

Marcus Joar Hauge

Improved method of creating automatically assessed diagram problems for exams

Master's thesis in Master of Science in Informatics

Supervisor: Guttorm Sindre

June 2022

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



Norwegian University of
Science and Technology

Marcus Joar Hauge

Improved method of creating automatically assessed diagram problems for exams

Master's thesis in Master of Science in Informatics

Supervisor: Guttorm Sindre

June 2022

Norwegian University of Science and Technology

Faculty of Information Technology and Electrical Engineering

Department of Computer Science



Norwegian University of
Science and Technology

Abstract

In the recent years there has been more digitalization of exams for computer science subjects, with the use of e-assessment systems such as Inspira Assessment. Traditionally, exam questions have been manually graded by the teachers and professors, however there are a number of potential benefits of using automatically assessed questions. Several of these question types are in use today, such as multiple-choice, however not all syllabus can be covered.

In order for teachers and professors to create automatically assessed questions that could sufficiently reflect the syllabus, proper tools are necessary. Inspira Assessment as well as similar tools are currently limited when it comes to creating questions for programming and diagrams in particular. Either because of lack of functionality, or simply having poor usability.

The aim of this thesis was to explore how the current tools could be improved in regards to the creation of drag-and-drop type of questions utilizing the Question and Test Interoperability specification (QTI) with diagrams. A design and creating research strategy was used to develop a new tool implementing some of the suggested features thought to improve usability, with the metrics effectiveness, efficiency and satisfaction.

An experiment strategy was used to evaluate the tool against Inspira Assessment. A group of five participants was given equivalent tasks to perform in both tools, while various data points was measured. In addition to the observed data, questionnaires and interviews were used as well. With the data generated from the experiment, the result showed a high overall improvement to the process of creating automatically assessed diagram questions.

Sammendrag

De siste årene har det blitt mer digitalisering av eksamener for informatikkfag, med bruk av systemer for elektronisk vurdering slik som Inspira Assessment. Tradisjonelt har eksamensoppgaver blitt manuelt vurdert av lærere og professorer, men det er en rekke potensielle fordeler ved å bruke automatisk vurderte spørsmål. Flere av slike oppgavestyper er i bruk i dag, som for eksempel flervalgsoppgaver, men ikke alle oppgaver kan dekke pensum like godt.

For at lærere og professorer skal kunne lage automatisk vurderte oppgaver som i tilstrekkelig grad kan reflektere pensum, er det nødvendig med gode verktøy. Inspira Assessment samt lignende verktøy er foreløpig begrenset når det gjelder å lage oppgaver til programmering og diagrammer spesielt. Enten på grunn av mangel på funksjonalitet, eller lav brukbarhet.

Målet med denne oppgaven var å utforske hvordan de nåværende verktøyene kan forbedres i forhold til lagning av dra-og-slipp-oppgaver med bruken av Question and Test Interoperability specification (QTI) med diagrammer. En design- og utviklingsstrategi ble brukt til å utvikle et nytt verktøy som implementerte noen av de foreslåtte funksjonene som var antatt å kunne forbedre brukbarheten, med måling av ulike aspekter.

Et eksperiment ble utført for å evaluere verktøyet mot Inspira Assessment. En gruppe på fem deltakere fikk tilsvarende oppgaver å utføre i begge verktøyene, mens ulike datapunkter ble målt. I tillegg til de observerte dataene ble det også brukt spørreskjemaer og intervjuer. Med dataene generert fra eksperimentet, viste resultatet generelt sett en stor forbedring av prosessen av å lage automatisk vurderte diagramoppgaver.

Preface

This master thesis was carried out in spring of 2022, with a preparatory project conducted in Autumn of 2021, and concludes my two year journey at the Norwegian University of Science and Technology, at the Department of Computer Science (IDI). It is the final submission for the degree of Master of Science in Informatics.

A special thanks to my supervisor professor Guttorm Sindre for all the help and guidance along the way. I would also like to thank all of the people who participated in the experiment and provided me with valuable insight. Finally i would like to thank my family for supporting me throughout the years.

Contents

List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Motivation	1
1.2 Research questions	3
1.3 Outline	4
2 Research method	5
2.1 Strategy	5
2.1.1 Design and creation	6
2.1.2 Experiment	7
2.2 Data generation	10
2.2.1 Documents	10
2.2.2 Observation	11
2.2.3 Questionnaire	17
2.2.4 Interview	18
2.3 Evaluation method	19
2.3.1 Research question 1	19

2.3.2	Research question 2	19
3	Background	23
3.1	Diagram questions in Inspera	23
3.1.1	Context	23
3.1.2	Creating questions in the different genres	23
3.2	Possible improvements	26
3.2.1	Improving the the manual process	26
3.2.2	Automated item generation	27
3.2.3	Optical character recognition	30
3.2.4	Image recognition	32
3.3	Related work	34
3.3.1	Scientific papers	34
3.3.2	QTI editors/tools	36
3.4	Theory	40
3.4.1	QTI	40
3.4.2	Inspira	42
3.4.3	UML	43
3.4.4	User interface	43
4	Result	44
4.1	Design and creation	44
4.1.1	Non-functional requirements	44
4.1.2	Development methodology	45
4.1.3	Technology	46
4.1.4	Initial setup	51

4.1.5	Iteration 1	51
4.1.6	Iteration 2	54
4.1.7	Iteration 3	57
4.1.8	Iteration 4	59
4.2	Experiment	65
4.2.1	Quantitative data	65
4.2.2	Qualitative data	73
5	Discussion	76
5.1	Design and creation	76
5.1.1	Focus	77
5.1.2	Expanding	78
5.2	Experiment	79
5.2.1	Quantitative data	79
5.2.2	Qualitative data	83
6	Conclusion and future work	85
6.1	Conclusion	85
6.2	Future work	86
6.2.1	Development	86
6.2.2	Research	87
Appendix		
A	Requirements and tasks	93
A.1	Non-functional requirements	93
A.2	Functional requirements	94

A.3	Tasks	95
B	System Usability Scale	96
B.1	Questionnaire form	96
B.2	SUS responses	100
B.2.1	Inspira	100
B.2.2	New tool	101
C	Interview transcript	104
C.1	Participant 1	104
C.2	Participant 2	105
C.3	Participant 3	106
C.4	Participant 4	107
C.5	Participant 5	108
D	Repository	110

List of Figures

2.1	Test diagram	13
2.2	Resulting QTI question for task 1	14
2.3	Resulting QTI question for task 2	15
3.1	simple class diagram and corresponding question created in Inspira	24
3.2	Digital circuit schema	28
3.3	Question editor in OpenOLAT with drag-and-drop	37
3.4	Onyx editor graphic match interaction	38
3.5	Hot spot question in Blackboard	39
3.6	QTI example	41
4.1	Single-page application	48
4.2	Example of simple image-to-text with tesseract	49
4.3	UI after first iteration	53
4.4	Figure of communication between client and server	60
4.5	Contour detection step 1: original image	61
4.6	Contour detection step 2	61
4.7	Contour detection step 3	62
4.8	Contour detection step 4	62
4.9	Contour detection step 5: final result	63

4.10	Total time and mean comparison	67
4.11	Total number of actions and mean comparison	68
4.12	Mean values comparison	69
4.13	Mean SUS score comparison	70
4.14	SUS grading	71
B.1	SUS questionnaire q1-q3	97
B.2	SUS questionnaire q4-q7	98
B.3	SUS questionnaire q8-q10	99
B.4	SUS Inspera participant 1	100
B.5	SUS Inspera participant 2	100
B.6	SUS Inspera participant 3	100
B.7	SUS Inspera participant 4	101
B.8	SUS Inspera participant 5	101
B.9	SUS new tool participant 1	101
B.10	SUS new tool participant 2	102
B.11	SUS new tool participant 3	102
B.12	SUS new tool participant 4	102
B.13	SUS new tool participant 5	103

List of Tables

4.1	Iteration 1 requirements	52
4.2	Iteration 1 tasks	52
4.3	iteration 2 requirements	55
4.4	Iteration 2 tasks	55
4.5	Iteration 3 requirements	58
4.6	Iteration 3 tasks	58
4.7	iteration 4 requirements	59
4.8	Iteration 4 tasks	59
4.9	Observation participant 1	65
4.10	Observation participant 2	66
4.11	Observation participant 3	66
4.12	Observation participant 4	67
4.13	Observation participant 5	67
4.14	SUS scores	70
4.15	Wilcoxon Signed Rank Test for time	72
4.16	Positive categories Inspera	73
4.17	Negative categories Inspera	74
4.18	Positive categories new tool	74

4.19 Negative categories new tool	74
---	----

Chapter 1

Introduction

1.1 Motivation

Technology enhanced learning is becoming increasingly popular in education [6], and a wide variety of tools are being used. Learning Management Systems (LMS) such as Blackboard and Canvas, and e-exam/e-assessment systems such as Inpera Assessment are much used. These platforms (or more standalone solutions like Piazza) have allowed the introduction of online video lectures, dashboards, forums and collaboration tools, online assignments and assessments etc. With this increase of digitalization of education, classes may handle a higher number of students allowing more to enroll. This being especially true for schools and universities that are completely virtual.

With a high number of students compared to teachers and professors, it may be difficult to conduct tests and exercises and have to grade all of them, as it could be too time consuming. A solution to this is for example peer-review where students grade each others work. This however cannot be done with the more important tests and exams. Another solution is to have automatically assessed tests. Much research has been done on this such as [4] and [15].

Test questions that can be automatically assessed and graded provides a number of benefits. First and foremost it can reduce the time it takes to grade a test substantially, which is especially useful for the case mentioned of having a large number of students in a single class, or a low number of professors compared to students. Secondly, automatic assessment provides a more reliable way grading the tests, as the correct answers are programmed for the question whereas humans are prone to make errors. Teaching staff could be freed from control-oriented tasks like if the student has delivered and how much they

scored, and instead focus more on helping the student directly.

Another benefit is in the use of formative [43] tests, i.e tests conducted for the purpose of practising the subject. Normally with such tests, it may take some time after submitting the test answers before it is graded by the teacher and feedback is received. Automated assessment instead provides instant feedback and depending on the complexity of the system and/or question design, can provide not only correct/incorrect feedback but also more details of why it was incorrect and then point to what the student could do differently or relevant resources such as videos or articles so that the student can learn and improve [13].

There are a number of different question types that are or can be automatically assessed. Some examples are *multiple choice*, *text or numeric entry*, *math entry*, *true/false*, and *drag-and-drop*. In order for automatic assessment to be viable, the tools to create such questions need to be sufficiently designed and cover a range of different question types that are often used in tests. Regarding digital tests in computer science and programming, the questions have usually consisted of the manually graded type, like for example text answers (code or not), either because the digital tools does not offer the proper features or because they are limited or cumbersome to use for creating programming questions. There are some automatically assessed question types that are used such as multiple-choice, however these are not always appropriate for the syllabus. They are useful for testing knowledge of a subject, but not so much for testing creativity and constructive skills [38]. There are currently many platforms and tools offering digital assessments, tests and exams, with their own test question editor integrated. As discussed in section 3.3.2, they are in many ways similar. There has been done work on analysing such tools, with focus on Inspira [19] as that is the platform used by NTNU for digital exams, and then discussing and developing improved tools such as [Parson master] and [parson master 2]. These focused on developing tools that are better for creating parsons problems, i.e a question consisting of a number of code snippets the student has to choose from to form a correct answer, and the results were positive.

In addition to having code and programming as syllabus, many subjects in computer science and informatics education utilize and teaches diagrams for the software development process and modeling purposes. For example database courses might use ER-diagrams and relational algebra diagrams, Information System analysis courses might use business process diagrams like BPMN etc. Other branches of engineering also utilizes various diagrams, electrical engineering have analog and digital circuits, chemistry have diagrams for molecular bindings and so on. In exams, questions for testing knowledge of the vari-

ous diagrams have often required the student to draw the diagram by hand with pen and paper, or sometimes use a digital drawing tool to create them. The disadvantages of hand drawn diagrams are that it can be hard to read the handwriting, are manually assessed, and requires an extra step of taking a photo/scanning it during the exam. Using a digital tool is better, but is often not accessible due to cheating-prevention on the computer during the exam. It could therefore be desired to have exam questions with diagrams contained inside the digital exam tool and not require the use of interaction outside of that. It is possible to create diagram tasks that can be automatically assessed in Inspira, but this process is rather tedious [16].

This thesis aims to explore the current process of creating exam questions specifically about UML diagrams in Inspira and find out if there are possibilities for automating parts of it, and overall making it less time-consuming and cumbersome to use, in the form of a new developed tool. A preparatory project was conducted in autumn of 2021 that consisted of examining the problem and conducting some of the background work for this thesis, such as finding existing work on the subject and decisions regarding development method and technology choices. The results of the project has been improved and expanded upon during the thesis as more information has arisen and the scope slightly changed.

1.2 Research questions

As the aim of the thesis was to identify potential improvements for Inspira's question authoring tool in regards to diagram question creation and evaluate them, it was natural to pose the following research questions:

- RQ1: How can an IT tool be designed to be more efficient in creating automatically assessed diagram questions for Inspira?
- RQ2: To what extent can the IT tool in RQ1 be used as a replacement to Inspira's authoring tool?

1.3 Outline

The thesis report is structured as follows. Chapter 2 describes the chosen research strategy, the various data generation methods used and the evaluation in regards to the research questions. Chapter 3 describes background work containing more context, related work and theory. Chapter 4 describes the results for each of the research strategies. Chapter 5 discusses the results and their significance. A conclusion along with potential for future work is described in chapter 6.

Chapter 2

Research method

2.1 Strategy

This thesis utilized the Design and Creation research strategy [29, p.108], as it was deemed the most appropriate choice in that an IT artefact was to be the main contribution in the project, and fits the research questions. In addition, the experiment strategy was also used for the evaluation of the IT artefact produced, and is described below. Oates [29] also describes the strategies survey, experiment, Case study, action research and ethnography relevant for information systems and computing research. These other strategies could potentially have been used as well.

A survey could have been used to gather information from testing the prototype, however it is more appropriate for a larger sample size, where a sample of at least 30 participants is recommended [29]. Considering the time frame of the project and the extensive process of the data gathering of each participant in this case, it would not be optimal. If a survey was to be used, the testing of the new IT artefact would have to be simplified to allow a greater number of participants. There is also the problem of the sample frame being too small, as ideally the participants would be professors/teachers at NTNU with experience in using Inspira's tool and the relevant question types and topics in order to receive better quality feedback.

A case study looks at many details, complexities and relationships in a situation in order to gain a very detailed understanding of it, rather than looking at fewer isolated factors with more measurable numerical data to determine the effect of changed/introduced factors. Another point is that a case study looks at the particular situation in its natural setting, in contrast to a "lab" setting where the situation is controlled in order to better isolate and

observe the specific factors.

Action research in the context of IS research is more focused on the improvement of processes such as a development methodology, rather than in this case the specifics of a software product, and was therefore not considered.

Ethnography is the study of the culture or norm of a specific group of people, for example a workplace, in its natural setting. In this context, it could be used to observe and experience how professors would use the new tool. However, this would most likely not provide very valuable insight as the tool developed in this thesis is not much different in terms of use compared to Inspira in that they serve the same task, but where the new tool is a potential improvement meant to be more efficient and usable.

2.1.1 Design and creation

The Design and Creation strategy focuses on the development of IT artefacts. A software application is the main contribution of this thesis, in addition to the increased understanding of possibilities and limitations when designing such tools. Not only from developing and evaluating the tool, but also knowledge gained from reviewing literature and analysis of existing work and tools related to the topic.

This thesis follows a five step iterative process described in [29, p.111]; awareness, suggestion, development, evaluation and conclusion. The awareness step consists of analysing Inspira and similar software solutions, and conducting a literature review in order to better understand the problem and discover potential solutions. The suggestion step involves describing ways to address the problem. The development step involves developing a working prototype containing some or all of the suggestions from the previous step, and may also lead to more recognition of the problem. The evaluation step involves testing the prototype with real users, gaining data for comparison to determine its efficiency and usability. The conclusion step discusses the results of the evaluation and outlines the contribution of this thesis and the potential for further research. Of the types of artefacts mentioned in the book, the artefact developed here falls under *instantiations*; "*a working system that demonstrates that constructs, models, methods, ideas, genres or theories can be implemented in a computer-based system* [29, p.108]."

2.1.2 Experiment

In order to answer RQ2 (and partly RQ1), the IT artefact needed to be measured and evaluated. Therefore in addition to design and creation, the experiment strategy was also used. Oates [29] defines the experiment strategy as looking at cause and effect relationships and whether or not a particular factor or factors are connected to the observed outcome of an event.

An experiment requires one or multiple hypotheses to be stated, and then tested empirically. Since the purpose of the use of experiment was in evaluating the new IT tool developed, and RQ2 being "To what extent can the IT tool in RQ1 be used as a replacement to Inspera's authoring tool?", a natural choice for a hypothesis was derived as follows:

- "The new IT tool is more usable in creating drag-and-drop QTI questions of diagrams than Inspera."

The experiment was then designed around to confirm whether it holds true nor not. It is important to isolate the factor that is studied to affect the outcome, and remove or minimize other factors which also may affect the outcome. This can potentially prove to be challenging as not all affecting factors are obvious, and some can go unnoticed. It is therefore important to make the experiment repeatable and repeat it a reasonable amount of times in order to obtain conclusive results that can be generalised.

Oates [29, p.127] characterises an experiment, among other descriptions, as "*A process of (1) observation or measurement of a factor; (2) manipulation of circumstances; and (3) re-observation or re-measurement of the factor to identify any changes*". In this case, the factor that was observed/measured was the creation of a QTI question derived from specific instructions or descriptions containing both textual and image content. Manipulation of circumstances here refers to the change of which QTI tool was used to create the question, i.e changing from Inspera Assessment to the new tool developed in this thesis. The re-observation then consists of measuring the same process described in the first step, with the other tool.

Oates [29, p.129] defines the terms dependent and independent variables in an experiment. The dependent variable(s) is something that exists in the situation at hand, and is what the results are drawn from. The independent variable(s) is that to be changed in the situation, and affects (or does not affect) the dependent variable(s).

It is important to control as many variables as possible in order to be sure that only the

independent variables are changed while the others remain the same. Some of the steps taken to ensure this were:

- Having each participant also serve as a control group (*within-subject* design), i.e they will complete their task in both tools instead of a different group for each tool, in order to eliminate the problem of different people performing at different levels.
- Making sure the participants have experience with use of both Inspira and the new one tool.
- Keeping technical factors constant, such as having the participant use the same computer and the same web browser during the experiment.
- Having the participant do a trial attempt of the given task beforehand to make sure they understand the task and the workflow within the tools.

Within-subjects and *between-subjects* are designs that differentiate between having the whole group of participants exposed to all the conditions, or separate groups exposed to one condition each [37]. There are a number of arguments for choosing either of them. Using between-subjects, the amount of time required of each participant is less, which could lead to people being more willing to participate if asked. It would however require more participants than within-subjects as the data per person is less. Also, since it would be a comparison of two groups with different people, there could be more variables at play such as how skilled they are in computer usage (i.e speed) or how fast they learn new things that may affect performance.

With the *within-subjects* design, those variables do not have the same influence, as the participant is kept constant for the measurements in both conditions. A downside of this could be a crossover effect, in that a participant may learn between the two tasks and perform better on the last one because of it. Or, especially if the experiment is done in a single session, the participant may get tired after the first task and performs worse on the second task.

Weighing the pros and cons of both designs, using only one group seemed like the better choice. The higher number of participants needed for two groups could prove difficult to acquire for such an extensive experiment, at least with the means available to the researcher. The crossover effect could be somewhat mitigated. The task instructions for the two tools described later in section 2.2.2 were different, so there was not much useful information to be learned between the tasks that could noticeably impact the data.

Secondly, the task involving the use of the new tool was shorter and estimated to be significantly easier to perform. A Latin Squares design could be used here, but was not because Inspira would presumably take more time for the participants to learn, more time to complete the tasks, and cause more frustration with the participant. Since Inspira reflects the general standard of similar question authoring tools, it seemed more reasonable to let that be the frame of reference for the participants. If they would start with the new tool and it was in fact a considerable improvement to Inspira, the participants could potentially have a very negative experience when then moving to Inspira, which could result in unreasonable data.

Deciding what data to look at

As the hypothesis of the experiment states that the new IT tool is more usable than Inspira, it is important to define what *usable* means. ISO 9241-11 [1] defines usability as “*the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*”. The three metrics stated are then further defined as follows:

- Effectiveness: *"accuracy and completeness with which users achieve specified goals"* [1].
- Efficiency: *"resources used in relation to the results achieved"* [1].
- Satisfaction: *"extent to which the user's physical, cognitive and emotional responses that result from the use of a system, product or service meet the user's needs and expectations"* [1].

Details of the exact data measured are described below in the data generation methods. Observations and measurements was conducted in a before and after manner(pre-test and post-test). The independent variable that was changed was defined as the tool used for completing a task.

A note about the new tool is that it did not attempt to be as extensive as Inspira containing a number of different question types available for a variety of use cases. Instead it merely focused on a subset of Inspira's features, and therefore only the use of these specific features in use were measured and studied.

2.2 Data generation

This thesis utilized several methods of generating data, both qualitative and quantitative. In regards to the first research question, development of the IT tool, mainly qualitative data was looked at. Qualitative data means data that is non-numeric, as in textual content, images, video, audio etc. For the second research question, or the experiment part of the thesis, a combination of both data types was used. Quantitative data consists numeric data or at least data based on numbers. The specific methods of data generation used are described below.

2.2.1 Documents

Documents was the chosen method for research question 1. This was chosen because the information sought after here is very much available on the internet already which makes it easy and quick to obtain. The documents in this context consists mostly of the type *found documents* and not *researcher-generated documents*. *Documents* can comprise of a variety of different data, and is not limited to only academic articles. Any written material can be seen as a form of document. Some examples are company documents such as employee data and bills, personal written information such as notes, and publications such as any kind of literature [29, p.234]. In addition, *documents* also encompasses more than just textual content. Multimedia documents [29, p.235] can be image, video or audio data. Complete websites can also be seen as documents, which is particularly relevant in this context as Inspera's QTI question editor is a web application. The following list of items was the main targets of the documents data generation method:

- The QTI standard documentation and implementation details [9] [8].
- Inspera's QTI question editor and the associated documentation and/or user manuals [19] [20].
- Exported QTI/question files from Inspera.
- Comparable software solutions/tools to Inspera and associated documents.
- Published articles or work related to Inspera, QTI or that are otherwise relevant.

2.2.2 Observation

The observation method was one of the chosen methods of generating quantitative data, and was used in context of research question 2. To *observe* can have a number of meanings, such as looking at something or listening [29, p.202]. In this context it was the act of watching other people performing specific tasks, while recording and taking notes of certain details of interest.

The approach of an observation can have a big impact on the results. Oates [29, p.203] mentions *covert* and *overt* approaches, where the difference is whether or not the ones being observed is aware of it. There are pros and cons of either approach. With a covert approach (participants not aware of observation), it is more likely that the participants would act naturally and be more relaxed. Knowing that they are being observed may cause them to feel pressure, become unsure of themselves and make more mistakes than in a natural setting. However, performing the observation covertly also poses some potential problems. The researcher would not be able to guide the participant to perform their task as intended in case of misunderstandings or other problems that may arise during the process. It would also mean that the researcher could not directly explain to the participant how to properly use the new IT tool and guide them in a trial run in preparation for the actual experiment, as this would hint that they are going to be observed.

In an overt observation on the other hand, the situation would be much more controlled. The participant would be able to ask for help if there is something they do not understand or for minor clarifications. Even though as mentioned the behaviour of the participants might be affected by the observation process, it would most likely still be more consistent than if done covertly. This is because each participant is observed performing the task in both Inspira and the new IT tool, so that even if they for example feel nervous about being observed, that nervousness would affect both performances. In contrast, for a natural setting with no observer present, the participant might do things like eating or become otherwise distracted by something during the process which could impact the results. Ethics is another point to be made, as in an overt observation the participants are asked beforehand if they want to take part of the experiment or not, while this is not the case of covert.

Considering the arguments mentioned, as well as the fact that it would be quite challenging to design the experiment with a covert approach while still obtaining all the desired data, an overt approach seemed optimal. Not to mention that covert observation would seem weird in this case, where one would need to recruit participants to perform a very

specific task. Then, pretending to leave the room but really observing them through a one-way mirror or hidden camera would potentially be unethical and illegal.

There is also two other contrasts in the approach of observation; *systematic* and *participant* observation. In the latter, the researcher put themselves in the situation to be studied in one way or another to gain the perspective of the people in that situation. Several types of participation exist such as *complete observer* and *participant observer*, however further details of this approach are not discussed here as the systematic observation approach was chosen instead. This is because systematic observation consists of the use of pre-planned events to be observed and specific design of the process and task provided to the participants.

To best answer the hypothesis of the experiment, the following data was selected to be observed/measured in the observation method:

- Time spent to complete a task.
- Number of performed actions/clicks.
- Number of times asked for help.
- Number of errors.

These metrics were chosen to measure the effectiveness and efficiency of the two tools. Effectiveness consisted of number of actions, errors and times asked for help. Efficiency consisted of time spent completing the task.

A task was given to the participant and was observed being performed. Because of the UI and workflow differences between Inspera and the new IT tool, a separate task description was made to make things easier and less confusing for the participants. To be able to evaluate more than just one specific component of the new tool, two different tasks were given to the participant, with some variations of steps. This was also done in order to make it more fair for both tools when comparing them. Two tasks allows the test to include both of Insperas drag-and-drop gap element types, one where the elements are text based and one where they are images.

Even with two different tasks, there were still features that went untested, but adding additional tasks to the experiment could potentially be detrimental, as the time it would take to complete the tasks would be higher and people would more likely choose to not participate because of it, or become increasingly frustrated or tired during the test.

Some steps were done in preparation of the test runs in order to make the process in both tools as similar as possible. When creating a new question in Inspera, one has to navigate the menu, click "Create new", and select the desired question type before creating the actual question. These steps were excluded from the test and done ahead of time as the new tool did not have them. Various other optional operations and settings were also excluded such as setting a name for the question, creating question description, setting scoring options etc.

The diagram image used for the tasks was a class diagram consisting of four classes containing class name, attributes and functions with data types, as well as association and inheritance arrows between them. The domain the diagram represents was simple so that the participants could easily understand it, although an understanding of the contents of the diagram was not necessary to complete the task and should not affect the results. The diagram is shown below:

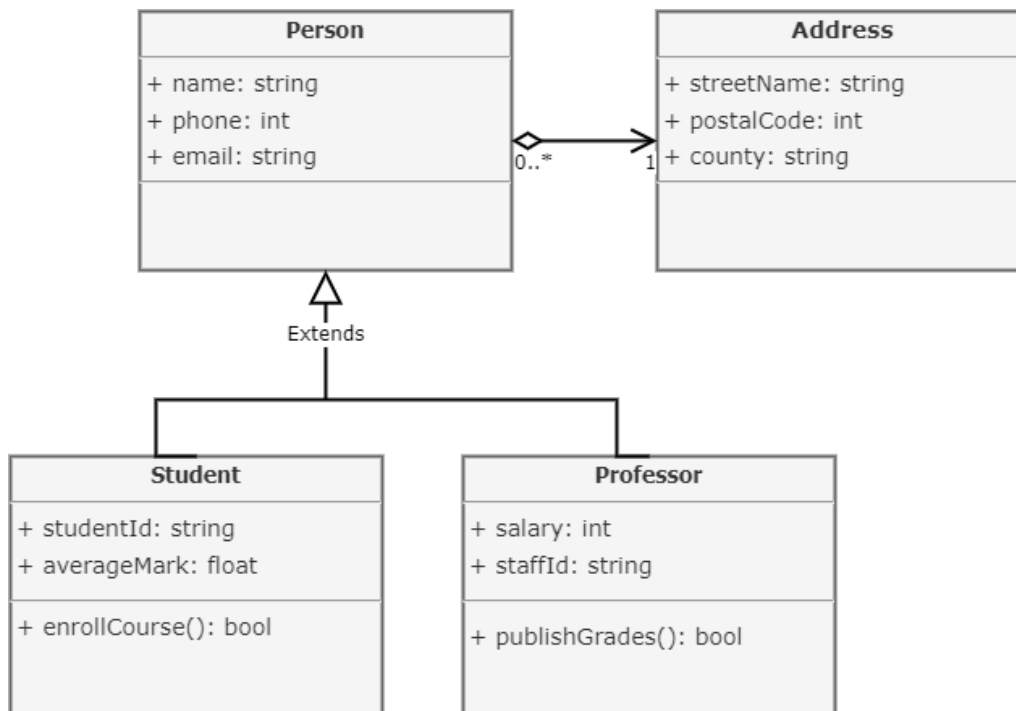


Figure 2.1: Test diagram

The resulting questions created with the diagram would then be the following:

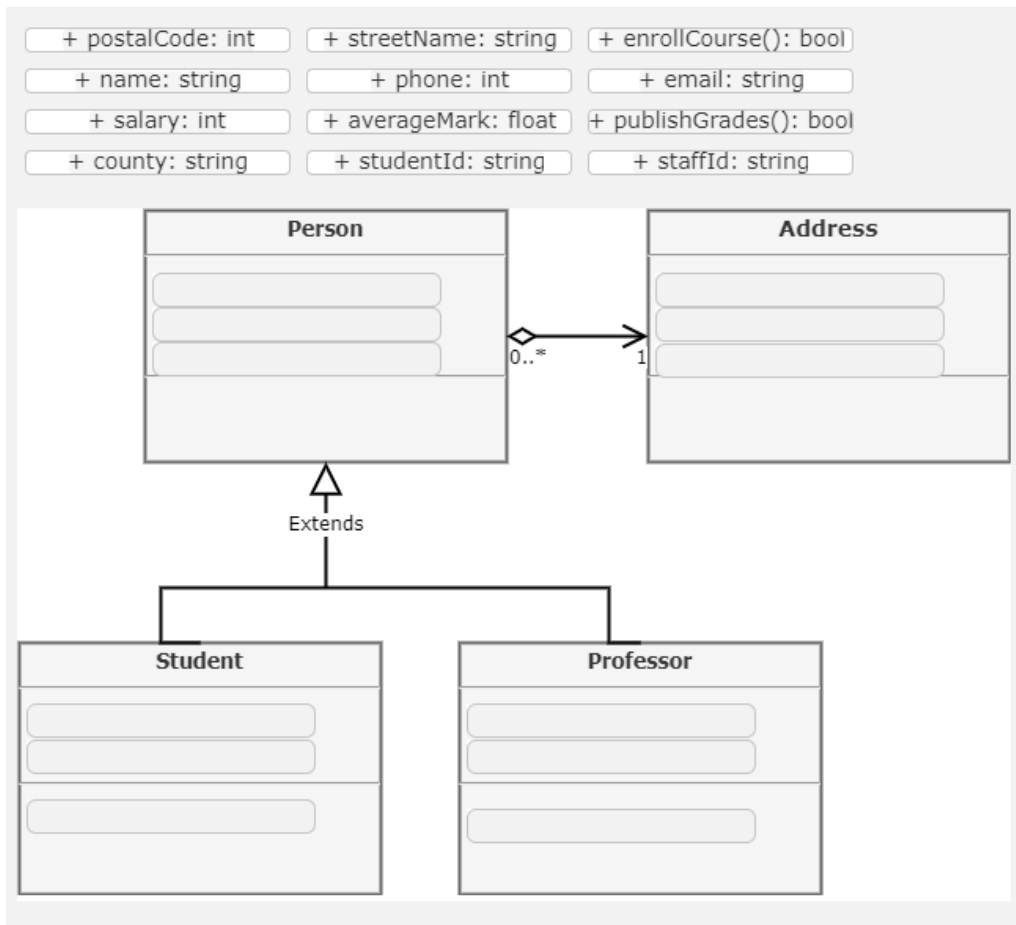


Figure 2.2: Resulting QTI question for task 1

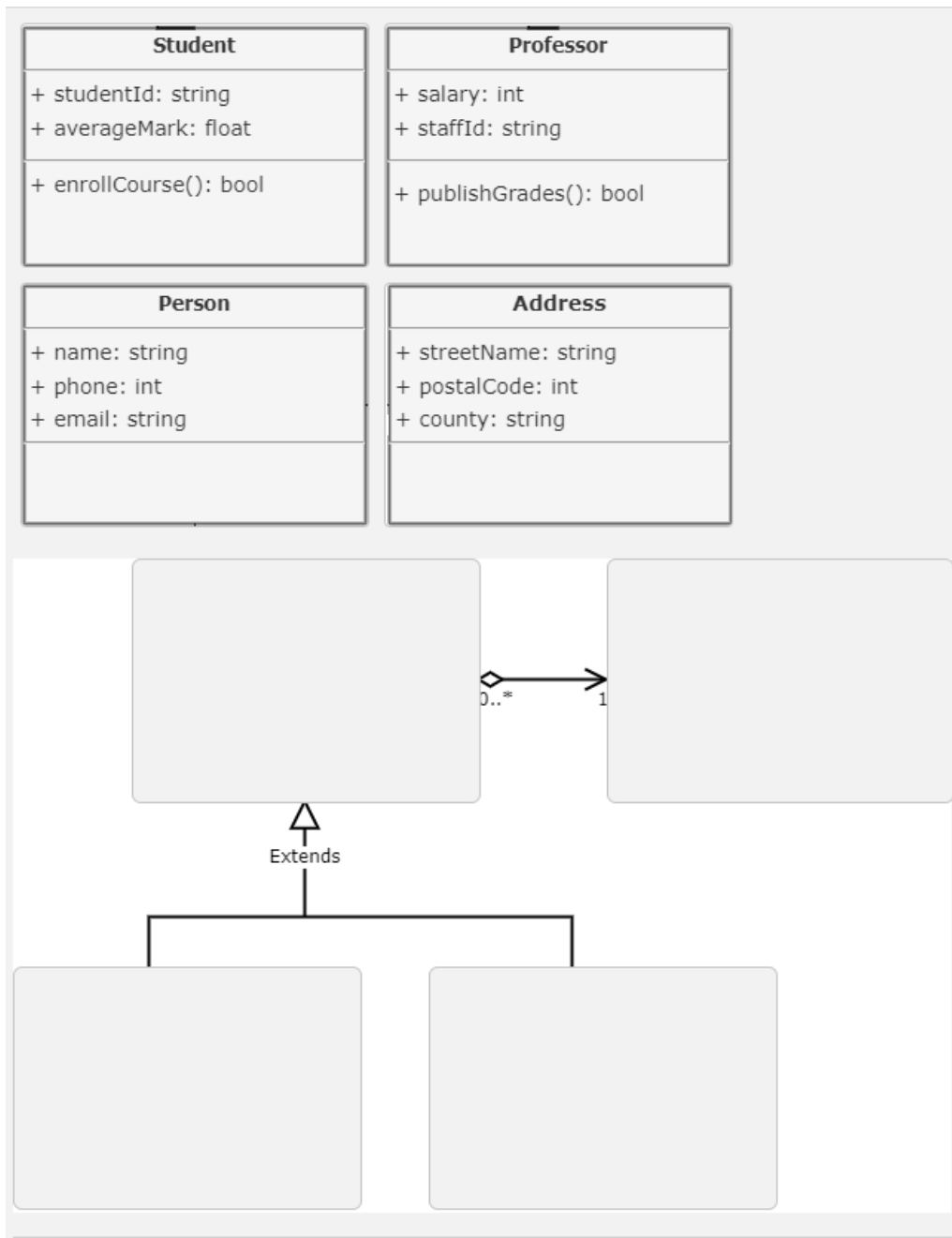


Figure 2.3: Resulting QTI question for task 2

The task descriptions are as follows:

Inspera task description

Task 1: word elements

1. Click on the blank background.
2. Click "Upload image" and upload specified image file.

-
3. Click on the background/content area, under Setup, set "Gap size" to "Same size".
To change the size of all gap elements, change the size of one of them.
 4. For each class member in the diagram:
 - (a) Create a new GAP element.
 - (b) Set the correct answer for it.
 5. Set the correct GAP sizes and position them to the correct location.
 6. Save the question.

Task 2: image elements

1. Click on the background/content area, then under Setup, set "Token type" to "Images".
2. For each class in the diagram, crop it to a separate image.
3. Click on the blank background.
4. Click "Upload image" and upload specified image file.
5. For each class in the diagram:
 - (a) Create a new GAP and place it in position.
 - (b) Click on the GAP and click "Add correct answer".
 - (c) Upload an image of a class.
 - (d) Set the Width of the correct answer so that it fits.
6. Save the question

New tool task description

Task 1: word elements

1. Upload specified image file.
2. Turn each class member into drag-and-drop with automatic detection(Detect class members inside button).
3. Export the QTI question

-
4. Import the zip file in Inspira, with the "Create new" option when prompted.

Task 2: image elements

1. Upload specified image file.
2. Turn each class box into drag-and-drop with automatic detection (button: Make classes into drag-and-drop).
3. Export the QTI question
4. Import the zip file in Inspira, with the "Create new" option when prompted.

2.2.3 Questionnaire

A questionnaire was another chosen method of collecting data. It consisted of a specific set of questions presented in a specific order. It was chosen in addition to the observation method as a way to obtain standardized data to be better able to answer RQ2 and the hypothesis, with more focus on the *satisfaction* part of usability. The questionnaire was *self-administered*. There already existed a questionnaire suitable for this context called the *System Usability Scale* [22]. SUS is an industry standard in obtaining assessment information about systems, and provided a nice way of comparing Inspira and the new IT tool. The questionnaire consists of the 10 following questions in order:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.

10. I needed to learn a lot of things before I could get going with this system.

Each question is answered by ranking the statement with a number between 1 and 5. There is a specific way of calculating the resulting score. For the questions with an odd number, 1 is subtracted from the score. For the ones with an even number, the value is subtracted from 5. All the values are then added together and multiplied by 2.5. This results in a normalized total score from 0 to 100. The average SUS score for a system is 68 [22], and can be seen as an OK system if it has around or above that number. However more importantly is how the score of Inespera and the new tool compares to each other.

2.2.4 Interview

Interview was the last method of collecting data for the experiment strategy. It was chosen as a way of getting some extra feedback or comments about the new IT tool as qualitative data. This could be used to uncover deficiencies, improvement potential, desired features for future development etc which would not be possible (or difficult) to extract from the questionnaire.

One normally defines three types of interviews [29, p.187]; *structured*, *semi-structured*, and *unstructured*. These types differ in how strict the questions are laid out and how the answers are expected. In this context any of the types could probably work fine, however since the data sought after here was more loosely defined, a semi-structured interview seemed like the best option. This leaves the interviewer in some control of the questions and conversation, while still allowing the interviewee to talk somewhat freely and potentially provide ideas or insight not thought of by the researcher.

The participants were given the topic/questions of the interview in advance before the experiment in order to make them more prepared and have the questions in mind when they use the new IT tool, potentially producing better answers afterwards. The interview and the experiment as a whole took place in the participant's office or otherwise preferred location in order for them to feel the most comfortable. Audio recording were performed as a way to capture the interview data. This saves time for both the interviewer and interviewee as well as making sure all the information are captured, compared to taking notes or relying on memory only.

The interview consisted of the following questions:

1. What did you like about the new system?

-
2. What did you dislike about the new system?
 3. What other features would you want in the new system?
 4. Are some aspects better in Inpera than the new system? what?
 5. Do you have any other suggestion or comment about the new system?
 6. How viable do you see the use of drag-and-drop diagram questions for either formative or summative tests?

2.3 Evaluation method

2.3.1 Research question 1

Regarding RQ1 and the Design and Creation strategy, minor evaluations were done in iterative steps during the development process. Each iteration ended with a small analysis of the produced result, what could be improved and what new features should be prioritized. However the main evaluation of the new system is linked to RQ2 and the experiment strategy.

2.3.2 Research question 2

Effectiveness, efficiency, and satisfaction were measured in different ways. Effectiveness was measured by number of performed actions, errors and number of times asked for help. Efficiency consisted of a measurement of time to complete a task. Satisfaction was measured with a combination of data from the questionnaire and the interview.

Quantitative data

The quantitative data consisted of the measured values from the observation as well as the SUS questionnaire. The data can be categorized by four types; nominal, ordinal, interval and ratio data [29, p.247].

SUS questionnaire:

The data from the questionnaire goes under the ordinal type, as it uses a Likert scale where questions are answered with 1-5 of how much they agree with the statement. As mentioned, a specific method is used on the data to calculate the final score. Something to note here is that the final score might not have any significant meaning if the score of the two systems are relatively close, e.g two scores of 60 and 70. Also, the score for one system by itself might not say much about its usability, but the comparison of two or more systems could potentially tell if one of them is generally better than the other.

Time to complete task:

This consisted of the time it took the user to complete a task, where completion is having performed the last step specified in the task description. The time goes under ratio data as the scale used has a true zero [29, p.248], i.e zero being a possible value. The measurement of time was made to be as comparable as possible between the two systems, therefore as mentioned some steps in the question creation process were altered or excluded such as navigating menus. The completion time of both systems was then compared to see if there was any significant difference.

Number of actions performed:

This consisted of how many actions the user performed to complete a task. An action would be any type of interaction with the system, such as clicking of buttons, selecting elements, keystrokes with the keyboard etc. When using the keyboard, the press of a single character would not necessarily be defined as an action by itself, but rather the whole word/sentence typed. The number of actions data goes under ratio data because it also counts from true zero.

Number of times asked for help:

This consisted of how many times the user asks for help during the task. For the same reason as above, this is also ratio data. The definition of asking for help was important to be specified. The participants may think out loud during the test and say comments in a question-like manner. These are not counted. Both questions regarding how to perform their task, as well as questions regarding how to operate the tool correctly were counted.

The observer were more reluctant to assist with the latter, as the difficulty of using the tools goes under usability and are a part of the test, however help would be available if they got stuck. Though this would be unlikely to happen much as all participants were given guidance and practise runs prior to the actual test.

Number of bugs/errors

This consisted of how many times errors were present in the resulting QTI question from the performed task, as well as errors/mistakes made by the user during the task. As this also were a count from true zero, it went under ratio data. It was important to establish the definition of an error. The following cases were counted as errors:

- Error caused by the user where the system behaves differently than they expected, either occurring during the test or with the resulting QTI question.
- Error caused by the system, i.e the user interacts with the system correctly, but the system behaves unexpectedly. This could for example be crashes.

Each error that occurred was either counted as a user-error or system-error to get a broader view of comparison. User errors were further categorized into the following three categories:

- Accidentally performed action
- Failed action
- Unnecessary action

Different errors made could be specified as a failed action for example, however multiple counts of the same error were not counted further. For example if the participant performed unnecessary re-sizing of GAP elements many times it would count as one error, and if they also performed unnecessary moving of GAP elements, that would count as one additional error.

Qualitative data

The qualitative data generated regarding RQ2 consisted of video recordings performed during the observation, and audio recording from the interviews. The recordings of the

observations were reviewed and transcribed into numerical data and used for the analysis of quantitative data. The recordings of the interviews were transcribed into text content.

As the interview was of the type semi-structured, the answers received might vary quite a bit and have different themes to them. It was therefore important to prepare the data to be better able to analyse and draw conclusions from it. The first step of organizing the data was to sort it into either relevant or not relevant to the research question. With all the relevant data information established, it was then further processed by categorising each segment into fitting themes using a *deductive approach* [29, p.269], i.e having the themes be pre-determined to some degree prior to the categorization process. Although, if there appeared to be other relevant themes in the data not previously thought of, they could be used as well.

Chapter 3

Background

Some of the work done in section 3.1, 3.3 and 3.4 was part of the preparatory project [17], and has been revised and improved throughout the thesis.

3.1 Diagram questions in Inspera

3.1.1 Context

There are three question genres in Inspera which are suitable for creating diagram questions: “graphic gap match”, “graphic text entry”, and “drag-and-drop”. These are described in more detail in the section below as well as in the theory section. In the next section, each question genre will be examined, where creation of some questions will be analyzed in order to see what steps could be automated, and potential general improvements. The questions, or rather diagrams, will be a simple "fill in the blanks" type where parts of a UML diagram are missing.

3.1.2 Creating questions in the different genres

Graphic gap match

The first question created in Inspera was a class diagram with the data types for the variables removed and available as alternatives to be dragged into the missing areas of the diagram. This was in the "Graphic gap match" genre, with the elements to be placed were in text form. Below is a figure of the solution diagram created beforehand, and the final

question from Inspera.

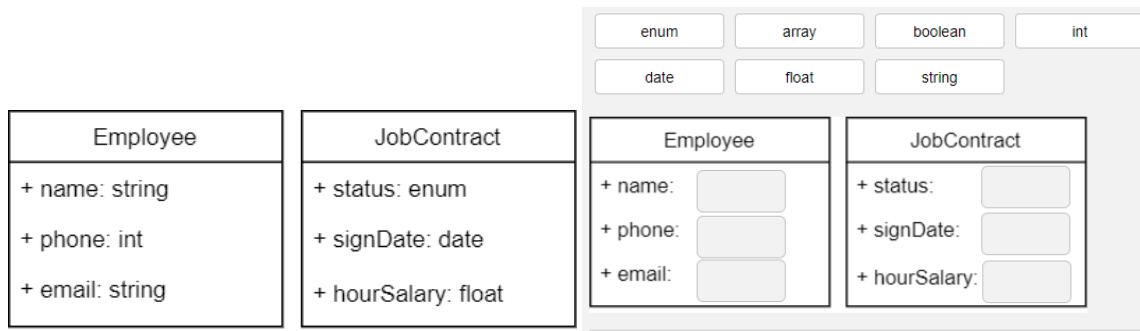


Figure 3.1: simple class diagram and corresponding question created in Inspera

Steps involved here are to first create the complete solution diagram, and then upload it to Inspera as the background image. In no specific order, one then have to set the background diagram size, create answer alternatives one by one, their gap size and position the gaps. Furthermore one has to set the correct answer for each field, and optionally add distractors. Problems with this is setting the fields positions manually by eye, as there is no option to snap to grid. Alternatively it is possible to set the coordinates for each field. There are also no option for a field to accept multiple alternatives for correct answer, for example in this case a class variable might have multiple data types deemed appropriate for it.

Another question in this genre was also created, where the drag elements to be placed were in image form instead of text. The same diagram as above was also used here. Similar as above, one has to create the solution diagram and upload it as the background image, as well as setting its size. One also need to set "token type" (type of gap element) to image instead of text. For creating the gap elements, one first has to insert a new gap and then upload the image for it, in addition to positioning it and setting the correct size. A problem with this, is that one cannot upload the whole diagram and select only a snippet of it to be used for a gap. Instead, one has to crop the diagram into multiple images outside of Inspera, to be used for each gap. This can be quite a tedious and time-consuming process if there are many elements. Once an image is uploaded and set to a gap, it is given a default pixel width, and not the actual width of the image. For the gap to be scaled and appear correctly, the user is required to either adjust the width several times until it looks acceptable, and not blurry. Alternatively, they could look up the image file properties of the specific gap image on their computer, and use that value in the gap width property in Inspera.

Graphic text entry

Graphic text entry is similar to graphic gap match. Here, empty text fields are placed on the background image, then the correct answer is set for each field. Here, it is possible to set multiple correct answers for one field. This question type and the process for creating it in Inpera is almost identical to graphic gap match, with the main difference of not having the option to set "token type"(either text or image drag elements). One potential big problem, is that the the text field gap size can not be changed. Because of this, the user is required to design their diagram to be properly scaled with the default gap size here.

Drag and drop

This question type is also similar to the other two in many ways, with some differences. The creation of drag elements and drop areas are a bit different, as they are created and changed separately instead of only interacting with the drop area (gap) in the others. One drag element can be mapped to multiple drop areas. The elements can be either text, images or both. This one actually has snap to grid functionality which makes it somewhat easier to make it look the desired way. It also has an option for the drop areas to be highlighted or not. However they can only be transparent, as in the area on the background image you are trying to hide and create a drag-and-drop element out of is always visible. Therefore, the user would have to edit the background diagram and remove whatever content at issue, before uploading it to Inpera. Another problem that may cause this question type to not be usable for diagrams, is that the drag elements can only be shown on top of the background image, and not at the top (or elsewhere outside it) like graphic gap match. Depending on how many drag elements are created and how big they are, they could then block out large portions of the background image.

General problems

There are also some general problems applying to all of the question types. The UI is problematic at times. For example, the menu on the side changes entirely depending on what the user clicks on. One specific menu appears when clicking on a gap element, hiding the other menu options. Two other menus can appear depending on where on the white space inside the question the user clicks. This could be very frustrating especially for new users, as they would seemingly have to guess and click around in order for the

correct menu to appear. Another problem that occurred, and recreated several times, was when uploading an image. After uploading, there is a progress bar and an "insert" button for the user to click. The progress bar often waits a little bit after reaching 100%, but if the user clicks "insert" too quickly, even if the progress is at 100, the process seemingly fails. The user would have to repeat the steps of clicking on background image in the menu, click upload, select the image, and then insert.

3.2 Possible improvements

Many improvements to the current question creation process could be thought of, but with highly varying level of complexity. Based on the current documentation of Inspira and the similar tools, as well as relevant articles, suggestions for further development is discussed here. However, the resulting tool in this thesis could not pursue all of these because of the limited time frame for development.

3.2.1 Improving the the manual process

This is the least complex approach of the mentioned, and initial idea pursued in the preparatory project and the first iteration of the new tool. Assuming the question creation process starts out with a finished solution diagram created in a third party drawing tool, how can the current steps in Inspira of turning the diagram into a drag-and-drop question be more efficient? An analysis of the current process was described in section 3.1.2.

The steps for creating a drag-and-drop diagram question in Inspira would generally be something like this: 1. Create the solution diagram. 2. Remove elements from the diagram (depending on question type) 3. Crop the diagram image into the smaller desired images for each gap element. 4. Create the gap elements in Inspira. 5. Position the gap elements in Inspira. 6. Set correct answer for each gap element in Inspira. (either write text or upload image) Particularly with the image form of gap elements, a lot of time can go into cropping each element into individual images before uploading to Inspira. A possible improvement to this is to allow cropping of the elements from the diagram directly in the application and automatically create "drop" elements from them. While cropping the elements, the position and size of the gap can be set at the same time, which would be the same position where the cropped area is. Snap to grid functionality can also be useful to save time, which was lacking in the "graphic gap match" type. An option for an answer field to have multiple correct answer alternatives was also lacking here.

From this, one can derive at least these features. Other features could be added during development from feedback or further discoveries.

- Open and display the diagram image
- Crop the gap images directly and automatically create the corresponding gap elements.
- Position the gap automatically as the element is cropped.
- Snap to grid for positioning.
- Option for multiple correct answers to a field.
- Ability to add distractors (both text and image form).
- "Fixed size" option for gaps, allowing creation of gap elements with correct size and position with a single click.
- Option for erasing/removing selections of the background image
- Window/area for previewing cropping/gap/area selection.
- Generate all the necessary QTI XML code. No needed extra work inside Inspira.
- Display the question so the author can validate it.

3.2.2 Automated item generation

AIG is the process of using a question template and automatically generate variable question content from randomized data or predefined collections of data. Good use cases for this is questions involving calculations like math or physics, where a question can stay the same(template) but the numbers in it are changed for each question.

Generating different QTI questions that uses UML diagrams may be challenging as the diagrams are very specific to a context, and creating a question template for it may not be possible. However there are other forms of diagrams that can benefit for the use of AIG, for example circuit diagrams. A digital circuit diagram consists of various logic gates representing boolean operations. Inputs and outputs in the diagram are binary, i.e 1's and 0's.

An example of an exam question involving this type of diagram could be a task where the student must design a circuit or choose the correct logic gates to comply with certain

specifications like a desired output of the circuit. Another example could be that the circuit is already made, but the student is tasked to determine its output.

There are a number of ways one could implement AIG for circuit diagram questions. With the first question example mentioned, the specifications could be a boolean expression. This expression could then be generated at random, and the correct diagram design would be different each time. With the other example, automated generation of diagrams could be done, and the student would for example be tasked to determine the boolean expression that fits the diagram.

More specifically, one way the process of creating the circuit questions could work was first having the user supply a diagram image, something resembling the illustration below.

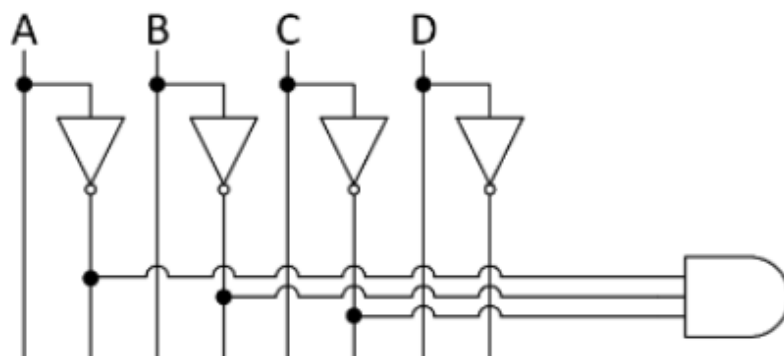


Figure 3.2: Digital circuit schema

Then, a desired circuit condition (boolean expression) would be set, for example $ABC\bar{D}$. The solution for this condition would be having a line from A, B, C and inverse D to the AND gate. The question could be implemented and structured by the use of QTI's "graphicAssociateInteraction", where the test performer has to create an association between available choices resulting in lines being drawn between them, as is the desired behaviour.

This could also be expanded upon by also having different logic gates available as drag-elements. Either integrate them as part of the system, or have the user define them. By not having them integrated, the prototype is potentially open to a bigger range of question subjects, but may also be more complex to do as the logic gate is needed to have some values or computation defined.

An important question is whether or not QTI's "graphicAssociateInteraction" actually works in Inpera. This was not available as an explicit question type option, however as Inpera does support import and export of QTI version 2.2 and the graphicAssociateInter-

action is a part of that version, it was assumed to work. Unfortunately it became clear it was not possible after some testing. Several attempts were made with varying specifications in the files, but resulted in buggy behavior in Inspera, and some helpful information printed in the browser console during import did give some insight. It became more apparent when inspecting Insperas front-end code, which handles parsing of the QTI. There was a switch statement with a case for each supported QTI interaction, and `graphicAssociateInteraction` was not one of them. Because of this, alternative interaction types relevant for this question type needed to be examined.

Gathered from Inspera's source code, the following QTI interaction types were supported

- `choiceinteraction`
- `extendedtextinteraction`
- `uploadinteraction`
- `textentryinteraction`
- `inlinechoiceinteraction`
- `matchinteraction`
- `gapmatchinteraction`
- `graphicgapmatchinteraction`
- `hotspotinteraction`

A few of these could be possible alternatives. `Choiceinteraction` consists of several boxes where the student is asked to tick of one or multiple of them, and can be used in this setting by having one box represent whether or not a line is connected between logic gates. `Matchinteraction` is similar to `choiceinteraction`, with the difference of having to pair the boxes. This might be a better option for having more complex circuit diagrams, as well as offering more answer combinations and potentially gives more freedom when answering and can require more thinking, especially in the case where the student may use multiple logic gates as drag elements. `Inlinechoiceinteraction` consists of having dropdown lists with sets of choices. This can for example be used with one choice list for each line in the diagram, with choices of where to connect it to. `Hotspotinteraction` consists of different spots the student has to select, either based on visible available spots on top of the diagram, or not visible and then select purely based on the question text and diagram

image. Similar to the other use cases here, this can also for example be used where each line may be a hotspot to be clicked to "connect" to another logic gate. Then there is also the option of using `graphicgapmatchinteraction`, where the student would have to drag the lines to the correct position to connect the logic ports. However this may have several limitations, such as requiring the background diagram to be drawn out in a specific way, and scaling the line image to different lengths.

An important aspect to consider when choosing an interaction type is the students ability to clearly visualise their answer, as it is not possible at this time to directly draw lines on the diagram. This may become a problem with `choice-`, `match-`, and `hotspotinteraction` as they are either on or off, or they at least may limit the diagram design. `inlinechoiceinteraction` however provides multiple choices for each point. An example of where this is better is if the diagram have more than one gate to connect to, and each point's dropdown list consist of the various gates. Therefore, it seemed the like the preferred alternative.

After some testing with Inespera, it appeared `inlinechoiceinteraction` did not allow to position the dropdown lists on top of a background image, only above or below the entire background, making it not usable for this case. Instead, the `textEntryInteraction` may be used, where instead of choosing from a dropdown list, the student writes text inside it. In this case, each logic port alternative can be numbered 1, 2 etc, and the student could then write the number of the port they want to connect the line to. However, after some testing, this interaction type also seemed sub-optimal, because the smallest size of the text entry field one could set was still quite big, even though a character limit of 1 was set. It would then span over multiple lines if the diagram was not designed with that in mind.

3.2.3 Optical character recognition

For the type of diagram questions where the task is to map the correct word or sentences in the correct location, the use of OCR might reduce the time required to create the question significantly. As described earlier in the thesis, the current process of creating such question can involve a lot of steps. Recognising text as well as its coordinates in an image automatically could be very useful for making some of the steps easier.

This can be done in a number of different ways, depending on the nature of the question. For example it could be desirable to hide all the text shown in the diagram image, and then ask the student to match text to the correct location. Alternatively it could be desirable to only hide certain parts of the text, and then let the rest be a guide/hint of where to place the hidden text. For this to be an automatic process, the problem arose of how to select

which text elements to hide or not hide.

The OCR software used here called Tesseract, as well as many other solutions, provides multiple ways of accessing and separating the text elements detected in an image. The text can be read as separated lines as displayed in the image if the words are located at approximately the same height, and can also be detected as paragraphs if displayed as such. Then there are the smaller groupings of text elements, *words*, which is characters separated by an empty space, and simply each character symbol by itself.

As mentioned, how text is selected is dependant on the question. For the case where one would want to turn all the text into drag-and-drop elements, one must still decide how to split up the text. For most cases, the smallest unit of text to be created a gap element out of is a word. A note here is that the grouping of text is not necessarily limited to the options provided by the OCR tool, such as words or lines. Alternatively, one could implement some type of filter which groups a number of words together if they satisfy a certain condition. An example is to systematically go through each detected word in order, and if two words in a row are for example 3 characters long, group those two into one element.

The simplest way of creating elements from the text is transforming all the detected words to gap elements, however this would most likely not fit with what the question is trying to test. UML diagrams do not contain a lot of text, but rather words or small sentences adhering to a standard which represents something deeper that is understood by the users of the diagram. Occasionally they may contain some extra descriptive text for clarification purposes, but most contain text phrases consisting of only one or two words. When there are two words or more, which words to select can be challenging. One way to handle this could be by calculating the distance from a group of words to adjacent words to determine if they are connected in a phrase or not. With UML class diagrams for example, a colon is oftentimes used to separate a class function name from it's returning data type. This could potentially be another method of determining word connections.

OCR accuracy is another important consideration with the use of OCR. Accuracy here means how well the software performs at converting an image into the correct text. The OCR tool provides a *confidence* score for each detected character in the output which tells how confident the tool is that the character is correctly detected. However, to measure the actual accuracy, i.e number of correct detections compared to wrong ones, the output of the OCR tool must be compared to the original text [18]. For example with a text consisting of 1000 character, and the OCR tool recognised 950, the accuracy would be 95%.

This would be the character-level accuracy, which is what most OCR engines operates with. Word-level accuracy is calculated the same way, counting correct vs false words, however the confidence level outputted by the OCR tool for a whole word works a bit different. Word-level confidence can depend on different factors, for example telling the OCR engine the text is in only one specific language, comparing with dictionaries, and algorithm used for the calculation based on confidence values for each character in the word.

Accuracy can have a significant impact on the efficiency and general usability of the new IT tool, and is therefore important to have as high as possible. One factor that can affect accuracy considerably is the quality of the source image. This is something the end-user must keep in mind when creating their diagram image. "Quality" does not only mean the resolution of the image, although it is important, there are other aspects to consider as well. A detailed specification of what image quality is not discussed here, but some aspects are contrast, color, sharpness, and pixel noise. Something perhaps more important in this context is the alignment of the text in the image.

Pre-processing of images is one method of increasing accuracy. This is a process that alters the image's features to make it into a more standardized form easier for a machine to read/recognise. The Tesseract OCR engine already includes such pre-processing steps internally before performing the OCR operation, however these steps can be very configurable and Tesseract's default method might not be optimal in all situations. There are many methods that can be used in pre-processing, but because OCR is only one part of this thesis and the IT tool is only a prototype, only a subset of steps were examined such as;

- Scaling
- Binarization
- Removing borders
- Dilation and Erosion

3.2.4 Image recognition

UML diagrams utilize standardized symbols and figures organised in certain patterns, for example a box in a class diagram representing a class. For questions involving matching such boxes to the correct position, it could be possible to use image recognition in order to

automatically detect specific shapes in an image, and then create a drag-and-drop element from it.

Contour detection is the process of detecting the edges, i.e boundaries with the same color, of an object in an image. OpenCV [32] is an open source library that provides computer vision and machine learning utilities, including contour detection. It has an interface for Python among other programming languages with easy built-in functions that makes it very easy to setup and use. In this case, using a Flask server as an API for the frontend that accepts images and returns coordinates for the detected contours.

Discussed in the previous section was the concept of accuracy of detection. This is very relevant and important here as well. There are many methods of improving accuracy, some mentioned above, but binarization is perhaps the most important one, specified in OpenCV's documentation [33].

An UML class diagram consists of rectangles with text inside them, and various arrow types pointing between them. For a QTI question where one would want to have each class rectangle as a drag-and-drop element, only the contours of the rectangles must be extracted from the image, and not other elements such as the arrows or text. OpenCV provides various methods and values one can change depending on the nature of the contours one would want to detect. The position and organization of contours in an image can be seen as a hierarchy arrangement, i.e an image can contain a square box which then contains several smaller boxes and so on, and this is one way of selecting desired contours. OpenCV contains different modes of retrieving the contours, for example one could retrieve them as a tree where all relationships between them are described, or choose to retrieve only external ones, the most outer contours detected. Another factor is contour approximation method. If the shape of the objects in the image are known to be rectangles, one could use an approximation method that only uses the coordinates of the edges of the object.

The QTI question case mentioned with class diagrams is not the only use case of using contour detection. It could be used and specified to target the arrows between the class rectangles, so that the QTI question would involve matching the correct arrow(class relationship) between the correct classes. It could also target other object shapes for different types of UML diagrams, or even unrelated non-UML images.

3.3 Related work

The aim of this literature review is to find work on QTI and test questions involving diagrams/images with drag and drop, technologies involving OCR and generally other themes which may be relevant for this thesis.

3.3.1 Scientific papers

Several articles implement and/or discuss some type of question generation with the QTI format. The implementation of how the QTI XML is generated will be specific and somewhat unique for different use cases and systems. However there may be some knowledge to be gained from other articles which this thesis can utilize or build upon.

One paper in particular [21] is very similar to the goal in this thesis, in that it focuses on the automatic generation of QTI compliant questions compatible with Inspira Assessment for use in IT exams. It looked specifically at Parsons problems. However this thesis will look at how automatic generation of exam questions can be applied to diagram tasks. This article by Jørgensen and Kvannli [21] discusses many details of QTI and Inspira, Drag-and-drop questions, and limitations of this in Inspira. As it is quite similar to how this thesis is focused, the findings of it can be very relevant and be used to avoid potential problems they encountered here.

A paper by Mikalus Gangur [12] is about automatic random generation of cloze questions as well as numeric questions, that are unique for each student. Cloze questions are text questions with missing words to be filled in. It proved relevant here in that it discusses and shows examples of generating/translating XML files to be imported in the LMS(Learning Managment system) Moodle, as well as providing an example of how AIG can be achieved and designed.

A paper by Muriel Foulonneau [11] generates simple QTI questions from DBpedia data in order to evaluate how feasible it is to use linked open data to generate questions for formative assessment tests. It describes the QTI XML format and the steps involved in the generation.

A potentially very useful tool in this project is an optical character recognition(OCR) engine. OCR is the process of converting non-digital text sources such as physical text documents(handwritten or printed), images or videos with text in them into the digital text form. This can be useful for a number of different things for example automatically

reading plate numbers of cars, creating digital copies of physical books that are not only images of the pages but instead text that can be searched, reading passport information automatically etc. The relevance in this project is that it may be used to automatically read text off of the diagram images provided by the user, and use this to create the various elements of a drag-and-drop questions. There are several software solutions available to use, both free and non-free, and some are offered as OCR-as-a-service online. One of the most popular free OCR softwares is Tesseract [44]. Smith [40] gives a detailed overview of Tesseract and how it works. Patel [35] does a case study of Tesseract. It describes it in detail and looks at its performance, as well as how it compares to another OCR tool Transym.

Regarding rendering of the QTI question, [34] can be relevant. It describes the development of a QTI player and editor on the web. It uses XSL to transform the QTI XML into HTML to be displayed in a web browser.

A paper by Aparna Chirumamilla and this thesis' supervisor Guttorm Sindre [7] discusses e-learning and its tools with regards to programming and programming exams. Among other questions types are the drag-and-drop genre discussed here, and to what extent these programming questions are supported in the tools. Inspira is one of the tools analysed in the paper. Therefore the paper provides valuable insight of the context for this thesis, pointing out weaknesses of the current tools.

A paper by Rose Holley [18] identified and analysed ways of improving the accuracy of OCR for digitization of older newspapers. It outlines how the OCR process work and some alternative methods of OCR, factors that can affect the accuracy and details of measuring accuracy. Various image pre-processing methods were discussed, with some of them tested and compared. This knowledge proved useful for the features using OCR in the new tool.

Regarding contour detection in images, the two papers [14] and [45] provided useful knowledge. The former discusses in detail several of approaches to contour detection, differences between them and how they are related. The latter used image edge detection to find the amount of copper cores in wires, using OpenCV and various pre-processing methods such as morphological operations, which were used in this thesis as well.

A paper by Bañeres et al. [2] discusses the use and syllabus of digital circuit diagram design in computer science and engineering education. It mentions the tools used in learning, and the need for manual assessment of the students circuit designs by the teachers. It discussed the potential benefits of automated assessment of such diagrams and developed

a tool that allowed this, as well as evaluating it in terms of students performance after using it. This paper became relevant here for two reasons; the results showed that students scored higher after using the tool, showing the value of automated assessment, and also for the paper's analysis and description of how such a tool can be developed.

3.3.2 QTI editors/tools

In addition to Inspira, there are other tools to create QTI questions and tests as well. Many Learning management systems implement their own QTI editor. Below is a description of some of the most popular ones, and how they compare to Inspira regarding drag-and-drop/graphical type of question creation. Note that not all tools are free or provide a free demo, therefore the researcher may not address all the major tools, with the exception that they provide good enough documentation or guides to understand their features.

OpenOLAT

OpenOLAT [23] is an LMS that is open source, and is based on another LMS called OLAT. OpenOLAT support many features such as setting up and managing courses, collaboration tools, and testing environments with integrated editors for creating tests and questions conforming to the QTI standard. The test editor supports a variety of question types such as single choice, multiple choice, and drag-and-drop, but not other variations of it such as Insperas "graphic gap match". The UI looks similar to Inspira, with similar functionality. Except some different question settings, it is practically the same steps as in Inspira when creating a drag-and-drop question. Here too, one has prepare and crop the images manually before uploading them.

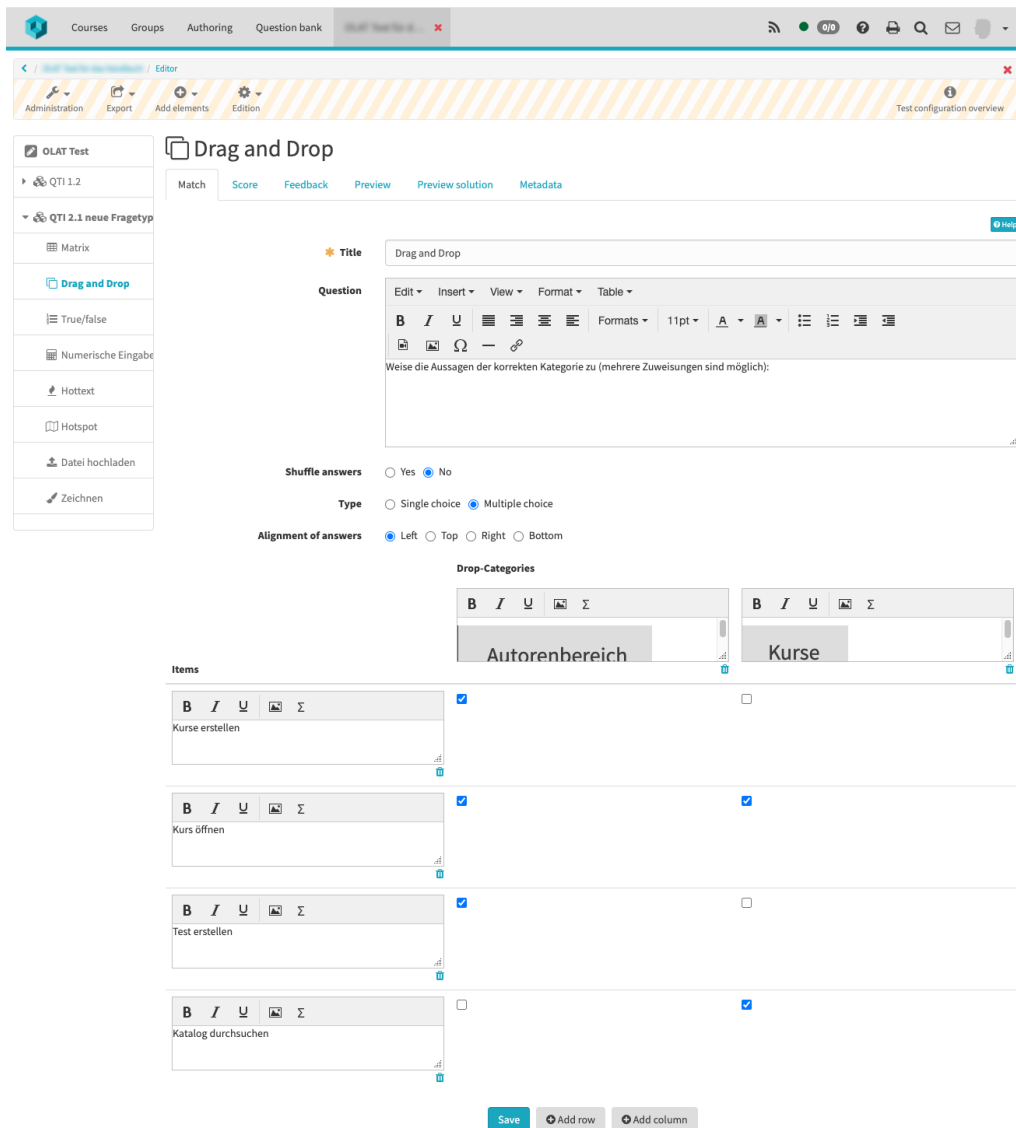


Figure 3.3: Question editor in OpenOLAT with drag-and-drop

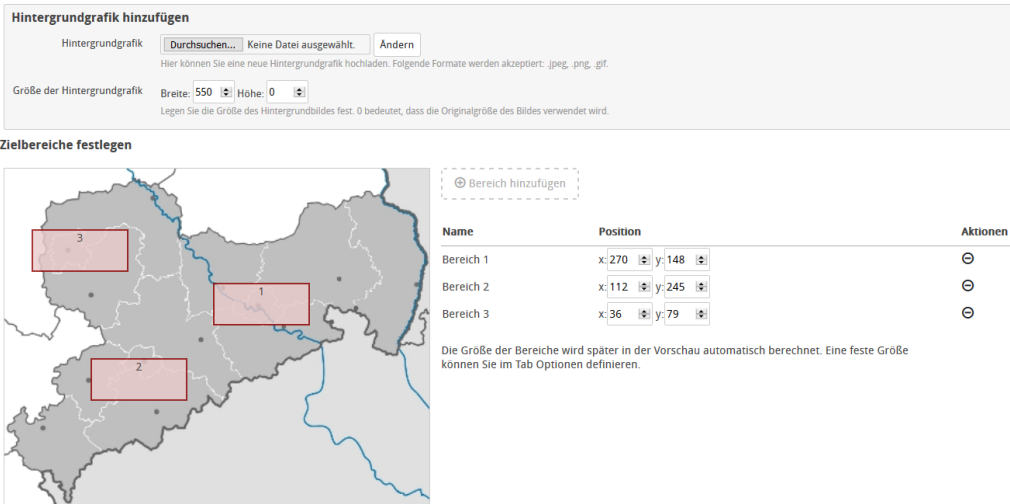
Moodle

Moodle [25] is another LMS and is also open source. It is used for e-learning and is very flexible in nature, allowing the creation and use of custom plugins to support a vast amount of features. Moodle does not actually use the QTI standard, but rather has its own formats. Moodle has three question types for drag-and-drop, namely "Drag and drop into text", "Drag and drop markers", and "Drag and drop onto image". The relevant one being the latter. It is the same type as Insperas "graphic gap match" where text, images or both can be placed into a background image with blank areas. What is interesting with Moodle's editor is that for creating the blank areas, one enters a preview mode of the question and drags the draggable items into the desired places (after one has created

the elements). This is a little bit faster than doing it in Inspira where one has to create each blank area manually and set their correct answer setting, as well as their position. However there is no option here for manipulating the image, such as cropping elements or hiding areas. The blank areas are also transparent, meaning one can not simply hide areas by setting a drop area box on top of it.

Onyx editor

The onyx editor [31] is a standalone tool for creating tests and test questions in QTI format. It offers about the same question types as the others, with a "graphic match interaction" type for the drag-and-drop concept. This question required a pro version licence to create in the editor, so the author could not test it properly. However from the documentation and the figure below, the process appeared to be the same as the other tools.



Hintergrundgrafik hinzufügen

Hintergrundgrafik Keine Datei ausgewählt.

Hier können Sie eine neue Hintergrundgrafik hochladen. Folgende Formate werden akzeptiert: .jpeg, .png, .gif.

Größe der Hintergrundgrafik Breite: 550 Höhe: 0

Legen Sie die Größe des Hintergrundbildes fest. 0 bedeutet, dass die Originalgröße des Bildes verwendet wird.

Zielbereiche festlegen

Name	Position	Aktionen
Bereich 1	x: 270 y: 148	⊖
Bereich 2	x: 112 y: 245	⊖
Bereich 3	x: 36 y: 79	⊖

Die Größe der Bereiche wird später in der Vorschau automatisch berechnet. Eine feste Größe können Sie im Tab Optionen definieren.

Figure 3.4: Onyx editor graphic match interaction

TAO

TAO [42] stands for Test Assisté par Ordinateur. It is a platform for online testing and assessment and is open source. Their QTI question editor has slightly more alternatives regarding question types. Several alternatives are available for graphical interaction (drag-and-drop type). These are graphic order interaction, graphic associate interaction, and graphic gap interaction. Only the latter one is considered here as the others are too specific or limited for the questions this thesis focuses on. This question type only supports images for gap elements and not text. When creating the empty areas, it is done using the mouse

to "draw" a desired shape (rectangle, circle etc) on the desired spot of the background image, which is slightly different and perhaps a better way than the other tools including Inpera. As with the other tools, this one also has no option for editing, creating or cropping the images and must therefore be done manually before uploading. All gap images must(should) be made to be the same size before uploading.

Blackboard

Blackboard [5] is a LMS containing a variety of tools to support online learning as well as face-to-face learning, and course management. Their question creator tool offers similar question types as the other tools, however alternatives for graphic and/or drag-and-drop questions are very limited. The two question types "matching" and "hot spot" are the closest. The "matching" question consists of two columns and the student are asked to match one with the other by the use of a dropdown with alternatives on each row. The "hot spot" question consists of having a image and the student is asked to click on the correct position. An example of this can be seen in the figure below.

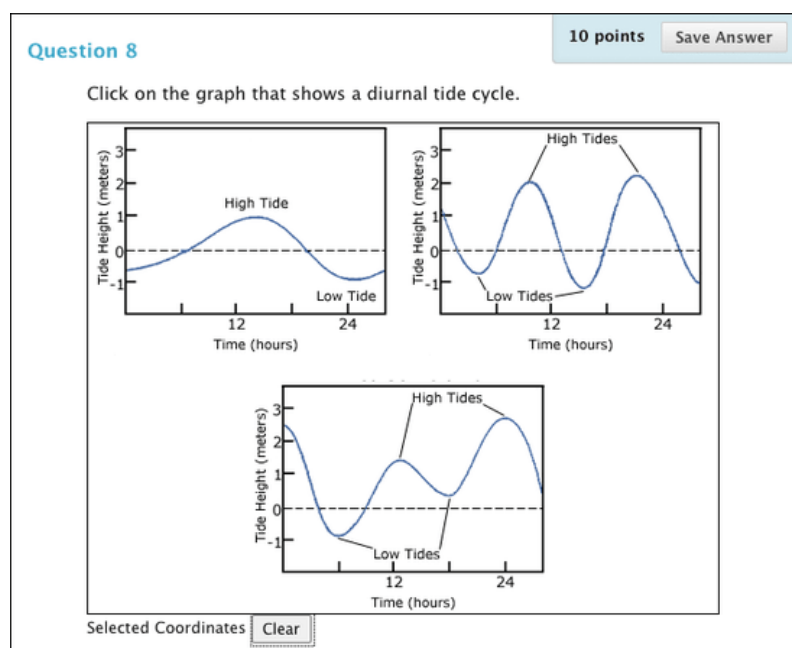


Figure 3.5: Hot spot question in Blackboard

3.4 Theory

3.4.1 QTI

QTI, Question and Test Interoperability, is a standard created by IMS Global Learning Consortium with v0.5 released in 1999. Several iterations have been done over the years with v3 being the most recent, released in 2019, and v2.1 being the most widely used. The goal of QTI is to be able to have a common standard to share, reuse and exchange test units across platforms/systems/technologies.

To cite IMS [9]: *“Specifically, QTI is designed to:*

- *Provide a well documented content format for storing and exchanging items independent of the authoring tool used to create them.*
- *Support the deployment of item banks across a wide range of learning and assessment delivery systems.*
- *Provide a well documented content format for storing and exchanging tests independent of the test construction tool used to create them.*
- *Support the deployment of items, item banks and tests from diverse sources in a single learning or assessment delivery system.*
- *Provide systems with the ability to report test results in a consistent manner.”*

QTI XML Structure

Questions and tests are stored in XML(source) format. There are a variety of elements which can make up a test. The most basic one is an assessment item. An item is usually a question or task and contains any number of "interactions", which are ways to respond to the question or complete the task. Furthermore the item includes the elements itemBody, responseDeclaration, outcomeDeclaration, responseProcessing and modalFeedback.

ItemBody is the core element of the item, containing specific interactions for the question. An interaction is a specification of how the user will interact with the question, which can be for example the choice interaction giving alternatives the user has to click on, or the extended text interaction which gives a text area to be written in. The itemBody can also have other content defined with a subset of HTML elements, such as the <p> tag to give some extra explanatory text.

ResponseDeclaration contains information on the question's correct answer. The itemBody can have multiple interactions, and therefore multiple responseDeclarations can be used. OutcomeDeclaration holds outcome variables which may be used for several things, but most often is the calculation of score. ResponseProcessing is where the answer is processed and defines what to be done after the question has been answered. ModalFeedback contains possible feedback, which can be presented conditionally by the ResponseProcessing. Below is an example of an assessmentItem accessed from imsglobal [10].

```

▼ <assessmentItem xmlns="http://www.imslobal.org/xsd/imsqti_v2p1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.imslobal.org/xsd/imsqti_v2p1
  http://www.imslobal.org/xsd/qti/qtiv2p1/imsqti_v2p1.xsd" identifier="choice"
  title="Unattended Luggage" adaptive="false" timeDependent="false">
  ▼ <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="identifier">
    ▼ <correctResponse>
      <value>ChoiceA</value>
    </correctResponse>
  </responseDeclaration>
  ▼ <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float">
    ▼ <defaultValue>
      <value>0</value>
    </defaultValue>
  </outcomeDeclaration>
  ▼ <itemBody>
    <p>Look at the text in the picture.</p>
    ▼ <p>
      
    </p>
    ▼ <choiceInteraction responseIdentifier="RESPONSE" shuffle="false" maxChoices="1">
      <prompt>What does it say?</prompt>
      <simpleChoice identifier="ChoiceA">You must stay with your luggage at all times.
      </simpleChoice>
      <simpleChoice identifier="ChoiceB">Do not let someone else look after your luggage.
      </simpleChoice>
      <simpleChoice identifier="ChoiceC">Remember your luggage when you leave.</simpleChoice>
    </choiceInteraction>
  </itemBody>
  <responseProcessing
  template="http://www.imslobal.org/question/qti_v2p1/rptemplates/match_correct"/>
</assessmentItem>

```

Figure 3.6: QTI example

Figure 3.6 shows a multiple choice question with an associated image. Choice A is set as the correct answer in the responseDeclaration. ItemBody contains one interaction, the choiceInteraction. A standard template called “match_correct” from IMS is used in the responseProcessing, which will set the outcome variable with identifier “SCORE” depending on correct or incorrect answer.

The element assessmentTest defines the test as a whole, and includes multiple assessmentItems and other information such as time limit and the order of questions. AssessmentTest also has its own outcomeDeclaration, outcomeProcessing and testFeedback. It is structured such that it is split up in testParts, which is split further up into assessmentSections which then contains the test questions assessmentItems.

3.4.2 Inspera

Inspira Assessment [19] is a digital assessment and examination tool with support for 25 different types of questions. It can make exams very interactive with use of video, audio or and/or images. A lot of work for teachers and professors are potentially simplified compared to exams done on paper, by having automatically graded questions, automatic detection of plagiarism, digital papers instead of physical, and more.

Even though the QTI standard is supposed to be a format compatible across different systems and platforms, it is not necessarily the case. There are of course different versions of QTI as mentioned, but also, different systems might implement slight variations to certain elements such as the question types. For this reason it is important to look at Inspera's QTI implementation, so that the XML file for a question created in the new application is compatible and can be uploaded to Inspera and recognised, and ideally also be able to be viewed and edited in Inspera's question editor.

A way to do this is to look at the files when exporting a drag-and-drop question made in Inspera. When exporting the question, a zip file is received with following structure:

- **resources**
 - ID_94564125.png
- ID_94567036-item.xml
- imsmanifest.xml

The resources folder contains the background image(s). The imsmanifest file contains some metadata and links the other files together. The "-item" file contains the actual question. Looking in the file, there are not many obvious Inspera deviations. Inside the <assessmentItem> tag is the attribute xmlns used to define a namespace called `inspera`:
`xmlns:inspera="http://www.inspera.no/qti"`.

This namespace is then used various places, among others is for the attribute "object-Type":

```
inspera:objectType="content_question_qti2_graphicgapmatch_v2"
```

which defines the question type.

`content_question_qti2_graphicgapmatch_v2` appears to be specific to Inspera and not general for QTI.

In the `graphicGapMatchInteraction` tag inside the `itembody` is also more use of attributes in the `inspera` namespace like `"tokenPosition"`, `"tokenOrder"` etc, and unless other platform uses these same attributes, it will probably not be compatible with them, at least not with the exact same appearance, but this remains to be tested.

3.4.3 UML

UML [30] stands for Unified Modeling Language and is a standard for creating visual models. The use cases for UML are vast, as it can be applied to a variety of domains such as business processes and system architecture. UML can be used for many different diagrams, and there are three main categories. Structure diagrams include for example class diagrams, behaviour diagrams include for example use case diagrams, and interaction diagrams include for example sequence diagrams. UML uses the XMI format, however many applications implement their own flavor of it making interchangeability more difficult. In this thesis only the image file of the diagram is used.

3.4.4 User interface

This thesis is focused on the user experience and the user interface, as it aims to make the application more usable than existing tools. Several sources and guidelines could be used when designing the UI. Don Norman's [28] work is well known within the world of design. His six principles of design; visibility, feedback, affordance, mapping, constraints, and consistency, could benefit the new tool.

Then there are also the 10 usability heuristics for user interface design by Jakob Nielsen [27]; 1. Visibility of system status, 2. Match between system and real world, 3. User control and freedom, 4. Consistency and standards, 5. Error prevention, 6. Recognition rather than recall, 7. Flexibility and efficiency of use, 8. Aesthetic and minimalist design, 9. Help users recognize, diagnose, and recover from errors, 10. Help and documentation.

In addition to this, one could also consider the Web Content Accessibility Guidelines (WCAG) and Authoring Tool Accessibility Guidelines (ATAG) [36].

Chapter 4

Result

4.1 Design and creation

4.1.1 Non-functional requirements

The application is mainly aimed to be used by professors and teachers, but can also be used by student assistants and such. People use different computers and operating systems, and therefore the application is required to be available on multiple systems. Specifically, it should be supported for Windows, MacOS, and Linux(Ubuntu).

The application needs to be easy to use and understand. Since it is supposedly an improvement of Inspira, a minimum should be the level of Inspira, but preferably better. This is important as otherwise few or no users may adopt it.

As with the goal of this thesis, the application should of course be more usable than Inspira for creating drag-and-drop diagram questions, for all of the three metrics used.

Performance is also a factor. Delays or general slowness of the system could be very frustrating to deal with as a user, and therefore every interaction the user does with the system should be immediate.

Security is an important aspect to consider, as the application will be dealing with exam questions, and therefore may be subject to potential students wanting a way to cheat by obtaining the questions before an exam. The created questions should securely handled, and unauthorized access to view them should be prevented.

4.1.2 Development methodology

Methodology framework

Multiple development methodologies were considered for this project. The development process of the application begins with a simple prototype with basic functionality, and then be built upon in multiple iterations, getting feedback and more knowledge underway. Therefore an agile approach was fitting. There are multiple agile frameworks suited for different situations. One major factor for this project was that it only had one single developer.

Scrum [41] is one of the most popular agile frameworks used. Scrum uses set time periods for specific work to be completed called sprints. An important part of scrum is the team and roles. It consists of a scrum master, a product owner and developers, each with specific responsibilities. Scrum activities other than the sprint include sprint planning, daily scrum, sprint review and sprint retrospective. Scrum also has artefacts such as sprint backlog for planning and organizing work. It is however very team-centric and focuses on management and different roles of team members, which was not that relevant here.

Kanban is a framework where in contrast to scrums fixed sprints, uses a continuous flow where release time of working software is up to the team. Kanban is not only applicable in IT but also other business and work processes. It is a set of principles such as to limit work being done simultaneously. Incremental change is focused in contrast to implement a whole system at once. Visualizing the workflow is an important aspect, and to do this a kanban board is used.

XP (Extreme Programming) consists of a set of values and practises very specific to programming. Work is done with regards to a weekly cycle and a quarterly cycle. Some of the practises include pair-programming, test-first programming(test driven development), incremental design, continuous integration, and energized work(40 hour work week).

It would most likely be excessive to implement a whole methodology framework in a one developer project, however certain activities and practises in them are still useful and applicable. The most important practises that are common in several frameworks and will be used in this project are:

- Incremental development
- Test-driven development(TDD)

-
- Feedback

Tools

To support the development practises and visualize progress and planning, a kanban board, specifically the tool called Jira, will be used. Git will be used for version control, and Github to host the repository remotely.

4.1.3 Technology

Platform

The first decision regarding technology use was web application or standalone program. There are several reason to choose either, and some major differences are discussed here. Not all of them are equally relevant or important, but are at least given some thought.

An advantage with a desktop application is that it can (if designed to) function without access to internet. With a slow internet connection a web application may be severely impacted and become difficult to use or not usable at all, causing frustration of the user. A web application may also be prone to downtime due to server crashes or other reasons.

Security is another point which favors the desktop's side. Here, the user has more control of the environment and their files, but of course it is not without its own security risks. Meanwhile with a web application there are more risks and more overhead. More potential vulnerabilities have to be examined and mitigated as one has to manage access, data transfer between client and server and more.

Something perhaps less important but may also be worth considering depending on the applications nature is performance. Usually a desktop application performs faster than web, as it is "closer" to hardware. There is not necessarily any need for data communication to a server as with a web application. However a program with heavy calculations can in a web application be made to be done on the server side and then the server may be scaled until performance is sufficient.

Concerning accessibility, several points are relevant. With a desktop application, it has to be installed in order to be used. This means if the user has multiple devices, it has to be installed on each one separately, where a web application is available from a web browser which most devices already have. Another point concerning the use of multiple devices

is platform support. As mentioned the web application is accessible from a browser, which means only one code base is needed to be developed, with potentially only some extra code to support different browsers. For a desktop application to be supported on multiple platforms, there is usually extra development work involved with a separate code base for each platform such as windows, mac etc. However there do exist some cross-platform development frameworks which allows for a single code base, usually with some compromises.

The process of updating the system is also different between the two. From the perspective of a developer, it is a lot easier to update the software for a web application, as it is centralized and fully controlled. It is somewhat more difficult for a desktop application. From a user's perspective, they might want to keep using a specific (older) version for either personal reasons or other. This is quite easy for desktop applications as one can simply either not update, or download older versions manually, while it is often times not possible in a web applications unless it is specifically developed to support it which requires more work.

A small factor to mention is storage space usage. A web application takes up practically no additional space in the user's devices, while the desktop application does. This could be either trivial or an actual factor depending on the application.

The user interface can be important to consider. For a desktop application, it is generally expected to comply with the platforms guidelines and design "theme" more than for web, or else users might find it unattractive to use.

A web application also has the additional cost of hosting, although a desktop application might also require to communicate with a server depending on its function. A domain name is also necessary to purchase unless a sub domain of another service is used. The cost of this can vary widely. However there are free hosting and domain services available, many popular services offers a free tier where some functionality, most often traffic, is limited.

Even though the new tool is be more of a prototype than a fully finished product, it is still important to plan for future work and development, as design and technology choices can have big consequences both positive and negative later down the line. From the points discussed above, either of the alternatives would probably be fine, but the decision landed on a web application. The main reason for this was simply the ability to access it anywhere. This would make it easy to ask users to test the tool along the development, as well as for the experiment, without the hassle of downloading and installation.

Development frameworks and languages

In web development, there are countless of frameworks available for use, as specifically front-end has and still is evolving fast.

System architecture and requirements are central when deciding which is the best alternative, though many of them can be quite similar and will not matter much for which one are used.

This new tool is quite UI centric with not alot of other complexities, such as a login system for users, intricate data objects and database management etc. It could potentially be done without a backend, but a backend could be useful to handle the QTI XML file generation, as well as for uploading questions to Inspera. It could also handle storage of questions with a database. Since good UI/UX is the focus of this project, a Single-page applications (SPA) is a good choice as it allows the user to navigate and interact with the site without having to request a new page from the server on each click. Instead, the entire site is loaded once initially, and then potentially perform some lightweight communication with the server where necessary. A client-server architecture was used with a minimalist (at least initially) API as the backend. Below is a basic illustration of this adapted from [24].

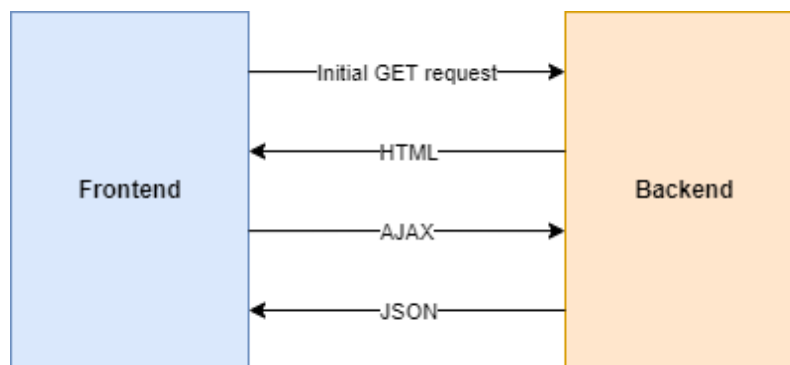


Figure 4.1: Single-page application

In a traditional site, the response from the server is usually a new HTML file, but in an SPA only a small amount of data is transferred.

With regards to the specific frameworks/libraries client side, there were specifically three that stood out and are used extensively in the industry; React, Angular and Vue. In many ways they are quite similar. Detailed description of all the differences is not discussed here, also many of the differences are subjective of preference. To mention some of the reasons React was chosen, first and foremost the researcher was already more familiar with it compared to the others, which shortens development time. It has a strong com-

munity which means there are a lot of resources and existing components available online. It uses JSX(Javascript XML) which has a number of advantages such as readability and being fast and easy to write. It uses virtual DOM which makes for a fast render, and it allows easy reuse of components.

On the backend side, it was less important which framework is used. It was also unknown at this stage to what extent it would be used. Therefore minimalist frameworks such as Flask or Express were logical choices.

An important element to consider when deciding the technologies was the OCR software. Discussed earlier in this report was the OCR software Tesseract. The main Github repository for Tesseract contains the OCR engine as well as a command line program for it. The engine has C and C++ APIs available, but would add extra complexity to the application. There are many 3rd party projects that utilize Tesseract like programs with a GUI and even online OCR services, which might be useful. Tesseract also has a variety of wrappers for other programming languages like Java, Python and Ruby to name a few. Since (for now) only a minimalistic framework was needed, the python project tesseractocr could be very useful as it provides very easy to use code to retrieve text from images, as illustrated in the figure below adapted from [39].

```
1 import tesseractocr
2 from PIL import Image
3
4 print(tesseractocr.tesseract_version()) # print tesseract-ocr version
5 print(tesseractocr.get_languages()) # prints tessdata path and list of available languages
6
7 image = Image.open('sample.jpg')
8 print(tesseractocr.image_to_text(image)) # print ocr text from image
9 # or
10 print(tesseractocr.file_to_text('sample.jpg'))
```

Figure 4.2: Example of simple image-to-text with tesseractocr

The tesseractocr wrapper for python in the Flask framework seemed like a good approach for handling the OCR. However, there was also found another viable alternative. A project called Tesseract.js [26] is a complete javascript port of Tesseract using emscripten to port the engine from C to webassembly. This would make it possible to do the OCR handling purely client-side.

It was at this point not set as a requirement, but the ability to store the questions created in the application could be considered. If this was added, one would also need to handle security and authentication properly. The Flask framework does have libraries for session

management, but thorough testing of the entire application would then be needed to make sure there are no major vulnerabilities. In the early stages at least, it would be enough to let the users download their questions to their own device, or possibly adding the option of uploading it to Inspira directly.

Uploading questions directly to Inspira from the application could be difficult to implement. Inspira did offer an API (Inspira Open API) that could be used to access some of the features of Inspira Assessment which requires user privileges, however it appeared to be limited to only send/retrieve data about candidates, tests, users and some other types, but not the possibility of importing a test or question item.

One approach for uploading the questions would be to automate all the steps involved when doing it manually in Inspira. This could be done by looking at the HTTP requests and responses and automate each of these, or using a tool such as Selenium, which is available in python, to simulate the correct inputs to perform a task. However, small changes made by Inspira could suddenly result in the function to be broken.

Web hosting

There are many alternatives to chose from for hosting the tool. The main criteria was that it was free to use, and two alternatives are compared here.

Heroku is one of the most popular free hosting services. It support a number of different programming languages and frameworks, including Flask for python, by using containers called dynos that can be deployed automatically when a branch on Github is pushed to. Its free tier has 512 mb of RAM, and has a default of 550 "dyno hours" each month meaning 550 hours where the application is active. When the application is inactive for 30 minutes, it goes into sleep which makes the next connection to the server take a bit longer. This is fine at least when prototyping.

Google Firebase is also a popular option. Firebase's Cloud Run is similar to Heroku, with hosting containerized applications. It supports as many or more programming languages. The free tier has a limit for CPU seconds, memory, number of requests, and network. This limit will most likely not be reached when prototyping.

Both of these two are good choices for this project, but the decision landed on Heroku for being very easy to use and quick to set up.

4.1.4 Initial setup

Environment

As described earlier, React was the main chosen technology to develop with. Setting up the development environment was done with one single command "npx create-react-app" using Node.js.

Jest, a javascript test runner, and React Testing Library comes bundled with create-react-app by default, and was used to create several test cases. Because of the nature of this prototype, with many features bound to the use of HTML Canvas elements, it is more difficult to mock correct application behavior with these kinds of tests. Conducting tests in a real browser environment is more ideal. Therefore the testing framework Cypress was integrated and handled the main testing.

Priority

A question which arose when initially starting to work on the prototype was how much time should be spent on designing the UI, and whether or not to prioritize UI or functionality first. The UI could have a big impact on the test users experience when evaluating the prototype, however it was decided that it was more important to have the proper functionality in place before focusing on UI because efficiency was the main target in the evaluation.

4.1.5 Iteration 1

The first iteration of the prototype was planned to reflect the main features described in the preparatory project and section 4.1.1. It consisted of the following requirements and tasks.

Requirements:

ID	Requirement description
1	The tool should have an understandable GUI
2	The user should be able to upload image files and view them
3	The user should be able to create drag-and-drop elements
4	The user Should be able to erase/cut parts of the image
5	The user Should be able to create distractor elements
6	The user Should be able to modify a drop area's correct answer
7	The user Should be able to erase/cut parts of the image
8	The tool should be able to export QTI compliant files recognisable by Inspira

Table 4.1: Iteration 1 requirements

Tasks:

ID	Task description
1	Create a simple GUI
2	Implement area selection tool with a preview area
3	Implement the various user GUI actions
4	Implement XML parsing for generating the QTI XML files

Table 4.2: Iteration 1 tasks

Implementation

A simple UI was designed with the probability of it being revised in the future in mind. It consisted of a button for each action the user could perform, an interactive display of the background image, and an area for displaying the drag elements. Most of the architectural decisions and skeleton code were done here, such as defining how the various React components should look like and function, and state management.

One of the challenges here was to generate the XML code correctly on export, and for Inspira to interpret it correctly. It was early on discovered and decided it is easier to utilize pre-defined QTI XML code and fill in or edit the blanks rather than generating all the necessary code, as much of it was reusable. The first step was to export a blank question from Inspira and set it up to be used as a template, and then add in the content.

As mentioned earlier in this thesis, Inspera exports two files `imsmanifest.xml` and the question item file, as well as a resources folder containing the images.

QTI elements for things such as scoring or feedback were not looked at at this point, and therefore left to the default values obtained from the Inspera export. The main focus was getting the question to behave as intended in Inspera, as well as display all elements correctly which also posed some challenges. There were some problems with the background image not being scaled properly, and drag elements and drop areas not having the correct size or position. This was solved by a trial and error process of exporting and importing to find out how these values are calculated in Inspera, as there was no documentation for this particular issue available.

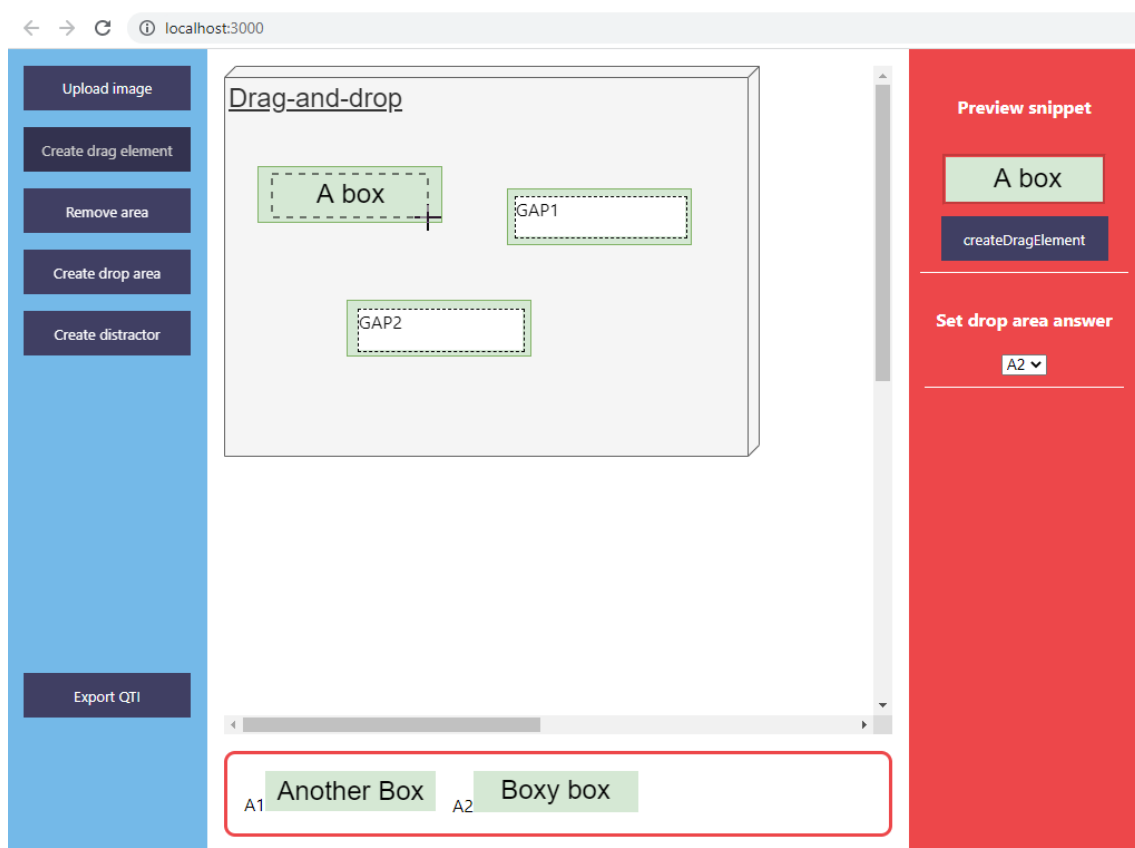


Figure 4.3: UI after first iteration

Figure 4.3 above shows the resulting user interface after the first iteration. A sample image was created and uploaded to demonstrate, which is the whole gray box in the middle. The create drag element feature was used to mark areas on the image to create the drag elements seen on the bottom.

Testing

Automated tests were created using Cypress, that goes through the application's features like uploading an image, creating the different elements, creating distractors, changing values etc. One thing such automated tests could not do, or at least not in an easy manner, was exporting the resulting QTI and import it to Inspera for further testing. Manual testing for this was conducted with multiple variations of elements and user input in the process to ensure the QTI XML code was functioning as expected.

Evaluation

Some takeaways in this iteration from manual testing and feedback was adding an option for setting a fixed marking size. This could be a useful addition for having the elements be the same size, as it was difficult to consistently mark multiple areas equally using the mouse. Another thing was the lack options to "undo" performed actions. One could change a drop area's answer, but one could not delete a drop area or drag element once created. Instead the user had to restart from scratch if they made a mistake. There was also no option to set the question's title and description, reason being not prioritized however would be included in the next iterations.

The prototype and the project in general was presented in a meeting with a professor from the department of Electronics and Telecommunications, as well as the supervisor of this thesis. This was done in order to get feedback and general thoughts for further development or changes, and because it was believed that subjects involving circuit design could be a good match for this project. More knowledge was obtained, and ideas for more use cases and specific types of automatically assessed question were proposed and discussed. One of these were the possibility of automatically generating questions, which could potentially be very useful for practising purposes for students. Specifically, a question may provide a partial digital circuit, and then ask to design the circuit further to satisfy a condition, where the condition is automatically generated.

4.1.6 Iteration 2

This iteration was focused around the implementation of OCR in order to automate the creation of drag-and-drop elements from text in the diagram. The following requirements and tasks were formulated.

Requirements:

ID	Requirement description
9	The tool should be able to automatically detect text in the uploaded image and select relevant words/sentences
10	The tool should make automatically detected text elements into drag-and-drop elements

Table 4.3: iteration 2 requirements

Tasks:

ID	Task description
5	Install and integrate the Tesseract.js OCR library
6	Define and implement the rules of what text should be targeted for automatic detection
7	Ensure acceptable accuracy of text recognition with Tesseract.js

Table 4.4: Iteration 2 tasks

Implementation

The first step was to get the prototype to recognize a test image and return its textual content. This proved to be a simple task using Tesseract.js API example code, although not all text in the images tested were recognized.

Next step was to retrieve the coordinates of text. One of the ways of reading the output in Tesseract is with the "words" object property, an array of substrings separated by space, containing among other things positional data of the bounding box. Then the bounding box pixel coordinates were used to create a drag element and corresponding drop area.

In order to make it into a proper question, only specific parts were desired to be transformed into elements. One example of a case is where the data types of attributes and functions are hidden and must be dragged in place. This problem has multiple potential solutions. One would be to compare each word detected against a pre-defined list of words containing data types, like int/integer, string, bool/boolean, float etc. The main issue with this is that it does not work with custom data types, but a solution could be an option for the user to write in the extra words they want, or simply just clicking on the word position in the diagram could be done. Another solution could be to require the diagram text to be written in a common standard, where certain text is expected to

be positioned a certain way. For example for an attribute in a class diagram, a common way to write it is <attribute name>:<data type>, and for a function <function name>(<parameters>):<data type>, and variation of having spaces around the colons. Using this, the data type could be found by looking at the word following a colon. The downside with this approach was as mentioned, the diagram has to be designed with this in mind, and existing diagrams the user possess may need to be altered before this functionality in the system can be used.

Both approaches had their pros and cons, but it was decided to first prioritise the latter one, as it is a very common convention of writing. After implementing this feature, another problem was discovered during testing with different UML class diagrams. Since a word in the context of Tesseract.js is only characters ending with a space, it will define a string such as "(boolean)" as a word including the parenthesis. A UML class diagram may contain a class with a function with included parameters, for example "addNumber(number: int): void", which will lead to "int):" being detected as a word. This can make it very obvious as to what the correct answer is for the student answering. The solution was to use Tesseract's "symbols" property to find the coordinates of each symbol and cut the unwanted character(s). This unfortunately proved to be unsuccessful, as the bounding box values for each character appeared inconsistent and oftentimes having overlapping coordinates. The requirement for the user to leave proper spaces in their diagram text remained.

Testing

Testing was performed with some newly created diagram images, as well as a few random ones collected from various sources online. During testing, it was discovered that Tesseract's OCR had trouble detection bold fonts. Although it was not used much in the diagrams, but sometimes bold font was used for the class name at the top of each class. At this point, it was not really a problem, as the class name was not targeted for drag-and-drop anyway, but could potentially be a problem later.

Other than bold font, the specific font used, font size, background color, color of class boxes, and line thickness were some of the properties that could affect the accuracy of detection the most. It was difficult to adjust parameters in a way for the OCR to be accurate with all the diagrams, as the mentioned image properties could vary quite a lot from image to image even when they would be the same diagram (class diagram).

Evaluation

Based on the feedback received from the supervisor, there was also another case that might be useful, which was making not only the data type as a drag element automatically, but the whole attribute/function name. Tesseract did not have an easy way to implement this as it did not provide list of "sentences" or groups of words in close proximity of each other as an output alternative. Instead, a "lines" output was used, and the distance between the bounding boxes of each word was calculated and compared to assess whether or not a word was located at a reasonable far distance from the others and deemed not part of a particular sentence or word group. This assumed each sentence was contained to a single line, which should be the case for class diagrams.

With this feature came a problem that if the diagram contained text outside of the class boxes, for example a number or symbol by the relationship arrows or some other descriptive text, it creates drag elements of them, or includes those words in a sentence for a class member close by. This may destroy parts of the diagram, give hints to the student about the correct answer/placement, or otherwise cause unwanted behaviour. One solution could be to ignore single character words or words containing only digits/special characters, although this does not allow the diagram to have other descriptive text outside the class boxes. Another solution was to automatically detect the box shapes in the image, and only perform OCR on these separately. Since box detection was planned to be used in iteration 4, this solution was delayed until then.

4.1.7 Iteration 3

This iteration was rather small, only focusing on one specific feature. It was built on an idea mentioned in the previous iteration, that by the use of OCR, one could implement the ability for the user to simply click on words/text elements in their diagram and automatically make it into a drag element and drop area. This made creating a drag element and corresponding drop area out of text content much more easy and efficient compared to the manual method.

Requirements:

ID	Requirement description
11	The user should be able to click on words in the image to make them into drag-and-drop elements

Table 4.5: Iteration 3 requirements

Tasks:

ID	Task description
8	Update the GUI to make it evident that words in the image can be clicked after performing OCR
9	Implement drag-and-drop element creation on word click

Table 4.6: Iteration 3 tasks

Implementation

The implementation for this was similar to the previous iteration. First, a button to perform an OCR operation on the diagram image was added. As the goal was to target every word in the diagram, Tesseract's "words" array output was used for coordinates, and no rules were implemented to filter certain text as in iteration 2. Instead of making every detected word into drag-and-drop right away, they were instead made into "buttons" the user could click, that would then create drag-and-drop.

Depending on the diagram image used and how much text it contained, the OCR operation could vary quite a bit of how much time the detection would take, sometimes lasting several seconds. When the user triggered an operation, there was at this point no indication for the user to see if the operation was ongoing. If it would take several seconds to perform, but the user could not know, they might think they did not click the button correctly or did something else wrong, which could negatively impact usability. Therefore a loading animation was also added.

Testing

Manual testing was conducted with various diagrams with different fonts, colors, image dimensions etc to assess whether all words could be detected. As with iteration 2, various

diagram images gave varying results of how many words were detected. Still, the feature worked in fact quite good, and very simple to use. Something that the user could be vary of was that the accuracy sometimes dropped drastically if some form of gradient color for the background was used, and should be avoided.

Evaluation

This feature was not only beneficial to UML diagrams specifically, but also any kind of diagram, figure or image containing text that would be desired be made into drag-and-drop elements. It was demonstrated to the supervisor and received positive feedback. Although not every word would be detected for all kinds of images, it would still be useful as the user could always resort to using the manual method if some words were not detected.

4.1.8 Iteration 4

This iteration was about automatically detecting the boxes representing classes in a UML class diagram image, and then create the drag and drop elements out of those. It also addressed the problem of unwanted behaviour discussed in iteration 2.

Requirements:

ID	Requirement description
12	The tool should be able to automatically detect classes in a class diagram

Table 4.7: iteration 4 requirements

Tasks:

ID	Task description
10	Set up and initialize a Flask server
11	Update the GUI to include option for class detection
12	Create a route in Flask and create the appropriate http request in the frontend
13	Implement contour detection with OpenCV in Flask
14	Create option of performing OCR only within detected contours

Table 4.8: Iteration 4 tasks

Implementation

The first step here was to set up a Flask backend server and utilize OpenCV to detect the rectangles in the image, and send back the coordinates and dimensions to the client. This resulted in the design shown in the figure below:

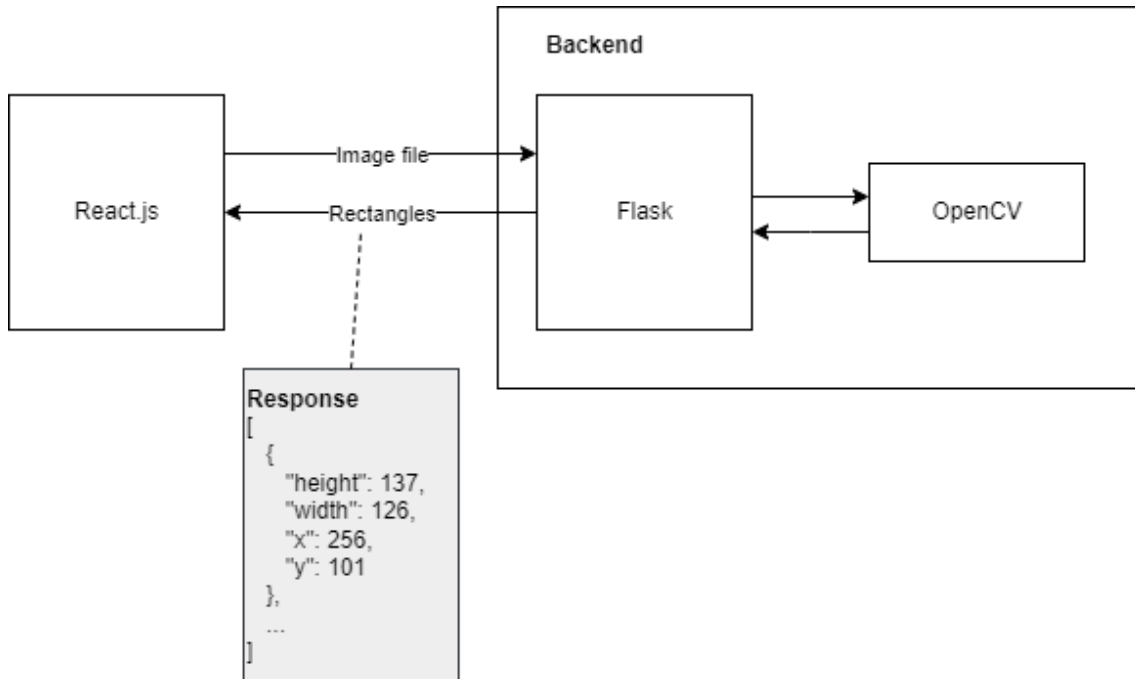


Figure 4.4: Figure of communication between client and server

Before attempting contour detection, the image needed to be pre-processed in order to improve accuracy. The image was first made into grayscale and using the thresholding method, made into a binary image (binarization). The contour detection worked best when detecting white objects in a black background. Since UML diagrams in most cases have a white background with black lines forming the rectangles, an inverse thresholding type (THRESH_BINARY_INV) was used to flip the colors. One problem with detecting rectangles in a class diagram was that the arrows associating the classes with each other may form additional rectangles or cause the detected rectangles to have the arrow lines as their border. To solve this, some additional operations needed to be performed. First the contours were filled, and then the morphological operation called *opening* was done to remove the noise, or arrow lines outside the rectangles.

In addition to the arrows interfering with the contour detection, there was also the case where a class rectangle could contain additional rectangles. The way a class in a class diagram often is constructed is with an outer rectangle containing the whole class, and then three smaller rectangles inside that. The first one containing the class name, the

second containing attributes, and the third containing functions. OpenCV provided several contour retrieval algorithms meant for different tasks. The one useful in this case was *external retrieval mode*, where the extreme outer contours were targeted, in this case the outer rectangle of a class.

The following figures shows the process of turning the class rectangles in an image into drag-and-drop elements:

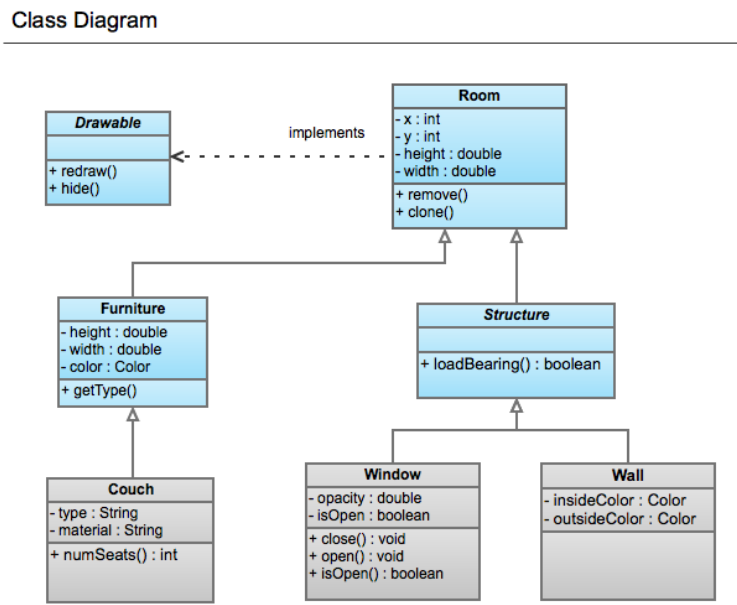


Figure 4.5: Contour detection step 1: original image

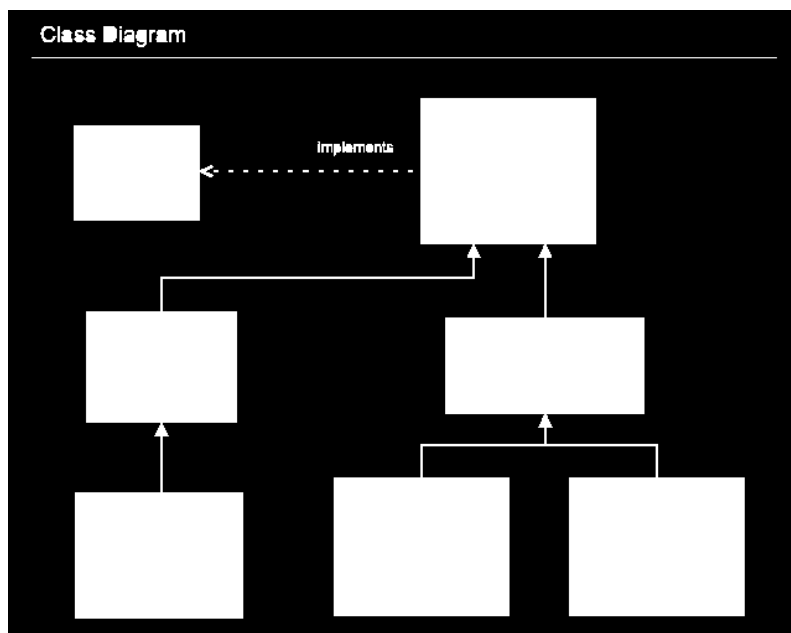


Figure 4.6: Contour detection step 2

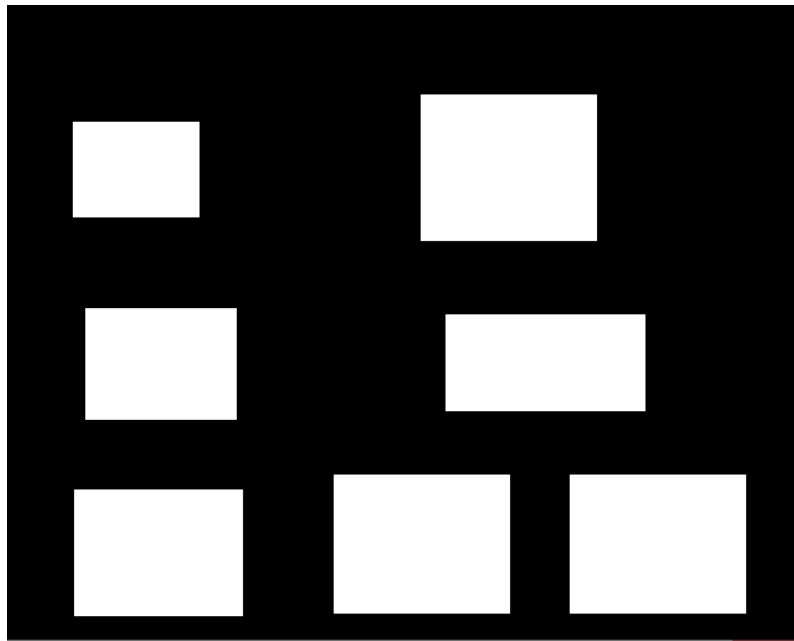


Figure 4.7: Contour detection step 3

Class Diagram

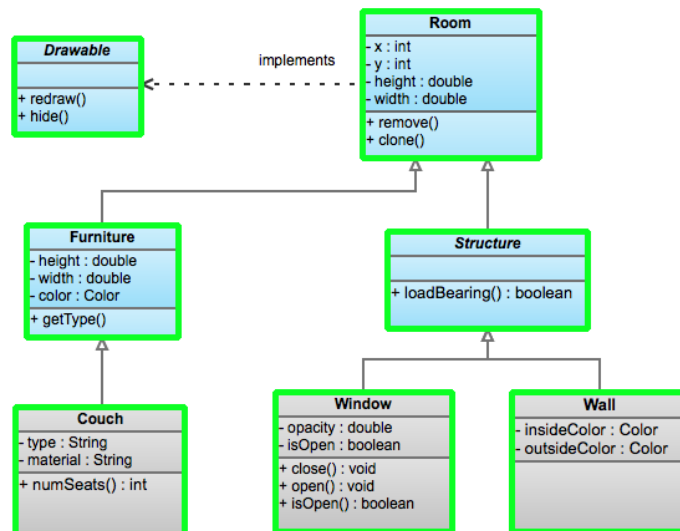


Figure 4.8: Contour detection step 4

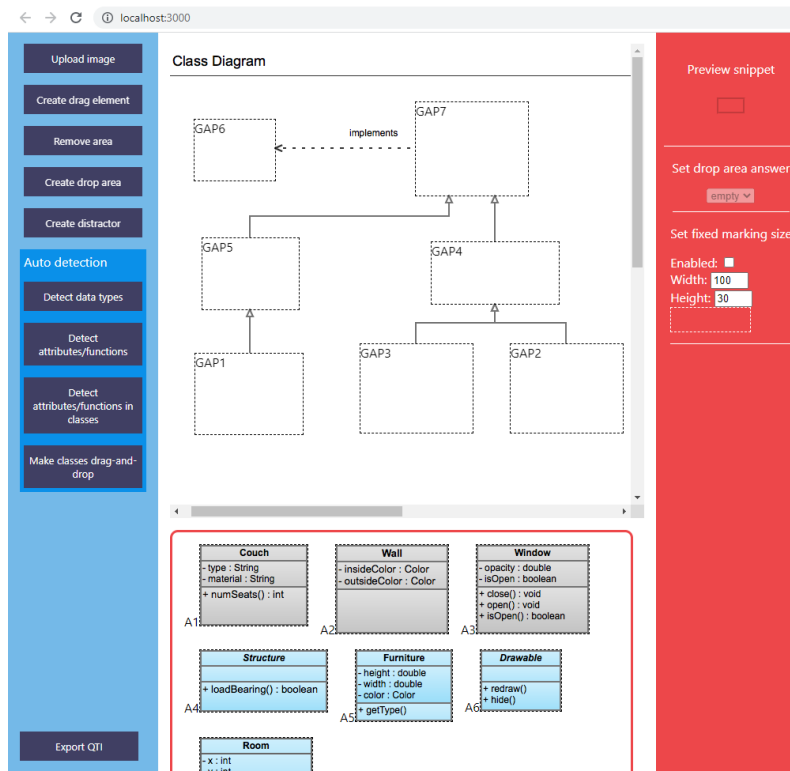


Figure 4.9: Contour detection step 5: final result

A quick note; the diagram used is only for demonstration purposes, and does not reflect a real diagram question.

Performing OCR only on content within the rectangles could be in two ways. One would be to create a separate image with each rectangle and process them with Tesseract individually. The other method was to process the whole image only once, but checking each word's position for whether or not they are within one of the rectangles. Aside from potential performance differences, a good argument for the first method was that it provides a more accurate way of detecting sentences, or rather attribute and function names combined with parameters and data types, than the current implementation. The current one checks each line's average distance between words for determining if words are connected or not. Processing each rectangle individually would make this calculation obsolete as each line in the rectangle is one full sentence/class member (provided the diagram adheres common practise). Therefore it seemed the better choice. Though, since the accuracy of detection of rectangles in a diagram image may be quite varying with different images, as they may contain variations of fonts, font size, colors, line thickness, positioning etc, it was reasonable to assume that having this implementation alongside the existing one could be desired by users. That way, if the rectangle detection algorithm performs poorly on the user's diagram and they want to detect the sentences automatically, they could still

resort to the "word spacing distance" implementation.

Testing

Testing was done using various images as in the previous iteration. Different combinations and values for the function arguments for the pre-processing methods were experimented with in order to find an acceptable level of accuracy.

Many of the same image properties considered when OCR was tested were also impactful here. Thickness of the lines in the diagrams was especially important. When too thin, the classes were simply not detected accurately. Though using a default thickness of various diagram drawing tools proved to be ok.

There was some weird behaviour when using OCR inside detected rectangles. In some cases, text that would normally be detected by OCR when performed on the whole diagram, would not be detected when performing OCR only within a class rectangle. It was for some detected after changing the rectangles background color to grey instead of white. Though, being such a rare case, further investigation was not prioritized.

Evaluation

As mentioned earlier, the OCR features implemented would in some cases target various text elements outside the class rectangles in a diagram, such as general descriptive text, association notation and in some cases recognize an arrow as a character/word. The implementation of allowing OCR to only be performed within the classes successfully mitigated this problem.

The new features were presented for the supervisor and received positive feedback. Not could it be used for making drag-and-drop creation with class diagrams easier, but potentially other diagrams as well. There are a variety of UML (and non-UML) diagrams that utilizes rectangular shapes that could benefit.

4.2 Experiment

4.2.1 Quantitative data

This section gives an overview of the data collected from the observation and the SUS questionnaire. Five different participants took part in the experiment. The data from each participant is structured in separate tables shown below, and then one table for the questionnaire results.

Observation

Participant 1:

	Inspira task 1	Inspira task 2	New tool task 1	New tool task 2
Time to complete	363	269	35	29
Actions	87	67	11	12
Accidental action errors	2	1	0	0
Failed action errors	2	1	0	0
Unnecessary action errors	2	0	0	0
System errors	1	0	0	0
Help needed	0	0	0	0

Table 4.9: Observation participant 1

Participant 2:

	Inspira task 1	Inspira task 2	New tool task 1	New tool task 2
Time to complete	514	244	35	26
Actions	133	72	11	12
Accidental action errors	1	1	0	0
Failed action errors	3	0	0	0
Unnecessary action errors	2	1	0	0
System errors	1	0	0	0
Help needed	0	0	0	0

Table 4.10: Observation participant 2

Participant 3:

	Inspira task 1	Inspira task 2	New tool task 1	New tool task 2
Time to complete	524	243	31	30
Actions	152	93	12	14
Accidental action errors	0	1	0	0
Failed action errors	3	3	0	0
Unnecessary action errors	3	2	0	1
System errors	1	0	0	0
Help needed	1	0	0	0

Table 4.11: Observation participant 3

Participant 4:

	Inspira task 1	Inspira task 2	New tool task 1	New tool task 2
Time to complete	370	316	44	42
Actions	85	73	14	12
Accidental action errors	0	0	0	0
Failed action errors	3	0	0	0
Unnecessary action errors	2	2	0	0
System errors	1	0	1	0
Help needed	1	1	0	0

Table 4.12: Observation participant 4

Participant 5:

	Inspira task 1	Inspira task 2	New tool task 1	New tool task 2
Time to complete	402	290	51	34
Actions	115	78	12	11
Accidental action errors	0	0	0	0
Failed action errors	2	3	0	0
Unnecessary action errors	2	0	0	0
System errors	1	0	0	0
Help needed	0	0	0	0

Table 4.13: Observation participant 5

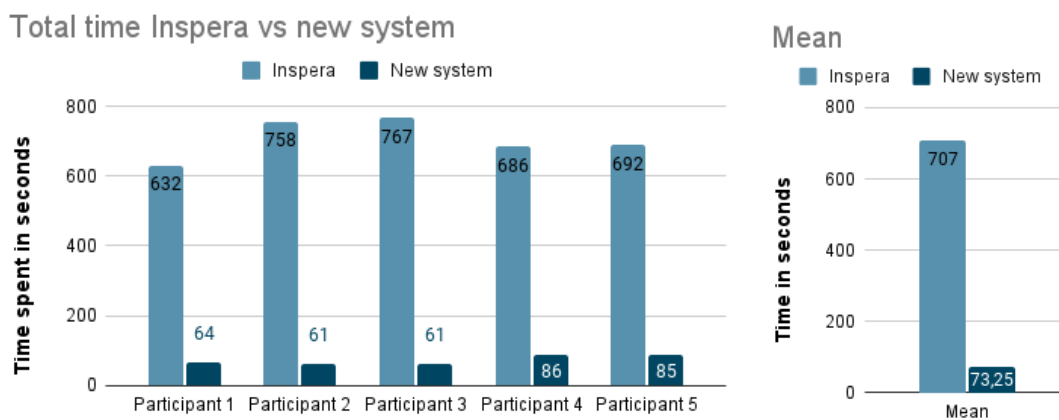


Figure 4.10: Total time and mean comparison

In the visualization above, the time spent on both task 1 and task 2 were added and compared for both tools. The mean time of all the participants was then also calculated. Looking at the chart, it showed quite a big difference between the two tools, and the data were also similar and consistent for the participants, especially for the new tool. The figure below visualizes the number of actions performed by the participants for each tool.

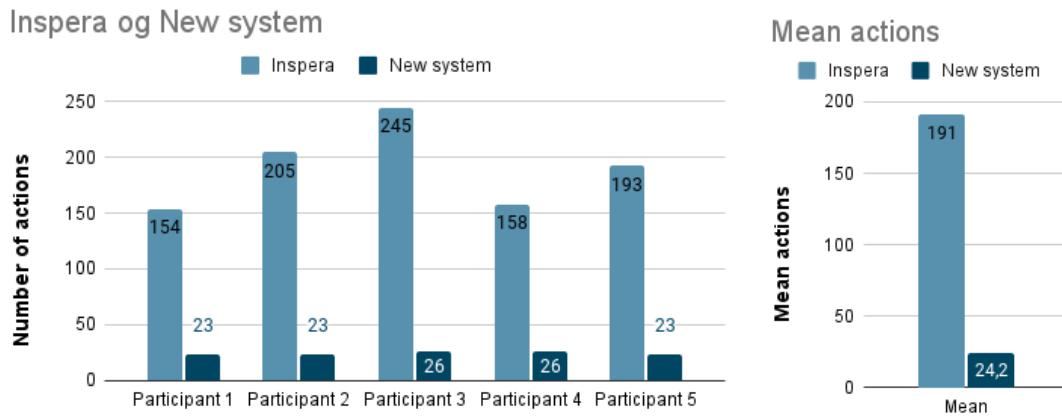


Figure 4.11: Total number of actions and mean comparison

Looking at the chart, it seems to be a similar distribution of data as with time spent, although not as severe of a difference between the tools. The charts can appear very similar, but it is important to look at the numbers as the bars are scaled differently. As both time spent and number of actions are significantly lower on the new tool compared to Inspera, it may suggest the new features implemented contributed to a faster question creation process and less work/input needed by the user to achieve the same result. It is important to note that the experiment was only conducted with five participants so this data may not be representative for a wider user base. However, with the data being so consistent between the participants, the pattern may very well appear with more user testing as well, but to draw firm conclusions a larger sample size and perhaps more extensive testing is needed. The figure below shows a visualization and comparison of the mean of the remaining data points measured.

Mean values Inspera vs new system

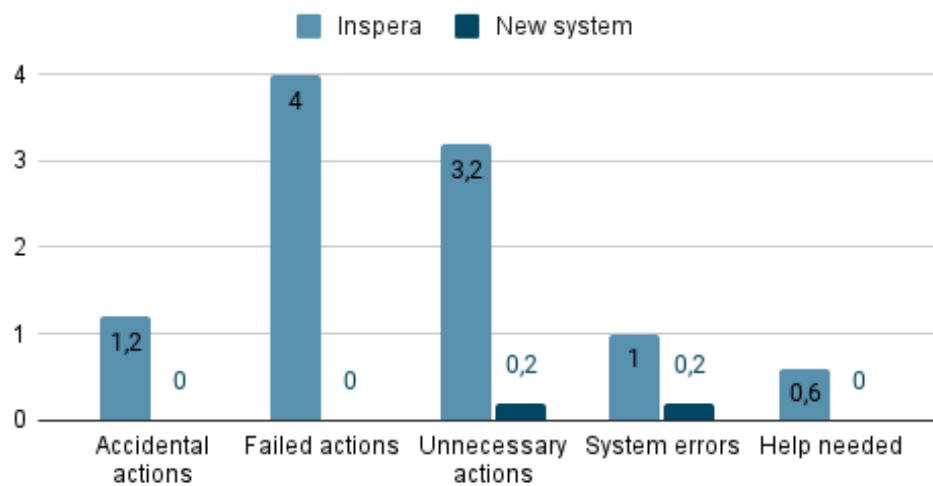


Figure 4.12: Mean values comparison

Something to note here was that the one system error that occurred in Inspera for all the participants was the same one. It was specifically related to manipulating the size of the gap elements. If it were intentional or not is unknown, but in the editor(not preview) the height of a gap element could only appear down to a fixed value and then stop shrinking. However if the user were to keep holding the mouse down and move it further in a "decreasing motion" as to make it smaller, it would in fact be set to height specified by the mouse cursor's location on release. The new set height would not appear in the editor, so the user would have to set the height and then preview the question in another browser tab to check if they are pleased with the height or not, and then possibly repeat that again until reaching a desired gap height.

As mentioned previously, the number of errors were not counted for repeated cases of the same error, i.e a user could perform the same error many times during their task and it would still count as one. This was decided in order to make the comparison more fair, as there were significantly less actions needed to perform the task in the new tool. Looking at the chart, all of the data points were higher for Inspera than the new tool which could suggest that it was easier to make user errors in Inspera. The number of times participants asked for help was low for both tools, most likely because the participants received guidance and time to practise before the actual experiment was conducted.

System Usability Scale scoring:

	SUS Inspera	SUS New tool
Participant 1	7,5	92,5
Participant 2	20	97,5
Participant 3	25	92,5
Participant 4	25	100
Participant 5	45	95

Table 4.14: SUS scores

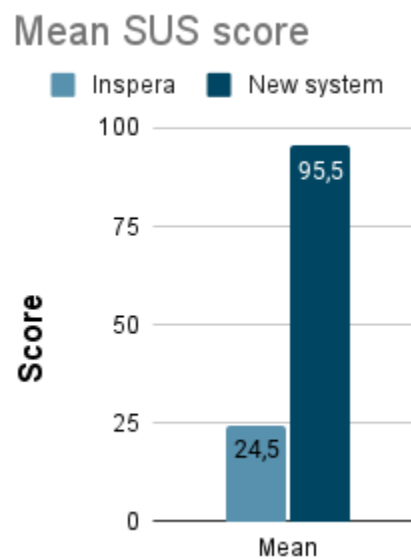


Figure 4.13: Mean SUS score comparison

The SUS questionnaire consisted of 10 questions with a scale of 1-5 as a response for each one. A total score for each for each questionnaire result were calculated using the standard method for SUS previously described in the evaluation section. However the individual responses for all of the questions can be seen in the appendix.

Looking at the table of the scores, the new tool scored higher than Inspera for all the participants. As mentioned, a SUS score of 68 can be considered average. Inspera achieved a mean score of 24,5 which is well below average, while the new tool achieved a score well above of 95,5. A SUS score for a system by itself might be difficult to interpret the meaning of, but comparing the results from two or more different cases it becomes more clear. The results here suggests that the new tool could be more usable than Inspera, but again a larger sample size would be necessary to draw conclusions from it. A paper by

Bangor et al., [3] proposed the possibility of assigning a grade and correlating adjectives such as "poor" and "good". The figure below shows this grading scale from their paper.

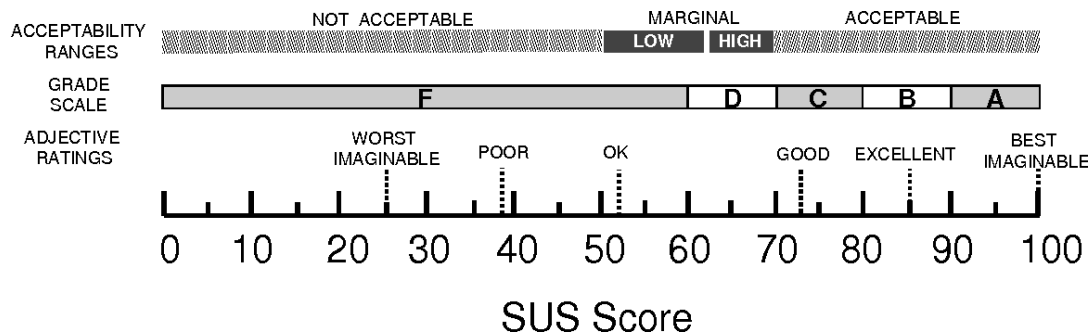


Figure 4.14: SUS grading

According to this, Inspera would receive the grade "F" and land exactly on the adjective rating of "worst imaginable" along with "not acceptable". The new tool would receive the grade "A", be between the adjective ratings of "excellent" and "best imaginable", and on the high end of the acceptable range.

Wilcoxon Signed Rank Test

A standard statistical method of comparing two groups is the t-test, either paired or unpaired. The problem is that the t-test is parametric and assumes the data being of gaussian distribution, and it is unknown whether the data collected here is. A normality test would not be meaningful with this small sample size. A paired t-test would be used here as the two groups are related (consisted the same people). The non-parametric equivalent is the Wilcoxon Signed Rank Test.

Test for time

- Null hypothesis: Time spent on tasks will be the same for both tools.
- Alternative hypothesis: Time spent on tasks in the new tool will be less than for Inspera.

	Inspira time	New tool time	Difference	Sign
Participant 3 task 2	243	30	213	+
Participant 2 task 2	244	26	218	+
Participant 1 task 2	269	29	240	+
Participant 5 task 2	290	34	256	+
Participant 4 task 2	316	42	274	+
Participant 4 task 1	370	44	326	+
Participant 1 task 1	363	35	328	+
Participant 5 task 1	402	51	351	+
Participant 2 task 1	514	35	479	+
Participant 3 task 1	524	31	493	+

Table 4.15: Wilcoxon Signed Rank Test for time

Calculating the Wilcoxon signed rank sums equals to $W_+ = 55$ and $W_- = 0$. The test statistic W equals the lowest of the two, i.e $W = 0$. Since the alternative hypothesis only states that time in the new tool is lower than in inspera, the test is one-tailed. With a number of samples $n=10$ and an alpha level $\alpha = 0.05$, gives a critical value of $W_{crit} = 10$. Because $W_{stat} < W_{crit}$, the null hypothesis was rejected.

Since the number of samples was low ($n < 20$), using normal approximation and calculation of the z-score could be unreliable, however the calculation was as follows:

$$\mu_w = \frac{n(n+1)}{4} = \frac{10(10+1)}{4} = 27,5$$

Standard deviation:

$$\sigma_w \sqrt{\frac{n(n+1)(2n+1)}{24}} = 9,81$$

Z-value:

$$z = \frac{W - \mu_w}{\sigma_w} = \frac{0 - 27,5}{9,81} = -2,8$$

Using a z-table, the p-value is 0,0026, which means that there would be a 0,26% chance that the decrease in time spent on tasks in the new tool was random.

Test for actions

Applying the Wilcoxon signed rank test for number of actions performed in each task would yield the same result. Similar null and alternative hypotheses could be stated for number of actions. As with the result of time spent, also number of actions were lower in each case, resulting in the values $W+ = 55$ and $W- = 0$. Again, since the test statistic $W_{stat}=0$ was less than the critical value $W_{crit}=10$, the null hypothesis was rejected.

Test for SUS

Because the data from SUS only contains $N=5$ pairs of scores, it is difficult to tell if there is a statistical significance within a reasonable alpha level. That said, simply looking at the results in the SUS table and the mean scores of both tools suggests that there at least could be a tendency of general perceived satisfaction among the users.

4.2.2 Qualitative data

The categorization of the interview data consisted of several steps. The first one was to note keywords for each unit of data, which would usually be a sentence although one sentence could point to multiple themes. These keywords were then used to form the initial categories. This resulted in a very long list of categories. Multiple iterations of refining them were done, combining related ones and removing some. When all of the categories had been established, they were then grouped into positive and negative categories for either of the tools in order to get a better view of the data. The number of occurrences of each category were counted and are seen in the tables below.

Positive Inspera:

Category	Frequency
More functionality	3
Better UI	2
Does not require export/import	1

Table 4.16: Positive categories Inspera

Negative Inspera:

Category	Frequency
Bad UI and UX	7
Unnecessary many steps/actions needed	5
Very tedious/cumbersome to work with	4
No aspects were better in Inspira	3
Very inconsistent	3
Requires a lot of learning to use	2
Flawed/lacking function(s)	2

Table 4.17: Negative categories Inspira

Positive new tool:

Category	Frequency
Less steps/actions needed	4
Easy/easier to use	3
Saves time / is fast	3
Alot of things automated	3
More accurate gap creation	2
Easy to learn/understand	1
Does not require writing any text for gaps	1

Table 4.18: Positive categories new tool

Negative new tool:

Category	Frequency
Bad UI	5
Could include more automation/AIG	3
Could include automatic import to Inspira	2
Could include explanation/tooltip for buttons	2
Need to use two systems	1

Table 4.19: Negative categories new tool

The tables show how many times a theme were brought up, however the importance of each one could vary a lot. For example when the participants were asked if there were

anything they disliked about the new tool, many of them were reluctant to answer at first or had to think for a moment. In contrast, when asked about Inspira, some of the negative response were given quickly and decisively.

Chapter 5

Discussion

This section discusses the process and results gathered for both the Design and creation strategy and the experiment strategy, with respect to the research questions.

5.1 Design and creation

This research strategy focused on the first research question:

- **RQ1:** *How can an IT tool be designed to be more efficient in creating automatically assessed diagram questions for Inspera?*

Throughout the development process of the tool and planning phase, there was a lot of uncertainty of what a final result would look like. Initially there were little knowledge of what was possible and what was not. The knowledge gained along the way influenced the scope a great deal and new ideas emerged between iterations. Several ideas seemed promising at first, however upon further examination and background work, could not be pursued either because of the limited time frame or simply were not viable.

After analysing how Inspera worked and the process of creating drag-and-drop questions with it, there were found many aspects to it that had potential for improvements. The improvement of the manual process was therefore the main focus initially. It was thought of as a good starting point, as it was not bound to a specific type of QTI question, for example making data types in a diagram drag-and-drop. Instead, improvements to the manual process would be useful for any type of drag-and-drop question within any domain.

5.1.1 Focus

Further in the process, it became apparent that specific ideas for improvement or automation was bound to specific cases of question types. Previous exams and syllabus for various IT subjects, along with insight from this thesis' supervisor and another professor, were used to find out what types of automatically assessed questions could be desirable to create. Various ideas came up, however they would mostly only help with one specific question type. As it was not possible to implement everything, priorities needed to be made.

Automated item generation, i.e generating many unique questions from little amount of work, was one of the first concepts explored. Implementing AIG can be done quite easily in the context of math problems, as one are dealing with numbers. With diagrams, it proved to be more challenging. An attempt was made to do it with diagrams, specifically circuit diagrams as they can involve numbers and math. There were many different ways to accomplish AIG, for example questions consisting of a generated boolean expression and the student must "draw" or interact with a diagram to meet the specific circuit condition. Different logic gates could be used as drag elements to design a correct diagram. Another example could be having a full circuit diagram automatically generated from a boolean expression, and random logic gates could be turned into drag-and-drop elements. The former example was pursued, however because of various issues that arose underway, it could not be done as easy as initially thought, and another approach was needed. At that time, implementing some form of image recognition seemed more viable given the time frame, and AIG was therefore given lower priority and did not make it into the implementation phase.

In order to develop a tool with some substance, it was decided to narrow down the scope a little bit and mainly focus on UML class diagrams. In terms of effectiveness and satisfaction of usability, aspects that caused an unnecessary amount of actions needed and frustration among users of Inspera were thought to be improved in the new tool after the first iteration focusing on the manual process. More focus was then set on improving efficiency, i.e the time it takes to create a question, which lead to the use of image recognition.

Using an iterative development process seemed to be a generally good choice and very fitting with the research strategy. Initially, the understanding of the problem was somewhat limited, and was not eminently clear how to approach it the best way. As mentioned early in the thesis, the Design and Creation strategy consisted of the five steps awareness,

suggestion, development, evaluation, and conclusion, performed in an iterative cycle. The awareness at the start was lacking, even with research done prior to starting development. With the knowledge available, suggestions were made on how to design a tool with better usability than Inpera Assessment, which continued to developing and evaluating them. After one cycle, more awareness or knowledge was gained of the problem which lead to more ideas not thought of earlier, and this continued in more cycles. As mentioned, some ideas might have failed, but the knowledge gained from attempting it were still valuable. Compared to utilizing something like the waterfall method, this most likely resulted in a better answer to the research question.

5.1.2 Expanding

Many aspects in the new tool could have been better if it had more focus. The user interface for example was very little prioritised. UI can affect user satisfaction and the other usability metrics, so some work was put into it. However the important considerations were that it needed to be understandable and easy to use, but not necessarily elegant. As seen in data from the experiment and discussed more in the next section, the category "Bad UI" was the one with the highest frequency from the interviews, however all participants scored much higher on the System Usability Scale, which could suggest that the UI was good enough for the purpose.

Contour detection is also something that could have been expanded upon. The resulting new tool had the feature of detecting rectangles with the purpose of creating drag-and-drop elements from classes in a class diagram. It could have been expanded to be used for diagrams and figures as well. For example it could detect and highlight all types of contours (or a specific one defined by the user) for the user and allow them to choose which object they want to turn into drag-and-drop with a mouse click, similar to the word selection feature implemented using OCR. For example, oval contours could be targeted for use-case diagrams, or rounded rectangles for BPMN process models.

The OCR features also has limitations. With the word detection and selection option the user can click on any word in the diagram to automatically make it into drag-and-drop with the correct position and dimensions. However, only the single word clicked on is used and there is no option for the user to for example have the word clicked on as well as the x number of words next to it selected and made drag-and-drop. An improvement to this could be either the user selects that they want x number of words selected with one click, or they use the mouse selection tool in the manual process to mark a text area to be

selected. The other OCR are limited in that they specifically only target class diagrams, although similar diagrams could also work. This could be expanded with more such specific features for a wider range of diagram types.

There was also an idea of implementing a diagram drawing tool inside the tool to eliminate the need of using a third party application outside of it, although it could potentially be a lengthy undertaking. In addition, teachers and professors often times use a dedicated modeling tool intended for the the diagram language they use, offering more features than only the drawing part.

5.2 Experiment

This section consists of a discussion of all the results of the experiment. The experiment strategy focused on the second research question:

- **RQ2:** *To what extent can the IT tool in RQ1 be used as a replacement to Inspira's authoring tool?*

The formed hypothesis from the experiment were stated as follows:

- **H:** *The new IT tool is more usable in creating drag-and-drop QTI questions of diagrams than Inspira Assessment.*

5.2.1 Quantitative data

The quantitative data collected consisted of the data points time, number of actions, number of the three types of user errors, number of system errors, and number of times asked for help, measured during the observation. It also consists the calculated SUS score of each participant. Each data point is discussed in order.

There are a few general things to note about the results. First of all, with such a small sample size of five, one can not really draw conclusions from the results. Secondly, even though it has been attempted to mitigate as much as possible, there could be external factors influencing the results. Prior experience with using Inspira is one already mentioned, but probably not relevant as each participant was given guidance on how each tool worked, and only performed the task after becoming comfortable enough with them. The

scale of the whole experiment process could affect how the participant behaved. Although the estimated time to complete the experiment per participant was around 50 minutes, it varied quite a lot between each one. For some, it took around 35 minutes, while for others it could take up to 1,5 hours. When the experiment went on for some time exceeding the estimation, participants could have gotten impatient and frustrated, having other matters in their lives to attend to, or just tired wanting it to be over with. This could result in an inferior performance of the tasks, i.e making more errors, not caring of getting the resulting question completely correct etc. It could also lead to them not wanting to take the enough time to learn either of the tools before performing the task, leading to more guessing.

Time

The first point is the time each participant spend on completing their task. The time started when the tool used was already set up and prepared. For Inspera, that would be in the question editor with a blank question, where navigating up until that point was excluded. For the new tool, that would simply be on the front web page. The time stopped for Inspera when the user had saved their final version of the question, and for the new tool when they had successfully imported it into Inspera. As seen in the results, time spent in Inspera was significantly more than in the new tool. Time in the new tool was around 10-12% of Inspera, consistently among all the participants. From the Wilcoxon signed rank test, there was a statistical significance for time in the new tool being less than in Inspera.

No firm conclusion can be drawn from these results as there were only five participants. However with approximately the same result for each of them and with such a large difference between the two tools, it is quite likely the new tool does in fact require less time for a user to create a question of the same type. Regarding usability, higher efficiency is likely, even though the exact difference could be lower with higher sample size.

Actions

An action was defined as one particular interaction causing some sort of event in the system, for example clicking, navigating, moving a gap element, changing the size of a gap element, and writing text. Though, one single click would not always count as one action, for example double clicking an image to upload it was only one action. The results here, as with time, showed that the number of actions were significantly less in the new tool compared to Inspera, being around 10-16% of Insperas values. The Wilcoxon signed

rank test had the same result as with time.

All types of actions were counted, including erroneous ones. If the user performed some steps of the question creation process in an sub-optimal way like in the wrong order, it could result it more actions needed. The guidance part for each participant before conducting the experiment was used in order for them to perform at a level similar to someone who would use the tools regularly and be familiar with them, but there is only so much one can learn in a short time, and only so much the participant would be willing to learn just for the sake of the experiment. Even if the participant learned how the UI worked and what buttons to click, does not necessarily mean they would be completely comfortable with it and perform actions effortlessly without much thinking. Especially for Inspera, there was quite a few quirks about the tool of how it behaved that were not very obvious, for example getting the correct menu to appear by clicking in particular but seemingly random spots in the UI. Such small details can be hard to remember especially when there are several of them. For someone who uses Inspera regularly would most likely be aware of all of them, however for someone who just learned the tool it might be easy to forget since it is so intuitive and hard to just go along with without being used to it. This could lead to them performing many extra actions out of guesswork, and also time and eventually the need for asking for help from the observer.

Even though one can not conclude that the number of performed actions is lower in the new tool than in Inspera, as with time spent, the results show a tendency of it, and with such a big difference between the two, it is a likely case. It is hard to say how much of an impact the external factors could have, but by looking at the task description for each tool, performing them in an optimal way would likely lead to less actions for the new tool, as there are less steps involved. The number of actions is one of the measurements for effectiveness, but could also have a great impact on satisfaction, as needing to perform many actions can be experienced as tedious, especially if they appear to be unnecessary and there are obvious ways the tool could be designed differently to prevent them from the user's perspective.

Errors and help

The number of errors were separated into four categories. As mentioned, each specific error was only counted once in order for the comparison to be more fair. The system errors are not that relevant to look at. The one same system error that occurred for all participants in Inspera could be seen as simply bad UI depending on definition. The system error that occurred for participant 4 in the new tool was that the user appeared to click on "Export

QTI" and nothing happened, and they had to click again. This case was recorded over Teams, and therefore the recording of their screen was of poor quality making it hard to determine exactly what happened at that particular action. The user could have simply moved their mouse too fast/inaccurate, missing the click or failed to click with their mouse button, but it was noted as a system error either way.

The three categories of user errors are perhaps a bit more relevant. Most of the participants had unnecessary action errors for one or both of their tasks in Inspira. These types of errors were mostly moving gap elements around or adjusting their size without it being necessary. This could be for a number of reasons. It could perhaps point to a unintuitive UI, or again, some of the external factors already mentioned such as experience with the tool. Failed action errors were errors where the user attempted to perform a deliberate action, but failed. This could for example be failing to have the correct gap size or position in the final question, or failing to use/find the correct menu or a particular feature. Accidental action errors were errors where the user performed an action by accident. Looking at the data for each of these three user errors separately might not give that much insight, as some errors could be appropriate to categorize as any of them depending on how one looks at it.

The case of asking for help only occurred for two participants, and only in Inspira. With so little data, it is difficult to say anything about it, as it could simply be random. It was also expected to be low or zero because of the guidance given prior to the experiment. With the errors, there was a general trend of more errors occurring in Inspira compared to the new tool, which could suggest there is higher effectiveness in the new tool. The fact that each user on average did more errors with Inspira than with the new tool, could be due to the fact that many more actions were needed to complete a task in Inspira than in the new tool. If each action performed had a probability of causing an error, there would naturally be more errors with more actions.

System Usability Scale

The SUS score is a measurement of usability, but perhaps measuring more towards the satisfaction aspect, as it is the users themselves who provides the data. The results show that all the participants scored higher for the new tool than for Inspira. With the mean value of both tools, using the grading scale in figure 4.14, Inspira was graded "F" while the new tool was graded "A". The highest single score for Inspira was from participant 5, with a score of 45. Although almost double the mean score, it would still fall under grade "F" and be within "Not acceptable", but be between the adjective ratings of "poor"

and "ok".

The experiment was structured with Inspera first and then the new tool. The participants learned using Inspera, performed the tasks in Inspera, and filled out the SUS questionnaire for Inspera with no knowledge of the new tool. This was a deliberate choice so that in case the new tool was in fact better in terms of usability, they would not simply rate Inspera terrible if the experiment would start with the new tool, and then they would supposedly use a worse tool (Inspera) afterwards. With that in mind, Inspera still received a very low score even when the participants had no frame of reference of a similar tool.

The argument could be made in the other direction as well. Because the experiment ended with the supposedly improved new tool, even just minor improvements could encourage the user to inflate their opinion of it unnaturally high, which could be a reason for the score being so close to 100.

As stated earlier, the individual scores do not say much by themselves, and the hypothesis was only that the new tool is more usable, and not *how* much more usable. Even if the SUS score for the new tool might be unnaturally high, improvement is still improvement, therefore the result could suggest that satisfaction and perceived usability is likely higher for the new tool.

5.2.2 Qualitative data

This section discussed the categorized results from the interview data. The interview was conducted in the end of the experiment after the participant had completed the tasks in both tools and answered the SUS questionnaires. The interview consisted of five questions. The goal was to gain more insight of the new tool such as good and bad aspects, and general thoughts of either of the two tools. A lot of the result are due to the specific questions asked, however the final question was more open-ended allowing the user to be free to say anything.

One thing the categories and frequencies does not reflect is how strongly the participant feel about the specific category. The negative categories for the new tool are not necessarily really negative, but could be suggestions of further development. With the questions of "*What other features would you want in the new system?*" is where most of those responses came from. The participants might have felt forced to say something rather than nothing. "*Bad UI*" was the category for the new tool with the highest frequency. Many of the responses for this was that the UI worked fine and easy to understand, but did not

look elegant or professional. Some provided tips specific improvements such as centering the whole UI in the browser window, changing colors, changing certain element positions or sizes etc. This feedback along with the other suggestions of implementing more automation, automatic import to Inopera, and explanation for buttons could be valuable for further development.

The positive categories for the new tool were more or less expected and reflects the data from the observation and SUS questionnaire, where the highest frequency category was "*Less steps/actions needed*".

For Inopera, there was only three positive categories, the highest being "*More functionality*". This mostly came from the question "*Are some aspects better in Inopera than the new system?*". Looking at the specific responses, it seemed to only be a guess or assumption from the participant, as they did not actually use these other functions. The "*Better UI*" category had two occurrence, however looking at the negative categories for Inopera, "*Bad UI and UX*" had seven, which could mean that Inopera's UI was also bad, but maybe not as bad as the new tool. That particular category specified UI and UX, as many of the responses mentioned themes relating to both.

The second negative category for Inopera was "*Unnecessary many steps/actions needed*", with a frequency of five. Specifically, the complaints consisted of unnecessary popup boxes causing extra work for no reason, wrong appearance of the gap elements making the user need to preview the question in a different browser tab repeatedly, the need of clicking around to get the correct menu to appear, and others.

Generally, the participants were more positive towards the new tool than Inopera, which reflects how they answered the SUS questionnaire. The new tool did receive negative categories that were brought up by multiple participants, however looking at the exact responses, were intended to be more constructive in nature. In terms of satisfaction and general usability, there is a trend in favor of the new tool.

Chapter 6

Conclusion and future work

6.1 Conclusion

Creating questions for digital testing that can be automatically assessed is very useful for both summative and formative tests. In order to create such questions that can cover the syllabus, the correct tools need to exist and be designed well. Because of the current limitations of the existing tools, specifically when it comes to drag-and-drop diagram question creation, it would be interesting to explore how this could be improved and develop a prototype for a new tool.

Research question 1: *How can an IT tool be designed to be more efficient in creating automatically assessed diagram questions for Inspera?*

During this thesis, the researcher has analysed the process of creating automatically assessed diagram questions in various existing tools, most of all Inspera. With a better understanding of how these tools work, several ideas for improvements have been explored and suggested. An IT artefact was then developed which implemented some of the improvements with regards to usability, using the three metrics effectiveness, efficiency and satisfaction. More discoveries and knowledge was gained during the development of how the tool could be designed and improved. In order to evaluate the developed IT artefact, an experiment was designed and conducted.

Research question 2: *To what extent can the IT tool in RQ1 be used as a replacement to Inspera's authoring tool?*

Derived from RQ2, was a hypothesis of that the new IT tool would be more usable than

Inspira. To test this, the experiment strategy was used with the three data generation methods observation, questionnaire and interview. The researcher compared the data gathered from a group of participants performing equivalent tasks in both tools and their perceived experience with them, with respects to each of the three metrics of usability. The results from each data generation method all pointed in positive favor of the new tool. All of the participants had similar results, and there was not really any outliers. The time spent on completing a task (efficiency) in the new tool was around 1/10 of Inspira. Number of actions, errors and times asked for help (effectiveness) were also all lower for the new tool. The mean SUS scores (satisfaction) were 24,5 and 95,5 for Inspira and the new tool respectively. The qualitative data from the interviews also indicated that the process was improved with the new tool.

With a sample size of only five participants, it is not certain whether these results holds true for the general user base in a real natural setting, and more extensive testing would be needed to conclude.

6.2 Future work

There are many possibilities for future work, both further development of the new tool as well as more research.

6.2.1 Development

One improvement to the tool that became obvious from the experiment is a better UI design, as well as including explanations for each feature inside it. Some other possibilities already mentioned in the discussion section is better text selection when using OCR features, and more shapes other than rectangles for the contour detection, possibly letting the user specify the contour themselves.

The accuracy of the image recognition features is also very likely to be improvable. For the purposes of the IT artefact, the testing and the effort that could be spent on these specific features were limited. A more systematic and extensive testing process, tweaking the various detection parameters, would probably yield better results. It would also be beneficial to map out which specifications of the diagram image affect the accuracy of detection the most. Properties such as image dimensions, colors, line thickness, font type, font size etc can have a great impact, and properly adjusting the detection parameters to

specifically perform better with diagrams could be an idea.

The tool could be developed to support more types of diagrams, although any kind of image could be used with the manual process. Currently it is lacking many of Insperas features, as Inspera is a big system with support for many QTI question types. There are possibilities for implementing more of those. Automatic upload to Inspera was mentioned and brought up by some during the interview, which could be implemented as well. AIG was also discussed quite a bit, but was not implemented due to limited time. This could be pursued further.

6.2.2 Research

As nothing can be concluded with only a sample size of five, more experimentation could be done with the new tool with a larger sample. An extended interview could also be used for gathering more thoughts and ideas from the users perspective. More of the features of the new tool could also be tested and evaluated, as only a subset was used in the experiment conducted here.

The findings from this thesis could prove useful for development and research of similar tools to the one developed here. The big Learning Management Systems such as Blackboard, OpenOLAT, Moodle etc, and other e-assessment systems like Inspera Assessment could potentially explore and use some of the suggestions discussed here, and perhaps implement them themselves, which could be beneficial a number of schools and universities, and encourage more use of automatically assessed questions in tests.

Bibliography

- [1] ISO 9241-11:2018(en). ‘Ergonomics of human-system interaction’. In: International Organization for Standardization standard 11 (2018).
- [2] David Bañeres et al. ‘Experiences in Digital Circuit Design Courses: A Self-Study Platform for Learning Support’. In: IEEE Transactions on Learning Technologies 7.4 (2014), pp. 360–374. DOI: 10.1109/TLT.2014.2320919.
- [3] Aaron Bangor, Philip T. Kortum and James T. Miller. ‘Determining what individual SUS scores mean: adding an adjective rating scale’. In: Journal of Usability Studies archive 4 (2009), pp. 114–123.
- [4] Alice Barana, Marina Marchisio and Matteo Sacchet. ‘Advantages of Using Automatic Formative Assessment for Learning Mathematics’. In: July 2019, pp. 180–198. ISBN: 978-3-030-25263-2. DOI: 10.1007/978-3-030-25264-9_12.
- [5] Blackboard. Blackboard. URL: <https://www.blackboard.com/>. (accessed: 06.12.2021).
- [6] Alina Bockshecker, Katharina Ebner and Stefan Smolnik. ‘Technology-Enhanced Learning Environments and Adaptive Learning Systems–Development of Functionality Taxonomies’. In: (2022).
- [7] Aparna Chirumamilla and Guttorm Sindre. ‘E-Assessment in Programming Courses: Towards a Digital Ecosystem Supporting Diverse Needs?’ In: Digital Transformation for a Sustainable Society in the 21st Century. Ed. by Ilias O. Pappas et al. Cham: Springer International Publishing, 2019, pp. 585–596.
- [8] IMS Global Learning Consortium. IMS Question Test Interoperability Implementation Guide. URL: https://www.imsglobal.org/question/qtiv2p1/imsqti_implv2p1.html. (accessed: 12.06.2022).

-
- [9] IMS Global Learning Consortium. IMS Question Test Interoperability Overview. URL: https://www.imsglobal.org/question/qtiv2p1/imsqti_oviewv2p1.html. (accessed: 06.12.2021).
- [10] IMS Global Learning Consortium. QTI assessmentItem example. URL: <https://www.imsglobal.org/question/qtiv2p1/examples/items/choice.xml>. (accessed: 06.12.2021).
- [11] Muriel Foulonneau. ‘Generating Educational Assessment Items from Linked Open Data: The Case of DBpedia’. In: The Semantic Web: ESWC 2011 Workshops. Ed. by Raúl García-Castro, Dieter Fensel and Grigoris Antoniou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 16–27.
- [12] Mikulas Gangur. ‘Automatic Generation of Cloze Questions.’ In: (Jan. 2011), pp. 264–269.
- [13] Ryan M Gibson and Gordon Morison. ‘Improving Student Engagement and Active Learning with Embedded Automated Self-assessment Quizzes: Case Study in Computer System Architecture Design’. In: Intelligent Computing. Springer, 2021, pp. 327–339.
- [14] Xin-Yi Gong et al. ‘An overview of contour detection approaches’. In: International Journal of Automation and Computing 15.6 (2018), pp. 656–672.
- [15] Aldo Gordillo. ‘Effect of an Instructor-Centered Tool for Automatic Assessment of Programming Assignments on Students’ Perceptions and Performance’. In: Sustainability 11.20 (2019). ISSN: 2071-1050. DOI: 10.3390/su11205568. URL: <https://www.mdpi.com/2071-1050/11/20/5568>.
- [16] Mary Sarah-Jane Gregory and Jason Michael Lodge. ‘Academic workload: the silent barrier to the implementation of technology-enhanced learning strategies in higher education’. In: Distance education 36.2 (2015), pp. 210–230.
- [17] Marcus Joar Hauge. ‘Master in Informatics, Preparatory Project’. In: NTNU Department of Computer Science (2021).
- [18] Rose Holley. ‘How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs’. In: D-Lib Magazine 15.3/4 (2009).
- [19] Inspera. Inspera. URL: <https://www.inspera.com/>. (accessed: 06.12.2021).
- [20] Inspera. Question types - Automatically marked. URL: <https://support.inspera.com/hc/en-us/sections/4564122809245-Question-types-Automatically-marked>. (accessed: 12.06.2022).
-

-
- [21] Joachim William Hegvold Jørgensen and Simon Kvannli. 'Efficient Generation of Parsons Problems for Digital Programming Exams in Inspera'. In: NTNU Department of Computer Science (2019). URL: <https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2624513>.
- [22] James R Lewis and Jeff Sauro. 'Item benchmarks for the system usability scale.' In: Journal of Usability Studies 13.3 (2018).
- [23] OpenOLAT LMS. OpenOLAT - infinite learning. URL: <https://www.openolat.com/>. (accessed: 06.12.2021).
- [24] Lvivity. Single-page App vs. Multi-page App: Pros, Cons, and Which is Better? URL: <https://lvivity.com/single-page-app-vs-multi-page-app>. (accessed: 14.12.2021).
- [25] Moodle. Moodle. URL: <https://moodle.org/>. (accessed: 06.12.2021).
- [26] Project Naptha. Tesseract.js, Javascript port of Tesseract. URL: <https://github.com/naptha/tesseract.js>. (accessed: 08.12.2021).
- [27] Jakob Nielsen. 'Enhancing the Explanatory Power of Usability Heuristics'. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '94. Boston, Massachusetts, USA: Association for Computing Machinery, 1994, pp. 152–158. ISBN: 0897916506. DOI: 10.1145/191666.191729. URL: <https://doi.org/10.1145/191666.191729>.
- [28] Don Norman. The design of everyday things: Revised and expanded edition. Basic books, 2013.
- [29] Briony J Oates, Marie Griffiths and Rachel McLean. Researching information systems and computing. Sage, 2006.
- [30] OMG. UML. URL: <https://www.omg.org/UML/>. (accessed: 06.12.2021).
- [31] Onyx. Onyx Editor. URL: <https://www.onyx-editor.com/>. (accessed: 06.12.2021).
- [32] OpenCV. OpenCV. URL: <https://opencv.org/>. (accessed: 12.06.2022).
- [33] OpenCV. OpenCV Modules. URL: <https://opencv.org/>. (accessed: 12.06.2022).
- [34] Ecaterina Pacurar, Philippe Trigano and Sorin Alupoae. 'A QTI editor integrated into the netUniversité web portal using IMS LD'. In: Journal of Interactive Media in Education 2005 (Aug. 2005). DOI: 10.5334/2005-9.
-

-
- [35] Chirag Patel, Atul Patel and Dharmendra Patel. ‘Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study’. In: International Journal of Computer Applications 55 (Oct. 2012), pp. 50–56. DOI: 10.5120/8794-2784.
- [36] W3C Process. Web Content Accessibility Guidelines (WCAG) international standard. URL: <https://www.w3.org/WAI/standards-guidelines/wcag/>. (accessed: 12.06.2022).
- [37] Neil Salkind. Encyclopedia of Research Design. pages 1639-1644. Thousand Oaks, May 2010. DOI: 10.4135/9781412961288. URL: <https://doi.org/10.4135/9781412961288>.
- [38] Karen Scouller. ‘The influence of assessment method on students’ learning approaches: Multiple choice question examination versus assignment essay’. In: Higher Education 35.4 (1998), pp. 453–472.
- [39] sirfz. Tesseractocr, Tesseract wrapper for python. URL: <https://github.com/sirfz/tesseractocr>. (accessed: 08.12.2021).
- [40] R. Smith. ‘An Overview of the Tesseract OCR Engine’. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). Vol. 2. 2007, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991.
- [41] Jeff Sutherland and Ken Schwaber. The 2020 Scrum Guide. URL: <https://scrumguides.org/scrum-guide.html>. (accessed: 06.12.2021).
- [42] TAO. TAO Testing. URL: <https://www.taotesting.com/>. (accessed: 06.12.2021).
- [43] Maddalena Taras. ‘Assessment–summative and formative–some theoretical reflections’. In: British journal of educational studies 53.4 (2005), pp. 466–478.
- [44] Tesseract. Tesseract-ocr. URL: <https://github.com/tesseract-ocr/tesseract>. (accessed: 06.12.2021).
- [45] Guobo Xie and Wen Lu. ‘Image edge detection based on opencv’. In: International Journal of Electronics and Electrical Engineering 1.2 (2013), pp. 104–106.

Appendix

Appendix A

Requirements and tasks

A.1 Non-functional requirements

ID	Requirement description
1	The system should be supported for Windows, MacOS, and Linux(Ubuntu).
2	The system should be easy to use and understand.
3	The system should be more usable than Inspira for creating drag-and-drop diagram questions.
4	Every interaction the user does with the system should be immediate.
5	The created questions should securely handled, and unauthorized access to view them should be prevented.

A.2 Functional requirements

ID	Requirement description
1	The system should have an understandable GUI
2	The user should be able to upload image files and view them
3	The user should be able to create drag-and-drop elements
4	The user Should be able to erase/cut parts of the image
5	The user Should be able to create distractor elements
6	The user Should be able to modify a drop area's correct answer
7	The user Should be able to erase/cut parts of the image
8	The system should be able to export QTI compliant files recognisable by Inspira
9	The system should be able to automatically detect text in the uploaded image and select relevant words/sentences
10	The system should make automatically detected text elements into drag-and-drop elements
11	The user should be able to click on words in the image to make them into drag-and-drop elements
12	The system should be able to automatically detect classes in a class diagram

A.3 Tasks

ID	Task description
1	Create a simple GUI
2	Implement area selection tool with a preview area
3	Implement the various user GUI actions
4	Implement XML parsing for generating the QTI XML files
5	Install and integrate the Tesseract.js OCR library
6	Define and implement the rules of what text should be targeted for automatic detection
7	Ensure acceptable accuracy of text recognition with Tesseract.js
8	Update the GUI to make it evident that words in the image can be clicked after performing OCR
9	Implement drag-and-drop element creation on word click
10	Set up and initialize a Flask server
11	Update the GUI to include option for class detection
12	Create a route in Flask and create the appropriate http request in the frontend
13	Implement contour detection with OpenCV in Flask
14	Create option of performing OCR only within detected contours

Appendix B

System Usability Scale

B.1 Questionnaire form

SUS questionnaire:

<https://forms.gle/1sy5FBGuHkFVKj2P8>

System Usability Scale for Inspera

[Logg på Google](#) for å lagre fremdriften din. [Finn ut mer](#)

*Må fylles ut

I think that I would like to use this system frequently *

1 2 3 4 5

Strongly disagree Strongly agree

I found the system unnecessarily complex *

1 2 3 4 5

Strongly disagree Strongly agree

I thought the system was easy to use *

1 2 3 4 5

Strongly disagree Strongly agree

Figure B.1: SUS questionnaire q1-q3

I think that I would need the support of a technical person to be able to use this *
system

1 2 3 4 5

Strongly disagree Strongly agree

I found the various functions in this system were well integrated *

1 2 3 4 5

Strongly disagree Strongly agree

I thought there was too much inconsistency in this system *

1 2 3 4 5

Strongly disagree Strongly agree

I would imagine that most people would learn to use this system very quickly *

1 2 3 4 5

Strongly disagree Strongly agree

Figure B.2: SUS questionnaire q4-q7

I found the system very cumbersome to use *

1 2 3 4 5

Strongly disagree Strongly agree

I felt very confident using the system *

1 2 3 4 5

Strongly disagree Strongly agree

I needed to learn a lot of things before I could get going with this system *

1 2 3 4 5

Strongly disagree Strongly agree

Send Tøm skjemaet

Dette innholdet er ikke laget eller godkjent av Google. [Rapportér uriktig bruk](#) - [Vilkår for bruk](#) - [Retningslinjer for personvern](#)

Google Skjemaer

Figure B.3: SUS questionnaire q8-q10

B.2 SUS responses

B.2.1 Inspera

I think that I would like to use this system frequently	1
I found the system unnecessarily complex	5
I thought the system was easy to use	2
I think that I would need the support of a technical person to be able to use this system	4
I found the various functions in this system were well integrated	1
I thought there was too much inconsistency in this system	5
I would imagine that most people would learn to use this system very quickly	1
I found the system very cumbersome to use	5
I felt very confident using the system	2
I needed to learn a lot of things before I could get going with this system	5

Figure B.4: SUS Inspera participant 1

I think that I would like to use this system frequently	1
I found the system unnecessarily complex	3
I thought the system was easy to use	1
I think that I would need the support of a technical person to be able to use this system	4
I found the various functions in this system were well integrated	1
I thought there was too much inconsistency in this system	3
I would imagine that most people would learn to use this system very quickly	1
I found the system very cumbersome to use	4
I felt very confident using the system	2
I needed to learn a lot of things before I could get going with this system	4

Figure B.5: SUS Inspera participant 2

I think that I would like to use this system frequently	1
I found the system unnecessarily complex	5
I thought the system was easy to use	1
I think that I would need the support of a technical person to be able to use this system	3
I found the various functions in this system were well integrated	2
I thought there was too much inconsistency in this system	4
I would imagine that most people would learn to use this system very quickly	2
I found the system very cumbersome to use	4
I felt very confident using the system	3
I needed to learn a lot of things before I could get going with this system	3

Figure B.6: SUS Inspera participant 3

I think that I would like to use this system frequently	2
I found the system unnecessarily complex	4
I thought the system was easy to use	1
I think that I would need the support of a technical person to be able to use this system	3
I found the various functions in this system were well integrated	2
I thought there was too much inconsistency in this system	3
I would imagine that most people would learn to use this system very quickly	2
I found the system very cumbersome to use	5
I felt very confident using the system	2
I needed to learn a lot of things before I could get going with this system	4

Figure B.7: SUS Inpera participant 4

I think that I would like to use this system frequently	2
I found the system unnecessarily complex	2
I thought the system was easy to use	3
I think that I would need the support of a technical person to be able to use this system	2
I found the various functions in this system were well integrated	3
I thought there was too much inconsistency in this system	3
I would imagine that most people would learn to use this system very quickly	2
I found the system very cumbersome to use	4
I felt very confident using the system	2
I needed to learn a lot of things before I could get going with this system	3

Figure B.8: SUS Inpera participant 5

B.2.2 New tool

I think that I would like to use this system frequently	4
I found the system unnecessarily complex	1
I thought the system was easy to use	5
I think that I would need the support of a technical person to be able to use this system	1
I found the various functions in this system were well integrated	4
I thought there was too much inconsistency in this system	1
I would imagine that most people would learn to use this system very quickly	5
I found the system very cumbersome to use	1
I felt very confident using the system	4
I needed to learn a lot of things before I could get going with this system	1

Figure B.9: SUS new tool participant 1

I think that I would like to use this system frequently	5
I found the system unnecessarily complex	1
I thought the system was easy to use	5
I think that I would need the support of a technical person to be able to use this system	2
I found the various functions in this system were well integrated	5
I thought there was too much inconsistency in this system	1
I would imagine that most people would learn to use this system very quickly	5
I found the system very cumbersome to use	1
I felt very confident using the system	5
I needed to learn a lot of things before I could get going with this system	1

Figure B.10: SUS new tool participant 2

I think that I would like to use this system frequently	5
I found the system unnecessarily complex	1
I thought the system was easy to use	5
I think that I would need the support of a technical person to be able to use this system	2
I found the various functions in this system were well integrated	5
I thought there was too much inconsistency in this system	1
I would imagine that most people would learn to use this system very quickly	4
I found the system very cumbersome to use	1
I felt very confident using the system	5
I needed to learn a lot of things before I could get going with this system	2

Figure B.11: SUS new tool participant 3

I think that I would like to use this system frequently	5
I found the system unnecessarily complex	1
I thought the system was easy to use	5
I think that I would need the support of a technical person to be able to use this system	1
I found the various functions in this system were well integrated	5
I thought there was too much inconsistency in this system	1
I would imagine that most people would learn to use this system very quickly	5
I found the system very cumbersome to use	1
I felt very confident using the system	5
I needed to learn a lot of things before I could get going with this system	1

Figure B.12: SUS new tool participant 4

I think that I would like to use this system frequently	4
I found the system unnecessarily complex	1
I thought the system was easy to use	5
I think that I would need the support of a technical person to be able to use this system	1
I found the various functions in this system were well integrated	5
I thought there was too much inconsistency in this system	1
I would imagine that most people would learn to use this system very quickly	5
I found the system very cumbersome to use	1
I felt very confident using the system	5
I needed to learn a lot of things before I could get going with this system	2

Figure B.13: SUS new tool participant 5

Appendix C

Interview transcript

C.1 Participant 1

I: Hva likte du med det nye systemet?

P: Det var mye som var bedre med det enn Inspera. For det første sparte jo det veldig mye tid for å lage en oppgave. Det var lett å bruke og å forstå hvordan man bruker det, og jeg trengte ikke å tenke så mye mens jeg jobbet.

I: Hva likte du ikke med det nye systemet?

P: Det er ikke så mye negativt å si om det egentlig, men må jeg si noe så er det brukergrensesnittet som kunne vært bedre.

I: Er det noe spesifikt da du tenker kunne vært bedre?

P: Det kunne muligens vært større? eller jeg kunne sikkert bare zoomet inn mer når jeg tenker meg om. Kanskje det hadde vært bedre å ha alt midtstilt på skjermen istedenfor på siden. Kanskje andre farger, jeg vet ikke hehe.

I: Hvilke andre funksjoner kunne du ønsket å ha i det nye systemet?

P: Hmm, det er litt vanskelig å svare på uten å prøvd å lage andre typer oppgaver også. Jeg vet ikke.

I: Er det noen aspekter som er bedre i Inspera enn det andre verktøyet?

P: Ingenting var bedre i Inspera, det var bare veldig forvirrende å bruke. Det har flere funksjoner kanskje men usikker.

I: Har du noen andre forslag eller kommentarer om enten det nye verktøyet eller Inspera?

I: I inspera trenger man mye hjelp eller tid til å lære å bruke det, det er unødvendig tungt. Det er veldig inkonsistent, i at man for eksempel trykker i setup instillingene at GAP elementene skal være samme størrelse men de blir ikke det før man så justerer størrelsen på et av elementene, og så alle nye gap elementer man legger til får en annen størrelse og må justeres på nytt. For å unngå dette er jo man nødt til å ikke justere størrelse før alle elementene er laget og klar. Dette kan man ikke vite før man har prøvd seg frem og gjort mange ting i Inspera.

C.2 Participant 2

I: Hva likte du med det nye systemet?

P: Må si det var veldig enkelt å lage de boksene, og de ble mye mer nøyaktig her enn i inspera. I inspera var det mye fikling på pikselnivå for å få til grei størrelse og posisjon på de. Veldig greit å ha en knapp som gjorde mye automatisk.

I: Hva likte du ikke med det nye systemet?

P: Syns det var litt rare knapper, med tanke på utseende. Kanskje du kunne brukt noe standard UI bibliotek eller noe for å få det til å se finere og mer profesjonelt ut. Ellers var det ikke så mye annet negativt.

I: Hvilke andre funksjoner kunne du ønsket å ha i det nye systemet?

P: Kanskje det hadde gått an med automatisk opplastning av spørsmålet til Inspera.

I: Ja det er noe jeg har tenkt på faktisk, Inspera har et API men mangler akkurat det med importering.

P: Da blir det nok litt vanskelig ja. Kan muligens se på http requestene hos Inspera og kopiere de på en eller annen måte, men uansett stress.

I: Er det noen aspekter som er bedre i Inspera enn det andre verktøyet?

P: Ikke som jeg opplevde i oppgavene. Slipper steget med å eksportere og importere til Inspera da, så har de vel flere funksjoner, selv om jeg ikke brukte de.

I: Har du noen andre forslag eller kommentarer om enten det nye verktøyet eller Inspera?

P: Inspera var utrolig slitsomt å jobbe med, spesielt med de GAP elementene. Teit at nye GAPs ikke blir samme størrelse som de andre når man trykker insert new i Inspera. Det hadde vært kult å ha noe som det nye verktøyet inni inspera, ihvertfall at noen av stegene var lettere eller automatisert.

C.3 Participant 3

I: Hva likte du med det nye systemet?

P: Likte at den hadde auto detekting, at den liksom lagde spørsmål og svar uten å måtte gjøre mye selv. Det sparte tid og energi. Det var lett å sjekke om ting er riktig, i motsetning til inspera som viste feil størrelse på boksene med mindre man så på preview hele tiden. Når man sparer mer tid på å lage et spørsmål får man da mer tid som kan brukes på å sjekke at spørsmålet er bra og teste det ordentlig.

I: Hva likte du ikke med det nye systemet?

P: Hmm det var vel UI da. Det kunne ha vært mer moderne, kunne fått et moderne touch hvis du skjønner. Ikke nødvendigvis mer brukervennlig for det var det, men mer appealing, se litt finere ut. Har du hørt om Material-UI? Det er et bibliotek til react for UI som ser veldig bra ut, det er veldig populært.

I: Hvilke andre funksjoner kunne du ønsket å ha i det nye systemet?

P: Kanskje det hadde gått an med enda mer automatisering? Få automatisk generert spørsmål utifra tema eller informasjon. Det kunne jo gitt mange forskjellige spørsmål til studenter så de kan øve seg, eller til eksamen.

I: Veldig bra svar. Det er noe jeg har tenkt på ja absolutt. Det heter AIG, automated item generation.

P: Ja serr? For det hadde jo vært sykt nyttig da, så kunne studenter øvd på et tema med mange oppgaver og få feedback med en gang på det de gjorde feil osv.

I: Er det noen aspekter som er bedre i Inspera enn det andre verktøyet?

P: Ikke noe annet enn UI, ditt system var mye mye bedre enn Inspera. Inspera var helt forferdelig å bruke, verste systemet jeg har brukt noensinne haha.

I: Har du noen andre forslag eller kommentarer om enten det nye verktøyet eller Inspera?

P: Inspera burde fjerne popup boksene som kommer, er helt unødvendig og irriterende. Inspera har mange unødvendige steg som gjør det jævlig å bruke. UI i ditt verktøy kan forbedres, og få mer funksjonalitet etterhvert som som automatisk lagde oppgaver som vi snakket om. Prosessen i å lage oppgaver i inspera er sykt tungvint. Tja, har vel ikke mer si da. Det andre systemet var mye bedre.

C.4 Participant 4

I: Hva likte du med det nye systemet?

P: Det var nice å bare trenge å trykke på 3 knapper hehe. Jeg slapp å dra ting rundt omkring og fikse på størrelsene, og slapp å skrive inn teksten for hvert element. Jo mer du kan kutte bort jo bedre.

I: Hva likte du ikke med det nye systemet?

P: Hmm, vet ikke jeg. Kanskje knapper jeg ikke vet hva betyr?

I: Noe spesifikt som kunne vært bedre da? Var det dårlig navn på knappene?

P: Nja, kanskje hatt mer forklaring på hva de gjør i oppgaven eller noe.

I: Hvilke andre funksjoner kunne du ønsket å ha i det nye systemet?

P: Det kunne jo vært en knapp til å direkte importere til Inspera. Utenom det, tooltip på alle knappene, så kan man holde musen over og så står det mer om hva de betyr.

I: Er det noen aspekter som er bedre i Inspera enn det andre verktøyet?

P: Jeg antar den har mer funksjonalitet, men man kan jo forsåvidt bare importere spørsmålet man har laget til inspera og så jobbe med det videre der, men da må man bruke 2 forskjellige systemer.

I: Har du noen andre forslag eller kommentarer om enten det nye verktøyet eller Inspera?

P: Nei egentlig ikke.

C.5 Participant 5

I: Hva likte du med det nye systemet?

P: Jeg likte at ting gikk automatisk, det var få antall klikk for å gjøre samme greien. Da ble det mye raskere. Ting var ikke gjemt sånn som i inspera. Der var jo menyer og knapper skjult og man måtte trykke på områder rundt omkring for å få de frem. Mens her var alt synlig.

I: Hva likte du ikke med det nye systemet?

P: UI-en så ikke helt clean ut. Altså så ikke helt ferdig ut for å si det på den måten. Ting var plassert i øverste hjørne istedenfor å være midtstilt som det som oftest er på nettsider. Så var det fargevalg da, kunne vel ha vært bedre, hvilke farger vet jeg ikke, men sikkert noe annet som hadde vært bedre.

I: Hvilke andre funksjoner kunne du ønsket å ha i det nye systemet?

P: Er litt vanskelig å svare på da, siden jeg ikke har mye erfaring å lage sånne oppgaver. Nei jeg vet ikke jeg.

I: Er det noen aspekter som er bedre i Inspera enn det andre verktøyet?

P: Nei egentlig ikke, UIen var ikke noe bedre i Inspera haha.

I: Har du noen andre forslag eller kommentarer om enten det nye verktøyet eller Inspera?

P: Det var mye småting med Inspera som var merkelig og helt unødvendig. For eksempel ting som hoppet rundt. Du vet når man laster opp et bilde?

I: Ja?

P: Så er det en progressbar før man trykker insert. Hele progressbaren forsvant plutselig og da flyttet insert knappen seg opp haha. Går jo ikke det. Man holder jo gjerne musen over knappen mens man venter på progressen skal bli ferdig, men så må man her flytte musen igjen liksom.

P: En annen ting var at om man trykker på insert for fort etter å ha valgt og lastet opp bilde, så forsvinner hele greien og man må gjøre det på nytt. Også, vet ikke om det var jeg som gjorde det feil eller hva, men når man har laget en ny gap og så skal legge til correct answer så var forrige gap sitt correct answer der.

P: Siden det var flere steg i inspera og mindre i det nye var jeg mer selvsikker med å bruke

det nye systemet og det var mindre feil jeg kunne gjøre.

Appendix D

Repository

The project repository is available here: <https://github.com/marcushauge/qti-editor>

