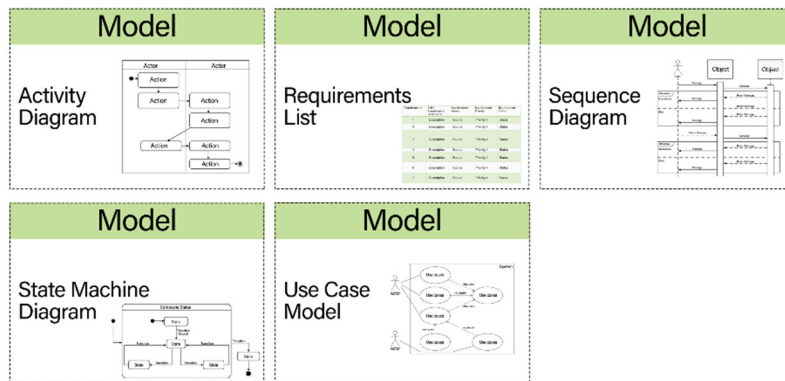
The introduction of the workshop will be around thirty minutes. In these thirty minutes, the workshop will be introduced and the goal of the workshop will be explained. Below is the information that should be told in this introduction.

What is the workshop about?

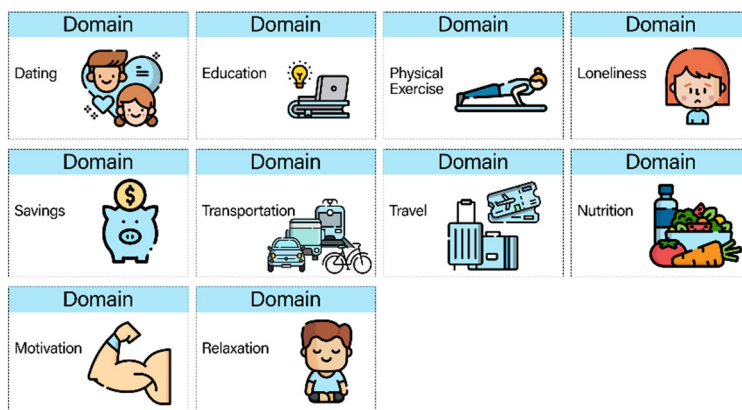
This workshop is meant to introduce modelling. It is split up into an introduction, 3 workshop blocks, a long break, and a conclusion. Between every block is a 15-minute break. The workshop blocks will be as follows: Every block starts with a lecture of 35 minutes introducing two kinds of models. Then, a card exercise will be done for 40 minutes with afterwards a feedback round of 35 minutes.

The card exercise goes as follows: Teams of 2 to 3 people will be made. Every group gets a card from every category (Model, Domain, Target Group, and Technology). For the model cards, there is an option for one of the two models talked about in that block. With these cards, the teams have to come up with an idea. Then they have to model with the chosen model card. When the teams have modelled their idea it is time to discuss and give feedback to each other. To do so, two groups are going to join each other to explain what they did and how they came to the model they have now. This will then be discussed with some possible feedback or questions from the other team and vice versa.

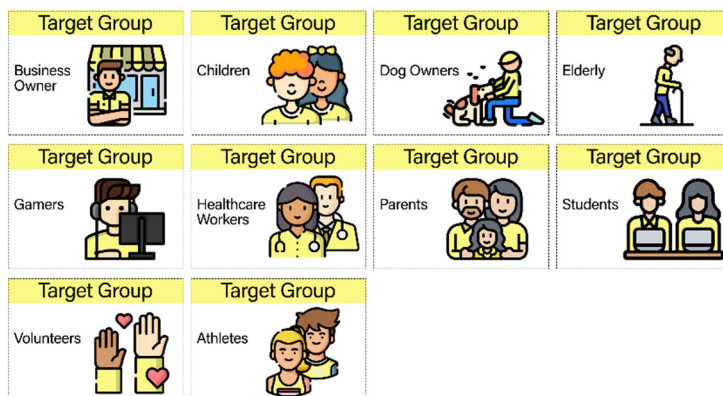
Model Cards



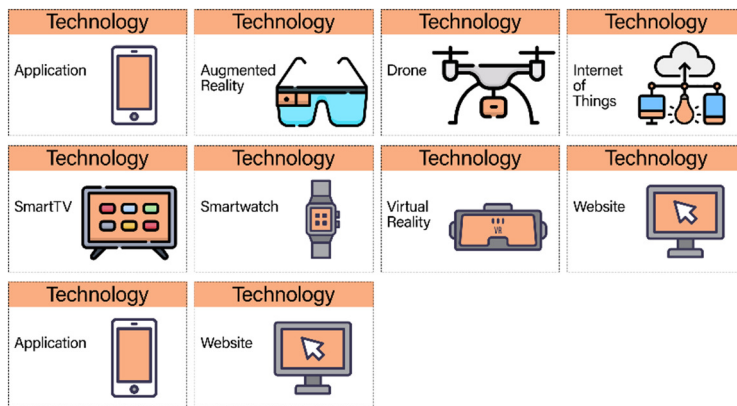
Domain Cards



Target Group Cards



Technology Cards



In the introduction of the workshop, the information will be given on modelling. This information is mostly the basics and emphasis on the fact that modelling is no exact science and that there are different ways of modelling. This means the workshop will give the students the mindset of modelling instead of the rules.

Then, the importance of modelling will be mentioned, this will mostly be shown by explaining a few real-life experiences of positive effects with using models.

Lastly, the example situation that will be used for the workshop will be explained. In this case, it will be shopping for a laptop online.

The Workshop

The above description gives a clear overview of the workshop.

Disagree 1 2 3 4 5 Agree

The above description has the right information for an introduction.

Disagree 1 2 3 4 5 Agree

Additional feedback on the introduction.

The Cards

The card exercise is clear (I know how to do it).

Disagree 1 2 3 4 5 Agree

The card exercise is engaging.

Disagree 1 2 3 4 5 Agree

The card exercise seems enjoyable to do.

Disagree 1 2 3 4 5 Agree

The model cards are clear.

Disagree 1 2 3 4 5 Agree

I am missing ... with the model cards.

The domain cards are clear.

Disagree 1 2 3 4 5 Agree

I am missing ... with the domain cards.

The target group cards are clear

Disagree 1 2 3 4 5 Agree

I am missing ... with the target group cards.

The technology cards are clear

Disagree 1 2 3 4 5 Agree

I am missing ... with the technology cards.

Additional feedback on the card exercise.

Structure of the workshop

The structure of the workshop is clear

Disagree 1 2 3 4 5 Agree

There are enough breaks in the workshop

Disagree 1 2 3 4 5 Agree

The structure of the workshop gives enough time for every part of the workshop

Disagree 1 2 3 4 5 Agree

If you disagreed with the structure of the workshopping giving enough time please explain where time is short.

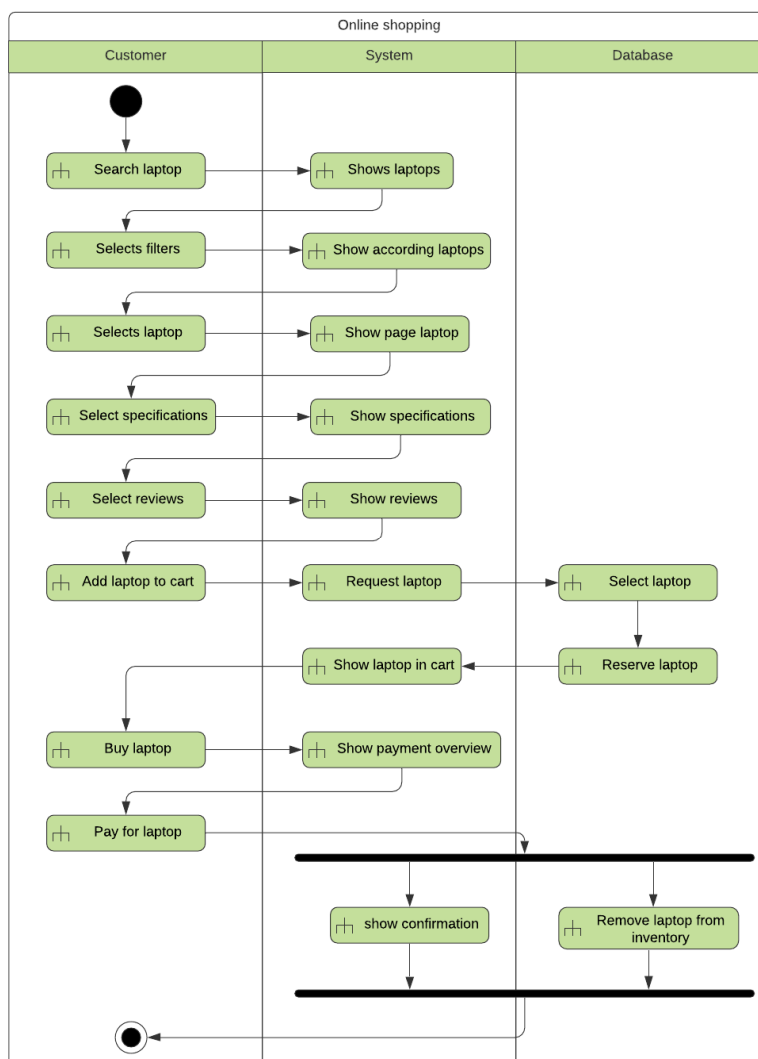
Blocks

As mentioned before the workshop consists of 3 blocks every block being 105 minutes. In the first block, the Activity diagram and the State machine diagram will be introduced. In the second block, the Requirements list and the Use case diagram will be introduced. In the last block, the Sequence diagrams will be explained and used.

Every block has the same structure as mentioned before. The lectures of each block will differ per topic but are mostly the same in structure. Every model in the workshop will be explained via an example diagram. This diagram will show a simple introduction to the diagram to give the students a basic understanding of the model structure. Before introducing the example model the goal of that specific model will be explained. Then, the way of modelling the model will be explained via the example.

The example models below are shown based on the example situation mentioned in the introduction. Using these example models the models will be explained.

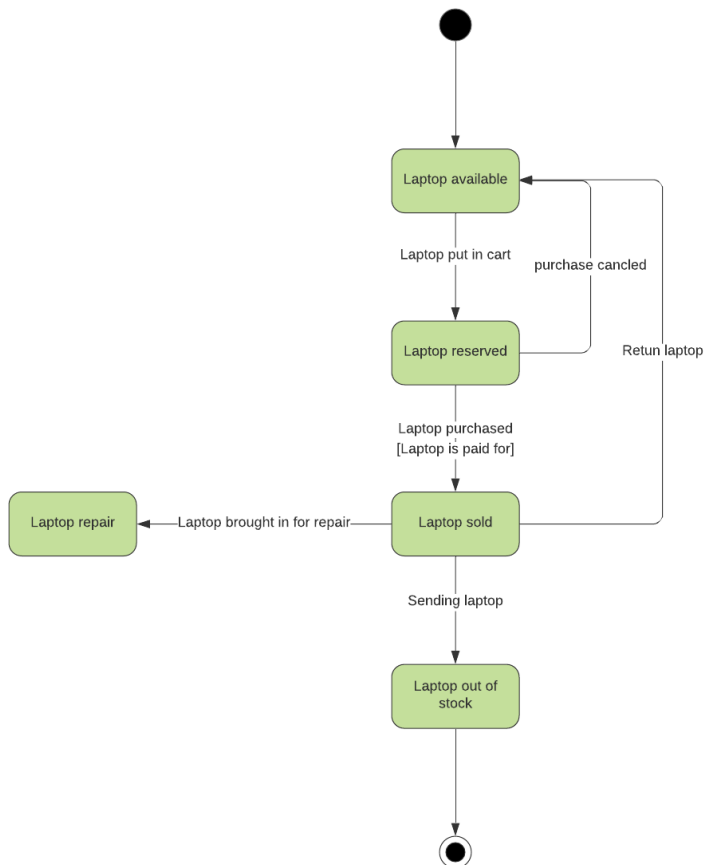
Activity diagram



Activity diagram

An activity diagram shows the main flow of activities from a system and its actors. The series of activities are carried out by their actors.

State machine diagram



State machine diagram

A state machine denotes possible states and transitions between states for an object of a particular class.

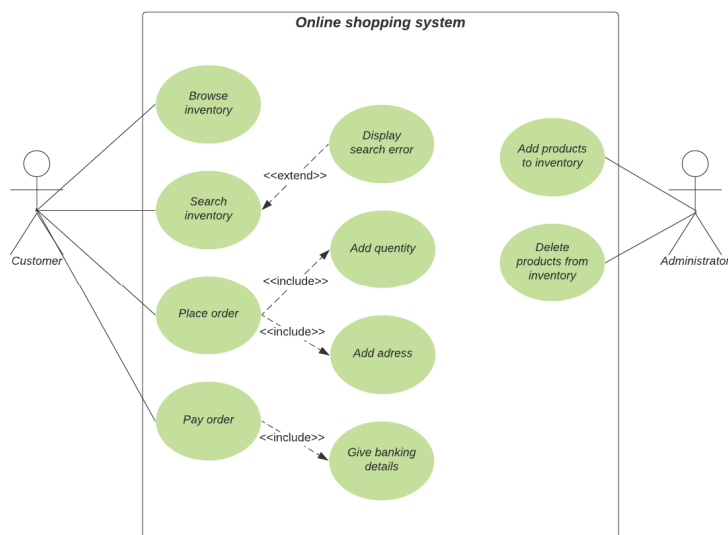
Requirements list

Number	Requirement description	Requirement source	Priority
1	Can search for specific products	Customer	1
2	Can compare products	Customer	3
3	Can filter on specifics	Customer	2
4	Can save products	Customer	3
5	Secure payment	Customer	1
6	Safely store personal details	Customer	1
7	Add and remove within inventory	Administrator	1
8	Notify of shipment	Customer	2

Requirements list

The requirements list is used to create a clear overview of what users want from the system.

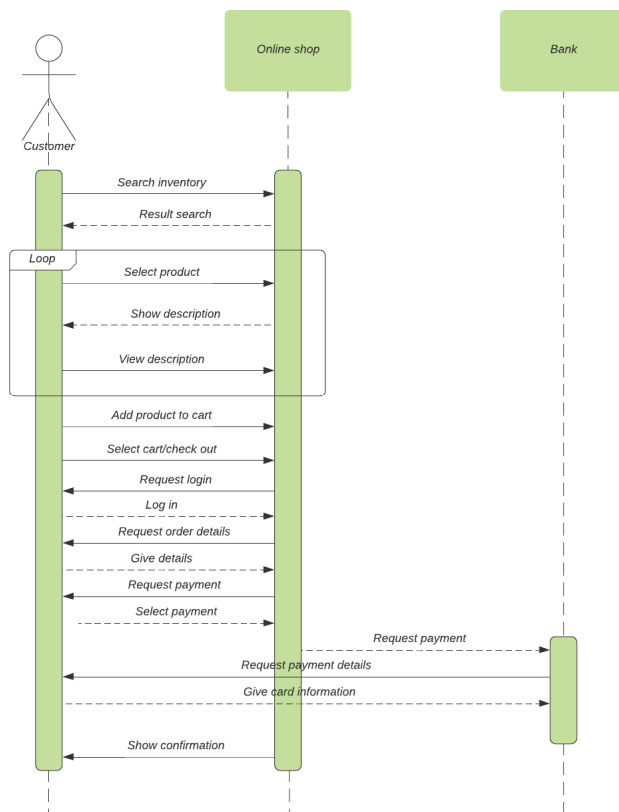
Use case diagram



Use case diagram

A use case diagram is a top-level description of the functionality of a system.

Sequence diagram



Sequence diagram

A sequence diagram describes how a use case is executed. They describe interactions between objects.

The way the example diagrams are used within the workshop is clear.

Disagree 1 2 3 4 5 Agree

It is clear what the content of the lectures should contain.

Disagree 1 2 3 4 5 Agree

The activity diagram is an accurate presentation of an activity diagram.

Disagree 1 2 3 4 5 Agree

The activity diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

The state machine diagram is an accurate presentation of a state machine diagram.

Disagree 1 2 3 4 5 Agree

The state machine diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

The requirements list is an accurate presentation of a requirements list.

Disagree 1 2 3 4 5 Agree

The requirements list is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

The use case diagram is an accurate presentation of a use case diagram.

Disagree 1 2 3 4 5 Agree

The use case diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

The sequence diagram is an accurate presentation of a sequence diagram.

Disagree 1 2 3 4 5 Agree

The sequence diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

In the first draft, a 6th diagram was added. If a 6th diagram would be added to the workshop what diagram fits best?

Conclusion

Of importance when giving the conclusion of this workshop is to take into account the hours the students have spent so far on the workshop. The workshop is fun and engaging but knowledge heavy. This could mean that the students are tired in the end or no longer interested. So, when giving this workshop there should be anticipated how the students are feeling at this point in the workshop. If it is obvious they are tired and/or no longer interested the conclusion should be kept short. If the students still seem to have some energy left, go into more depth with the conclusion.

The conclusion is about reminding the students about the importance of modelling and that modelling/software design is not an exact science. Alongside that, a quick overview will be given on what the students learned in this workshop. Recapping the models and often heard feedback.

Lastly, trophies will be given out for the best model, the most original design, and the best idea. Alongside these trophies, every student will get a small certificate.

The content of the conclusion is clear.

Disagree 1 2 3 4 5 Agree

It is a good idea to give out certificates.

Disagree 1 2 3 4 5 Agree

It is a good idea to have trophies given out.

Disagree 1 2 3 4 5 Agree

I am missing ... in the conclusion

Final remarks

What category do you fall under?

- ☐ Student
- ☐ Teacher
- ☐ Other

Do you have any experiences with modelling/software design that have shown the importance or influence of models? If so, please explain what that experience was.

Thank you for taking the time to fill in this evaluation.

Do you have any final remarks on the workshop or the thesis?

Appendix C – Responses survey

The response of the workshop was 4 out of the 6 people asked to fill in the survey.

The Workshop

The above description gives a clear overview of the workshop.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

The above description has the right information for an introduction.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	100%
5	0%

Additional feedback on the introduction.

- Text is well-written; Some repetitions can be avoided here and there, e.g. “workshop”
- I don’t completely follow how the example case (laptop shopping) ties into the actual exercises (the card ‘game’).
- How to get from an idea to a model is not yet very clear

The Cards

The card exercise is clear (I know how to do it).

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	0%
5	100%

The card exercise is engaging.

Disagree 1 2 3 4 5 Agree

1	0%
---	----

2	0%
3	0%
4	100%
5	0%

The card exercise seems enjoyable to do.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	75%
5	25%

The model cards are clear.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

I am missing ... with the model cards.

- You might want to think about adding the class diagram, but I totally understand that is a bit too much for this workshop.
- More free-style diagrams. How much does the target group know about the modelling methods mentioned?

The domain cards are clear.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	0%
5	100%

I am missing ... with the domain cards.

- Culture, music, entertainment, inclusion

The target group cards are clear

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	0%
5	100%

I am missing ... with the target group cards.

- Maybe a cultural person group, like an artist or an actor
- Some vulnerable groups, addicted (drugs, gaming), visually impaired, refugees ...

The technology cards are clear

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	0%
5	100%

I am missing ... with the technology cards.

- Smart speakers like Alexa (from amazon) or google assistant.
- Domotica
- Website and application are double. I miss Arduino, (sensors, motors, other actuators – perhaps protobox)

Additional feedback on the card exercise.

- A really clear way of doing it, rather than getting lost in the details. I like it!
- Please make sure that you motivate the people in the workshop to do the exercises, design can be very boring and confusing.
- This seems fun and useful. However, I am concerned that the third time will become tedious. Especially because they get to see the different model types from the other groups in a rather long feedback round. If the point of this workshop isn't to learn the exact rules of UML (which is fairly difficult and needs much more exercise), then 2 rounds is enough to both keep it engaging and to get the point across.
- Actually, I think the cards over-specify the exercise. I expect that most of the ideas coming from that are somewhat boring and do not result in more than the expected. So either, I would leave some creative space (e.g. leave one of the three as a joker), or add some very unexpected domains/target groups

Structure of the workshop

The structure of the workshop is clear

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

There are enough breaks in the workshop

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	50%
4	0%
5	50%

The structure of the workshop gives enough time for every part of the workshop

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	25%
4	25%
5	50%

If you disagreed with the structure of the workshopping giving enough time please explain where time is short.

- I think you should look at how long every group is busy with the exercises. If the groups need to make a model from scratch, then 40 minutes are enough to make a decent Model, but what I said earlier, Design can be confusing and boring if you don't understand it very well or don't see the point of design, then 40 minutes is a bit to much time. If there are enough breaks? I think so, but I encourage you to make after 2/3 blocks do a longer break. The outcome of several resources give that in a longer break the brain takes the time to process all the information that you covered in the workshop.

Blocks

The way the example diagrams are used within the workshop is clear.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	25%
4	0%
5	75%

It is clear what the content of the lectures should contain.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	50%
5	50%

The activity diagram is an accurate presentation of an activity diagram.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	25%
4	25%
5	50%

The activity diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	25%
4	25%
5	50%

The state machine diagram is an accurate presentation of a state machine diagram.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	50%

5	50%
---	-----

The state machine diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

The requirements list is an accurate presentation of a requirements list.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	25%
4	25%
5	50%

The requirements list is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

The use case diagram is an accurate presentation of a use case diagram.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	50%
5	50%

The use case diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

The sequence diagram is an accurate presentation of a sequence diagram.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

The sequence diagram is an accurate presentation of the example situation.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

In the first draft, a 6th diagram was added. If a 6th diagram would be added to the workshop what diagram fits best?

- How to set up extended use case descriptions.
- A structural diagram would be necessary – e.g. a component diagram

Conclusion

The content of the conclusion is clear.

Disagree 1 2 3 4 5 Agree

1	0%
2	0%
3	0%
4	25%
5	75%

It is a good idea to give out certificates.

Disagree 1 2 3 4 5 *Agree*

1	0%
2	0%
3	25%
4	25%
5	50%

It is a good idea to have trophies given out.

Disagree 1 2 3 4 5 *Agree*

1	0%
2	0%
3	50%
4	0%
5	50%

I am missing ... in the conclusion

- Replace trophies with something small students actually care about (candy or whatever), random trophies are overrated and not quite as motivating as we think. Certificates: What is the purpose of them? Only useful if you think this is a good enough exercise to add to for example LinkedIn certificates.
- I think that the lectures should also explain for the examples produced which aspects show quality, and what the criteria of quality are.

Final remarks

What category do you fall under?

- *Student*
- *Teacher*
- *Other*

Student	25%
Teacher	50%
TA	25%

Do you have any experiences with modelling/software design that have shown the importance or influence of models? If so, please explain what that experience was.

No possibility to fill something in.

Do you have any final remarks on the workshop or the thesis?

- I like it a lot, well done!
- No, I wish you good luck with your thesis.

- I couldn't give feedback on the models, but what I missed for the use case diagram was a bubble that interacted with multiple agents (otherwise, the diagram could just as easily have been a list). For the list of requirements: I'd strongly prefer a MoSCoW model over priority numbers (those are rather meaningless, especially when making a quick list).
My experience (the questionnaire doesn't have a text box for it): In software development, proper use of models are a must if you want to get your point across. Modelling makes sure that you cover full work flows. I have worked at 3 IT companies and while none cared about proper UML, it was always appreciated if you could clearly communicate your ideas through models.
- Modelling for formal methods (cs) was my research focus for a number of years. (as the possibility to answer under the question is missing). For the different sorts of models it takes some time to understand, get experience and fluency. I would probably not stick too much to the existing UML versions of models, but take them more as inspiration and allow free-style models. Point is that people start to reflect, identify structure, and make that explicit, and not to struggle with definitions of different categories of models.

References

- [1] S. Brown, 'Software Architecture for Developers', p. 78.
- [2] 'What is UML | Unified Modeling Language'. <https://www.uml.org/what-is-uml.htm> (accessed May 08, 2020).
- [3] J. A. Robinson, *Software Design for Engineers and Scientists*. 2004, p. 414.
- [4] P. Freeman and D. Hart, 'A Science of Design for Software-Intensive Systems', *Commun. ACM*, vol. 47, no. 8, pp. 19–21, Aug. 2004, doi: 10.1145/1012037.1012054.
- [5] A. M. Madni and M. Sievers, 'Model-based systems engineering: Motivation, current status, and research opportunities', *Syst. Eng.*, vol. 21, no. 3, pp. 172–190, 2018, doi: 10.1002/sys.21438.
- [6] C. Piaszczyk, 'Model Based Systems Engineering with Department of Defense Architectural Framework', *Syst. Eng.*, vol. 14, no. 3, pp. 305–326, 2011, doi: 10.1002/sys.20180.
- [7] S. Schulz, J. W. Rozenblit, M. Mrva, and K. Buchenriede, 'Model-based codesign', *Computer*, vol. 31, no. 8, pp. 60–67, Aug. 1998, doi: 10.1109/2.707618.
- [8] S. Ahmed and G. Ashraf, 'Model-based user interface engineering with design patterns', *J. Syst. Softw.*, vol. 80, no. 8, pp. 1408–1422, Aug. 2007, doi: 10.1016/j.jss.2006.10.037.
- [9] T. Menzies, 'Editorial: model-based requirements engineering', *Requir. Eng.*, vol. 8, no. 4, pp. 193–194, Nov. 2003, doi: 10.1007/s00766-002-0156-7.
- [10] F. Patou, M. Dimaki, A. Maier, W. E. Svendsen, and J. Madsen, 'Model-based systems engineering for life-sciences instrumentation development', *Syst. Eng.*, vol. 22, no. 2, pp. 98–113, 2019, doi: 10.1002/sys.21429.
- [11] A. L. Ramos, J. V. Ferreira, and J. Barceló, 'Model-Based Systems Engineering: An Emerging Approach for Modern Systems', *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 1, pp. 101–111, Jan. 2012, doi: 10.1109/TSMCC.2011.2106495.
- [12] B. Cameron and D. M. Adsit, 'Model-Based Systems Engineering Uptake in Engineering Practice', *IEEE Trans. Eng. Manag.*, vol. 67, no. 1, pp. 152–162, Feb. 2020, doi: 10.1109/TEM.2018.2863041.
- [13] A. Mader and W. Eggink, 'A DESIGN PROCESS FOR CREATIVE TECHNOLOGY', p. 6.
- [14] J. M. Bryson, 'What to do when Stakeholders matter: Stakeholder Identification and Analysis Techniques', *Public Manag. Rev.*, vol. 6, no. 1, pp. 21–53, Mar. 2004, doi: 10.1080/14719030410001675722.
- [15] H. Sharp, A. Finkelstein, and G. Galal, 'Stakeholder identification in the requirements engineering process', Feb. 1999, pp. 387–391, doi: 10.1109/DEXA.1999.795198.
- [16] 'Stakeholder Analysis: Winning Support for Your Projects'. http://www.mindtools.com/pages/article/newPPM_07.htm (accessed Jun. 19, 2020).
- [17] 'What is MoSCoW Prioritization? | Overview of the MoSCoW Method'. <https://www.productplan.com/glossary/moscow-prioritization/> (accessed Aug. 08, 2020).
- [18] 'Introduction to MosCoW Prioritization', *Lucidchart*, Nov. 27, 2019. /blog/introduction-to-moscow-prioritization (accessed Aug. 08, 2020).
- [19] 'MoSCoW Method', *Project Smart*. <https://www.projectsmart.co.uk/moscow-method.php> (accessed Aug. 08, 2020).
- [20] J. Lazar, J. H. Feng, and H. Hochheiser, 'Interviews and focus groups', in *Research Methods in Human Computer Interaction*, Elsevier, 2017, pp. 187–228.
- [21] 'Photo, image & design editing software | Adobe Photoshop'. <https://www.adobe.com/products/photoshop.html> (accessed Aug. 15, 2020).
- [22] 'Online Diagram Software & Visual Solution', *Lucidchart*. <https://www.lucidchart.com> (accessed Aug. 15, 2020).
- [23] J. Lazar, J. H. Feng, and H. Hochheiser, 'Surveys', in *Research Methods in Human Computer Interaction*, Elsevier, 2017, pp. 105–133.
- [24] S. Huron, P. Gourlet, U. Hinrichs, T. Hogan, and Y. Jansen, 'Let's Get Physical: Promoting Data Physicalization in Workshop Formats', in *Proceedings of the 2017 Conference on Designing*

Interactive Systems, Edinburgh United Kingdom, Jun. 2017, pp. 1409–1422, doi:
10.1145/3064663.3064798.